

Split-and-Fit: Learning B-Reps via Structure-Aware Voronoi Partitioning

YILIN LIU, Shenzhen University, China and Simon Fraser University, Canada

JIALE CHEN, Shenzhen University, China

SHANSHAN PAN, Shenzhen University, China

DANIEL COHEN-OR, Tel Aviv University, Israel

HAO ZHANG, Simon Fraser University, Canada and Amazon, Canada

HUI HUANG*, Shenzhen University, China

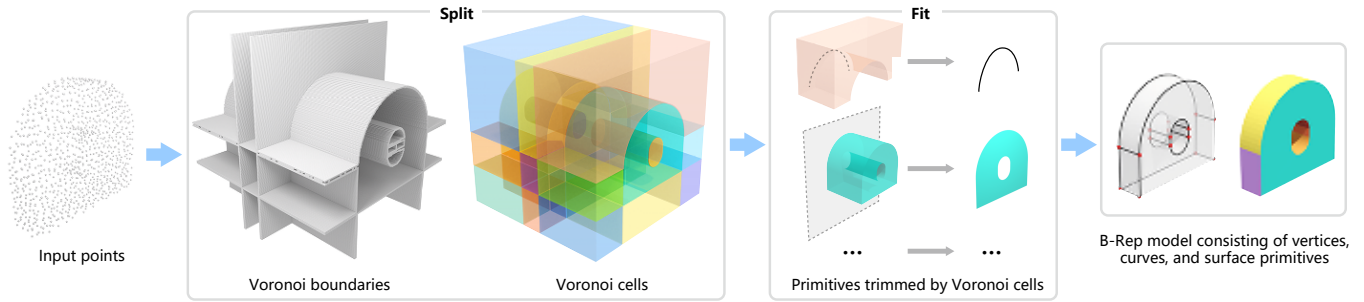


Fig. 1. **Split-and-fit** offers a completely new perspective for acquiring B-Reps for 3D CAD shapes, e.g., from a point cloud. We first obtain a *spatial partitioning* of the volumetric space (i.e., the “split”) and then fit a *single* primitive per partition. This is realized with the classical Voronoi diagrams.

We introduce a novel method for acquiring boundary representations (B-Reps) of 3D CAD models which involves a two-step process: it first applies a *spatial partitioning*, referred to as the “split”, followed by a “fit” operation to derive a single primitive within each partition. Specifically, our partitioning aims to produce the classical *Voronoi diagram* of the set of ground-truth (GT) B-Rep primitives. In contrast to prior B-Rep constructions which were bottom-up, either via direct primitive fitting or point clustering, our Split-and-Fit approach is *top-down* and *structure-aware*, since a Voronoi partition explicitly reveals both the number of and the connections between the primitives. We design a neural network to predict the Voronoi diagram from an input point cloud or distance field via a binary classification. We show that our network, coined NVD-Net for neural Voronoi diagrams, can effectively learn Voronoi partitions for CAD models from training data and exhibits superior generalization capabilities. Extensive experiments and evaluation demonstrate that the resulting B-Reps, consisting of parametric surfaces, curves, and vertices, are more plausible than those obtained by existing

alternatives, with significant improvements in reconstruction quality. Code will be released on <https://github.com/yilinliu77/NVDNet>.

CCS Concepts: • **Computing methodologies** → **Shape inference**.

Additional Key Words and Phrases: Neural Voronoi diagram, CAD modeling, boundary representation

ACM Reference Format:

Yilin Liu, Jiale Chen, Shanshan Pan, Daniel Cohen-Or, Hao Zhang, and Hui Huang. 2024. Split-and-Fit: Learning B-Reps via Structure-Aware Voronoi Partitioning. *ACM Trans. Graph.* 43, 4, Article 108 (July 2024), 27 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnn>

1 INTRODUCTION

Computer-aided design (CAD) models play critical roles in design, engineering, manufacturing, and robotics applications. The de facto and preferred shape representation for general 3D CAD models is the *boundary representation* or B-Rep [Fayolle and Friedrich 2023; Jayaraman et al. 2023; Lambourne et al. 2021]. A B-Rep describes a 3D solid by explicitly defining the limits of its volume in a structured and compact way through parametric surfaces, curves, vertices, and their topological relations. The wide use of B-Reps for CAD modeling and editing has generated much interest in B-Rep reconstruction from unstructured inputs such as point clouds or distance fields.

Classical approaches to B-Rep modeling over point clouds typically resort to clustering or primitive fitting via RANSAC [Fischler and Bolles 1981; Li et al. 2011; Schnabel et al. 2007], region growing [Lafarge and Mallet 2012], or variational shape approximation [Cohen-Steiner et al. 2004; Skrodzki et al. 2020; Yan et al. 2012]. Recent deep learning-based methods employ deep features for instance segmentation [Huang et al. 2021; Li et al. 2023b; Sharma et al. 2020; Yan et al. 2021] or direct primitive detection [Guo et al. 2022]. All of these approaches are bottom-up and based on *local*

*Corresponding author: Hui Huang (hhzhiyan@gmail.com)

Authors’ addresses: Yilin Liu, whatsevenyl@gmail.com, Shenzhen University, China and Simon Fraser University, Canada; Jiale Chen, chenjiale0303@gmail.com, Shenzhen University, China; Shanshan Pan, psshappystar@gmail.com, Shenzhen University, China; Daniel Cohen-Or, cohenor@gmail.com, Tel Aviv University, Israel; Hao Zhang, hao.r.zhang@gmail.com, Simon Fraser University, Canada and Amazon, Canada; Hui Huang, hhzhiyan@gmail.com, College of Computer Science & Software Engineering, Shenzhen University, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 Association for Computing Machinery.

0730-0301/2024/7-ART108 \$15.00

<https://doi.org/10.1145/nnnnnnnn.nnnnnnn>

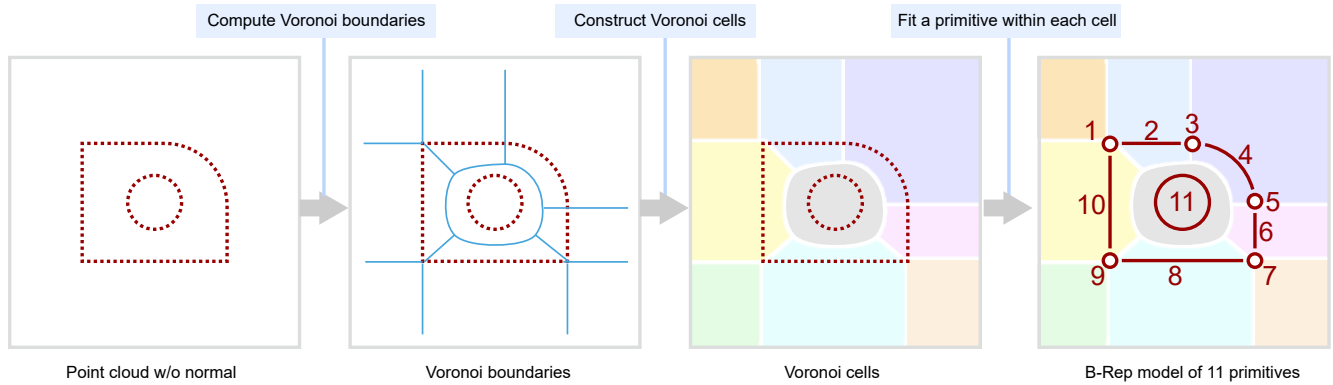


Fig. 2. Overview of our method. Given an input model (i.e., a point cloud, mesh, or distance field), we first compute the Voronoi boundaries of the underlying shape and subsequently construct the Voronoi cells. Then we fit an elementary primitive for each Voronoi cell individually, with the cell boundary serving to naturally trim the primitive, and extract their corresponding connectivity from the Voronoi diagram. Finally, we reconstruct the CAD model (5 vertices, 4 lines and 2 circles in the figure) in B-Rep by combining the primitives and their topological relations.

geometric or topological features without explicit optimization or supervision with respect to global structural properties such as primitive counts. Unsupervised learning of constructive solid geometry (CSG) trees [Daxuan Ren et al. 2021; Kania et al. 2020; Yu et al. 2023, 2022] has found some recent success for CAD modeling. However, all of these methods are trained to minimize the reconstruction error, which is not strongly tied to CSG tree optimization. Technically, there are infinitely many CSG trees which would yield zero reconstruction error. This causes an inherent *ambiguity* in the optimization setup. As a result, the obtained CSG constructions often contain unnatural and an excess of redundant primitives.

In this paper, we introduce a new perspective on acquiring B-reps for 3D CAD models which represents a significant departure from conventional approaches. Instead of directly operating on an input cloud, either via clustering, segmentation, or primitive fitting/detection, we first perform a *spatial partitioning* of the volumetric space in a “split” operation. This is followed by a “fit” step to derive a *single* primitive within each partition to reproduce a primitive in the *ground-truth (GT) B-Rep representation* of the input point cloud. Specifically, we enforce our spatial partitioning to attain a *unique* and well-defined target, which is the classical *Voronoi diagram* of the set of GT B-Rep primitives. In contrast to prior bottom-up B-Rep constructions, our Split-and-Fit approach is *top-down* and *structure-aware*, since a Voronoi partition explicitly reveals both the number of and the connections between primitives.

We design a neural network to infer the Voronoi diagram from an input point cloud or distance field by training a binary classifier, which predicts whether each voxel in 3D space lies on the boundary of a Voronoi cell or not based on local features. We show that our network, coined NVD-Net for neural Voronoi diagrams, can effectively learn Voronoi partitions for CAD models from training data. By confining the per-cell single primitive fitting to Voronoi cell boundaries, the primitives are automatically trimmed without needing a separate, intricate process as in other works such as ComplexGen [Guo et al. 2022]. Also importantly, our learning problem does not suffer from the ambiguity issues for neural CSG constructions, since our training target, the GT Voronoi diagram, is unique.

Our contributions can be summarized as follows:

- Introducing Split-and-Fit, a novel paradigm for B-Rep reconstruction through spatial partitioning of volumetric space, which is top-down and structure-aware.
- NVD-Net, a deep neural network for neural Voronoi diagram prediction from point clouds or distance fields.
- An efficient scheme to extract B-Rep surfaces, curves, vertices, and their connectivities from a Voronoi diagram.

We train our NVD-Net and test our B-Rep reconstruction method on the ABC dataset [Koch et al. 2019], which offers GT B-Reps. Extensive experiments and evaluation demonstrate that compared to state-of-the-art alternatives, our method produces more plausible B-Rep constructions with lower geometric errors, higher topological consistency, and improved instance identification, while exhibiting superior generalization capabilities over unseen 3D shapes.

2 RELATED WORK

The reconstruction of CAD models has been widely studied in the past decades. Early attempts either use analytical or learning-based methods to fit individual primitives from the input point cloud. Recently, some methods have been trained to predict intermediate representation to facilitate CAD reconstruction. We summarized the three categories of methods in the following.

Shape fitting. Traditional methods usually reconstruct CAD models by detecting the primitives (e.g. planes, cylinders, spheres, cones, torus, etc.) from input point clouds. They either use RANSAC [Fischler and Bolles 1981; Schnabel et al. 2007] or Region Growing [Lafarge and Mallet 2012; Oesau et al. 2016] to detect these primitives, which are then merged into a single CAD model by inferencing the topologies between them [Bauchet and Lafarge 2020; Nan and Wonka 2017]. Li et al. [2011] proposed a method to further constrain the detected primitives with the global relation between primitives. Also, variational shape approximation [Cohen-Steiner et al. 2004; Skrodzki et al. 2020; Yan et al. 2012] has been adopted to reconstruct primitives from point clouds. The explicit optimization of the shape partitions enables better modelling of the shape relations.

Additionally, Attene et al. [Attene and Patanè 2010] adopt a "fit-and-segment" pipeline, iteratively clustering triangle faces based on local primitive fitting. This process enables the extraction of primitives from the underlying CAD models based on the generated clusters. Further details can be found in the survey on basic shape primitive fitting [Kaiser et al. 2019]. However, these methods individually address geometric and topology properties, making the reconstruction process easily stuck in a local minimum.

Point segmentation. Recent learning-based methods [Fu et al. 2023; Huang et al. 2021; Li et al. 2019a,b, 2023b; Sharma et al. 2020; Yan et al. 2021] used a "segment-and-fit" pipeline to reconstruct CAD models from point clouds. These methods train a point-based neural network to perform instance segmentation on the input point cloud, and then fit primitives to the segmented clusters. Specifically, SPFN and ParSeNet [Li et al. 2019a; Sharma et al. 2020] embed the input points into a representation space, where the feature code for points that belong to the same primitive are closer to each other. After a mean-shift clustering, the clustered points are fed into the fitting module to produce the final geometry. HPNNet [Yan et al. 2021] and SEDNet [Li et al. 2023b] further enhance the representation learning by hybrid shape descriptors and multi-stage feature fusion mechanisms. NerVE [Zhu et al. 2023] also proposed directly predicting the structured edges instead of the pure point segmentation. Meanwhile, Point2CAD [Liu et al. 2023] further employs an analytic-neural reconstruction method to fit and recover the structured CAD models based on the segmentation. However, all these methods have been designed to approximate the underlying shape in a bottom-up manner, which is also prone to local minima. The mixed combinatorial assignment of point labels and continuous parameter fitting also makes the learning process unstable.

Direct CAD learning. Instead of fitting primitives, some other representations have been adopted for CAD reconstruction. Li et al. [2023a] and Lambourne et al. [2022] attempted to learn 2D sketches and 3D extrusion parameters from point clouds, which is conventional in traditional CAD modelling. DEF [Matveev et al. 2022] learns to construct a distance-to-feature field to represent the input range scan and then reconstruct feature curves in CAD models. BSPNet [Chen et al. 2020] was trained to predict a set of planes to build a binary space partitioning tree, where the learning process is motivated by minimizing a reconstruction error. The resulting planes can be assembled into a watertight mesh to represent the underlying shape. CAPRINet [Yu et al. 2022] and D2CSG [Yu et al. 2023] further enhance the partitioning by supporting quadric primitives and performing single-object optimization to produce better CSG-Trees. Additionally, Xu et al. [Xu et al. 2021] present a method to infer and reconstruct the modelling sequence of a CAD model by utilizing zone graphs to represent the spatial partitioning induced by the model's BRep faces. However, while the reconstruction error is minimized, the structure of the CSG-Tree is often far from optimal.

Recent advances in autoregressive models and transformer architectures have paved the way for the direct generation of CAD models in Boundary Representations (B-Reps). PolyGen [Nash et al. 2020] employs a transformer-based pointer network to sequentially

generate mesh vertices and faces, with its probabilistic model design facilitating the creation of novel structures from diverse inputs. Building upon this, SolidGen [Jayaraman et al. 2023] extends the capabilities to B-Rep CAD models, incorporating elementary surfaces such as cylinders or cones. Utilizing the Indexed Boundary Representation framework, SolidGen methodically produces vertices, curves, faces, and subsequently converts them into B-Rep models. Furthermore, MeshGPT [Siddiqui et al. 2023] has introduced an autoregressive generation of compact meshes with sharp edges. Unlike direct prediction of surfaces and curves, MeshGPT's learning process operates in a pre-quantized latent space. However, while these methods succeed in generating compact meshes, they usually lack parametric surfaces and curves [Nash et al. 2020; Siddiqui et al. 2023], which are essential for subsequent editing and rendering tasks. Although SolidGen [Jayaraman et al. 2023] outputs parametric surfaces, aligning the latent code of the condition with the generative model remains a challenge, often leading to outputs that are inconsistent with the input conditions and lacking in detail.

In contrast, ComplexGen [Guo et al. 2022] formulates CAD reconstruction as a detection task, directly determining the parameters and topologies of each primitive via a conventional object detection framework. This approach includes an optimization step to refine the topology and geometry of the inferred primitives. Nonetheless, their learning process is prone to local minima due to the indeterminate number of primitives, and the mixed topological and geometric optimization complicates the reconstruction process.

While *Point-based* Voronoi diagrams are prevalent in computer graphics and computational geometry, with applications in mesh reconstruction [Alliez et al. 2007; Maruani et al. 2023], Medial Axis Transform (MAT) computation [Wang et al. 2022], and mesh simplification [Liu et al. 2015], these diagrams have seen limited application in CAD model reconstruction. SEG-MAT [Lin et al. 2022] utilizes a form of Voronoi Diagrams, the Medial Axis Transform (MAT), for shape representation and segmentation. However, this approach is restricted to part segmentation and does not generate structured CAD models. Our method leverages the *primitive-based* Voronoi diagram as an intermediary representation in reconstructing B-Rep models. The construction of the Voronoi diagram, achieved through a simple binary classification of Voronoi boundaries, along with the straightforward inference of topological relations from the connectivity of Voronoi cells, significantly reduces ambiguity in the learning process.

3 PRELIMINARIES

Definition of the Voronoi diagram over B-Reps. Unlike most previous methods [Alliez et al. 2007; Liu et al. 2015; Maruani et al. 2023; Wang et al. 2022; Williams et al. 2020], which define Voronoi diagrams on point sets, our Voronoi diagram is defined on primitives (vertices, curves and surfaces). Similar to SOTA methods [Chen et al. 2020; Guo et al. 2022; Yu et al. 2023, 2022], our primitives are restricted to planes, spheres, cylinders, cones, torus as surfaces and lines, circles, ellipses as curves. Given a set of primitives, the Voronoi diagram $G_v(N_v, E_v)$ is a partitioning of the volumetric space, which consists of a set of adjacent Voronoi cells N_v . Here, E_v is the

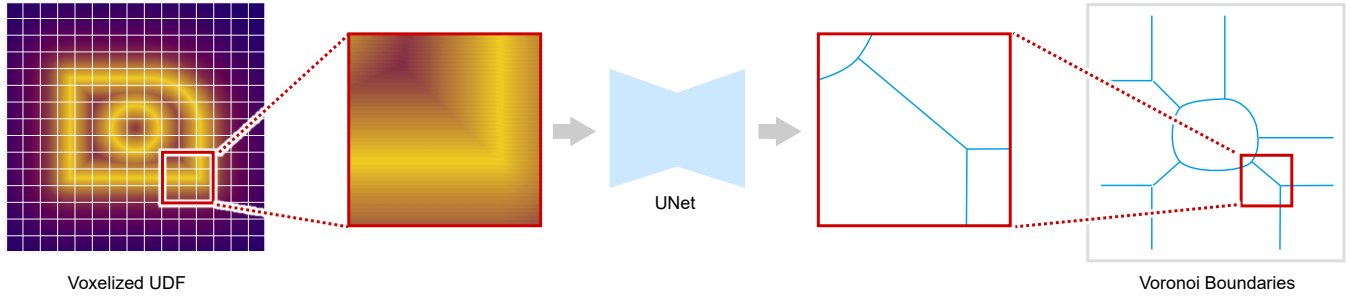


Fig. 3. Overview of our network. We first voxelize the input UDF field. Each voxel contains 4 features (d, g_x, g_y, g_z) which indicates the UDF value and the gradient vector of the UDF field. The voxel grid is split into overlapped local patches and fed into the standard UNet individually. A single channel binary flag is predicted within each voxel to indicate whether this voxel contains any Voronoi boundary.

adjacent matrix that indicates whether two Voronoi cells are neighbouring. Based on the bisection property of Voronoi diagrams, any point inside one cell should always have the closest distance to its corresponding primitive than to other primitives. Since the primitives and their corresponding Voronoi diagram are dual structures, we do not need to store explicit information about the primitives (e.g., type, parameters, etc.). Instead, we store the dual Voronoi diagram. Storing the dual structure gives us the freedom to adapt to various types of primitives during learning. More importantly, it allows us to convert a surface-based BRep model to a volumetric representation (described in Section 4). Learning on a surface representation is vulnerable due to a lack of a suitable representation to simultaneously indicate a mixture of geometric, topological, and combinatorial features. In contrast, learning a volumetric representation through binary classification is more straightforward and robust. Additionally, unlike the CSG-Tree used in previous CAD reconstruction methods [Chen et al. 2020; Yu et al. 2023, 2022], the Voronoi diagram remains unique, reducing ambiguity in the training process.

Problem Statement and notions. The input to our method can either be a sparse 3D point cloud with or without normals, i.e., $P = \{p_i\}_{i=1}^k$, where $p_i \in \mathbb{R}^3$ or \mathbb{R}^6 , a mesh, or a continuous distance field $f: X \rightarrow \mathbb{R}$. Our goal is to reconstruct the *unique* Voronoi diagram of the GT primitives, which is then used to extract primitives and their connectivities for B-Rep model reconstruction (as shown in Fig. 2). Notationally, we denote a B-Rep model as $M(V, E, F, \partial, \mathcal{P})$, where $V = \{v_i\}_{i=1}^k$ is a set of vertices, $E = \{e_i\}_{i=1}^k$ is a set of edges, and $F = \{f_i\}_{i=1}^k$ is a set of surface patches. $\partial_n, n = 1, 2$, represents the topological relations between vertices, edges, and surface patches (e.g., $\partial_2 f_i \in E$ represents the boundary of surface patch f_i). We additionally recover the connectivity between surface patches to further facilitate the reconstruction of curves [Li et al. 2023b]. And \mathcal{P} represents the geometric properties of each primitive.

Method Overview. To compute the Voronoi boundaries for a given point cloud, we first convert the input point cloud into an unsigned voxelized distance function (UDF) field via Neural Dual Contouring [Chen et al. 2022]. As shown in Fig. 3, we train a UNet-like neural network to predict the Voronoi diagram from the UDF field. In order to recover the Voronoi diagram, our network is trained

to identify the Voronoi boundaries from the UDF field (described Sec. 4.1). Based on the predicted Voronoi boundaries, we employ a region-growing strategy to extract the Voronoi cells and their connectivities. Inside each Voronoi cell, we fit a primitive using an analytical method and then reconstruct the CAD models in B-Rep by combining the primitives and their topological relations, described in Sec. 4.2.

4 METHODOLOGY

4.1 Voronoi Diagram Prediction

Our Voronoi predictor takes a voxelized UDF field f as input and predicts the Voronoi diagram $G_v(N_v, E_v)$. We use UDF fields as input instead of previously used point clouds [Guo et al. 2022; Li et al. 2023b; Sharma et al. 2020], since a UDF is a continuous function defined over the entire volumetric space, which is better suited for our space partitioning problem. Also, point clouds can be easily converted into UDF fields [Chen et al. 2022].

Given a UDF field f , we want to predict a binary field $b: X \rightarrow \{0, 1\}$, where $b(x) = 1$ indicates that x is on the Voronoi boundary, and 0 otherwise. In order to train such a network, we first discretize the UDF field f into a voxelized grid $V_{r \times r \times r}$ with a fixed resolution r ($r = 256$ in all our experiments). Each voxel contains a 4-channel feature code (d, g_x, g_y, g_z) , where d is the UDF value and g_x, g_y, g_z are the first-order derivatives of the UDF field, also denoted as the gradient vector. Followed by a UNet-like network $F(V)$, the 4-channel feature code is mapped to a binary flag b for each voxel, indicating whether it is on the Voronoi boundary; see Fig. 3. The detailed structure of the network is shown in the supplementary material.

Inspired by Neural Dual Contouring [Chen et al. 2022], we only leverage local features to predict Voronoi boundaries. We split the whole voxel grid into N local patches with a fixed stride s and size k ($s = 16$ and $k = 32$ in all our experiments). Note that the local patches are overlapped, where each voxel can be contained in multiple such patches. Patch overlapping can ensure that the prediction on the boundary of the local patches is consistent. The network is trained to minimize the focal loss [Lin et al. 2020] between the predicted flag b and the ground truth flag b^* ,

$$\mathcal{L}_{focal} = -\alpha(1 - b^*)^\gamma \log(1 - b) - (1 - \alpha)b^* \log(b), \quad (1)$$

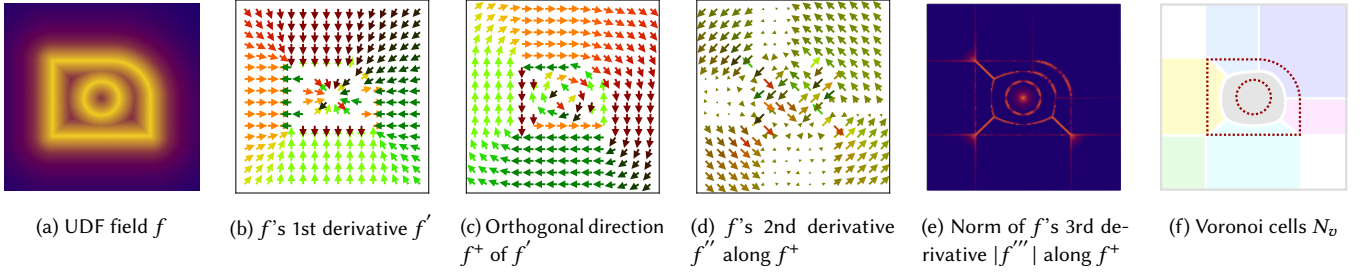


Fig. 4. The relation between the Voronoi boundaries and the derivative of the UDF field.

where α and γ are the hyper-parameters of the focal loss. Please refer to the supplementary material for more details about the data preparation and training process.

Generalizability. It is worth noting that ComplexGen [Guo et al. 2022] leverages a triple-branch transformer architecture to predict the parameters and topologies of each primitive. Different cross-attention modules ensure full information exchange between the voxel and primitive features. However, the comprehensive information exchange also makes the learning process hard to optimize, which might lead to poor generalizability; see Sec. 5.3 for further details and a comparison. In contrast, our prediction of Voronoi diagrams only relies on local geometric cues, which can significantly improve the model’s generalizability.

Discontinuity over UDF. Estimating Voronoi boundaries is similar to identifying discontinuities over the UDF field, which only relies on local geometric features; see Fig. 4. Overall, the distance field of a CAD model is a piece-wise smooth function. The discontinuity of the distance field is usually related to a change of the nearest primitive. To find the discontinuity, we first calculate the derivative of the UDF field. Fig 4(a) and Fig 4(b) show the original UDF field and the first-order derivative. We can see that for points whose nearest primitive is a plane, their gradient vectors are parallel. Thus, the discontinuity is always located at the joint of two planes.

We further calculate the second- and third-order derivatives of the UDF field to identify the joint between two quadric primitives. Since the first-order derivative of the UDF field is a vector field, we choose the orthogonal direction of each gradient vector and calculate the second- and third-order derivatives along this direction in Fig 4(d).

We visualize the L_2 -norm of the 3rd derivative in Fig 4(e), and we can see the discontinuity with high L_2 -norm is located at the joint of two quadric primitives, which is consistent with the Voronoi boundaries of the target shape. Thus, identifying the Voronoi boundaries is equal to finding the discontinuity of the second derivative of the UDF field, as shown in Fig 4(f). However, real-world UDF fields are usually noisy, which makes the discontinuity harder to identify. Thus, we choose to predict the Voronoi boundaries using a neural network to approximate this process, leading to improved robustness against minor local disturbances in the input data.

4.2 Extraction of primitives and topologies

Based on the predicted Voronoi boundaries, we first conduct region growing in the volumetric space to reconstruct the Voronoi cells

N_v . We select a seed voxel v_s and grow the region by adding the neighbour voxels v_n with flag $b(v_n) = 0$. The region growing process is repeated until all the neighbour voxels are added. The voxel grids will be clustered into regions, denoted as the Voronoi cells N_v . The connectivity of the Voronoi cells E_v can be easily inferred from neighbourhood relations of N_v .

By definition of the Voronoi diagrams, each Voronoi cell contains exactly one primitive. Based on the UDF field, we use the least-square method [Eberly 2000] to fit a primitive inside each Voronoi cell. For each Voronoi cell, we iterate over all the possible primitive types and choose the one with the lowest fitting error. Note that the fitting process is only conducted in each Voronoi cell *individually*. Unlike ComplexGen [Guo et al. 2022] and search-based methods [Bauchet and Lafarge 2020; Li et al. 2011], we do not need to consider assigning each point to a specific primitive. Thus, the fitting process is considerably less ambiguous and more robust.

Since the connectivity of the Voronoi cells E_v is already known, we can easily infer the topological relations ∂ between primitives. For each primitive i , we find its adjacent Voronoi cells N_j and examine whether the distance between the point inliers of the two primitives is less than a prescribed threshold.

After extracting the primitives and their topologies, we can reconstruct the CAD models in B-Rep by combining the primitives and their topological relations. Similar to SEDNet [Li et al. 2023b], we additionally calculate the intersection curve between two adjacent surfaces to add missing curves during the fitting process. The resulting B-Rep model can be edited, meshed, and visualized by most CAD software. Please refer to the supplementary material for more details about the detailed explanation and the pseudocode of fitting.

5 EXPERIMENTAL RESULTS

5.1 Dataset and metrics

We benchmark our method against leading CAD model reconstruction techniques on the ABC dataset [Koch et al. 2019]. Our comparative analysis includes state-of-the-art methods such as ComplexGen [Guo et al. 2022], HPNet [Yan et al. 2021], SED-Net [Li et al. 2023b], and Point2CAD [Liu et al. 2023], as well as the traditional fitting method RANSAC [Fischler and Bolles 1981; Schnabel et al. 2007]. Our model was trained on a dataset identical to that used by ComplexGen, HPNet, and SED-Net, comprising approximately 20,000 models featuring elementary and B-Spline primitives.

For a fair and comprehensive assessment, all methods were tested on the same test set of around 1,000 models. This test set included models with surface types such as Plane, Cylinder, Sphere, Cone, and Torus and curve types like Line, Circle, and Ellipse. Notably, B-Spline primitives were excluded from the test set due to observed instabilities in their analytical fitting from point data.

Nevertheless, it is essential to underscore that our primary contribution, the learning-based Voronoi Partition process, is agnostic to the actual primitive type. This is further elaborated in Section 5.5.

Evaluation Metrics: To assess each method’s performance, we employ three metric categories:

- **Geometric Error:** The Chamfer Distance (CD) and Light Field Distance (LFD) [Chen et al. 2003] are utilized for quantifying geometric inaccuracies and visual fidelity discrepancies between the reconstructed and ground truth CAD models. Notably, models with high accuracy but fragmented primitives can still display minimal geometric error. Hence, following ComplexGen [Guo et al. 2022], detection metrics are also employed for a more nuanced performance evaluation.
- **Number of Effective Primitives:** As an additional metric, we report the count of effective primitives in the reconstructed CAD models, providing insight into the model’s structural complexity.
- **Detection Score:** The averaged Precision, Recall, and F1-Score are used to assess the accuracy of primitive instances within the methods. This metric offers a complementary perspective to geometric error, facilitating a direct comparison between predicted and ground truth primitives. We apply multiple thresholds (0.1, 0.05, 0.02, 0.01, 0.005) for a thorough evaluation.
- **Topological Error:** To gauge the accuracy of predicted relationships among surfaces, curves, and vertices, we employ the F1-Score as a measure of topological error.

5.2 Comparison

We sample 10k points on the GT mesh as the input to all the methods. All methods are tested with normal input except for our method. In order to benchmark our approach against three baselines, we first extract the trimmed mesh for each reconstructed primitive. Then we sample points on the surfaces and curves and compute the previously mentioned metrics. For RANSAC, we use the implementation from the CGAL library¹. We report the results of both default parameters (*RANSAC Default*) and the best parameters we tuned (*RANSAC Tuned*), where $\epsilon = \{1\% * bbox_diag, 0.001\}$, $prob = \{0.01, 0.001\}$, $min_points = \{1\%, 0.5\%\}$, $\epsilon_{normal} = \{0.9, 0.9\}$, respectively. For HPNet [Yan et al. 2021] and SEDNet [Li et al. 2023b], we employ Point2CAD [Liu et al. 2023] to extract the mesh from the point segmentation. As for ComplexGen [Guo et al. 2022], we utilize its official implementation for extracting the trimmed models. The points essential for metric computation are derived from the trimmed meshes using Poisson disk sampling. The quantity of these sample points is determined based on the area of the surfaces or the length of the curves. Quantitative comparisons between each

¹https://doc.cgal.org/latest/Shape_detection/index.html

Table 1. The **geometric error** of our method compared with RANSAC, ComplexGen, HPNet+Point2CAD and SEDNet+Point2CAD.

Method	CD ↓			LFD ↓
	Vertex	Curve	Surface	
RANSAC Default	-	-	0.0205	3123
RANSAC Tuned	-	-	0.0162	2502
ComplexGen	0.0901	0.0601	0.0402	4280
HPNet+Point2CAD	0.0782	0.0222	0.0157	2104
SEDNet+Point2CAD	0.0832	0.0268	0.0192	2262
Ours	0.0327	0.0144	0.0093	908

Table 2. The **topological error** between the reconstructed and ground truth B-Rep models. **FE** and **EV** denote the F1-Score of surface-curve connectivity and curve-vertex connectivity, respectively. Results from RANSAC are not available as they do not output topological structures.

Method	FE ↑	EV ↑
ComplexGen	0.599	0.572
HPNet+Point2CAD	0.739	0.674
SEDNet+Point2CAD	0.696	0.647
Ours	0.778	0.753

method are detailed in Table 1, 2, 3, 4. For qualitative analysis, we have chosen 30 **representative** CAD shapes from our test set, showcasing a variety of geometric features such as standard furniture, high-genus structures, and complex compositions of planes, cylinders, cones, and open surfaces. A subset comprising five examples is presented in Fig. 5, with additional details available in the supplementary material. To ensure an unbiased presentation, we have also **randomly** selected 30 cases from the test set for visualization, avoiding any manual cherry-picking. Five of these selections are illustrated in Fig. 6, and further examples are included in the supplementary material.

As evidenced in Table 1, our method surpasses all baselines in performance. The CAD models generated by our approach demonstrate superior accuracy and visually appeal. Additionally, Table 3 indicates that our method outperforms others in detection metrics, excelling in both Precision and Recall. Further, Table 2 underscores our method’s enhanced capability in capturing topological relations. These advantages are also visually apparent in Fig. 5, Fig. 6 and Fig. 7. We observe that detection-focused methods like ComplexGen struggle to accurately reconstruct CAD models, particularly for complex structures that diverge from their training datasets (details can be found in Sec. 5.3). Although HPNet and SEDNet demonstrate a higher degree of generalization, they are prone to inaccurately assigning points to primitives, often resulting in impractical primitive shapes. Furthermore, these methods often incorrectly predict the topological relationships between primitives. Such inaccuracies in primitive identification and topology typically lead to Point2CAD generating CAD models with fragmented and disconnected components. Traditional methods such as RANSAC can accurately fit primitives but require finely tuned parameters. Moreover, they struggle to recover low-dimensional primitives like curves and vertices

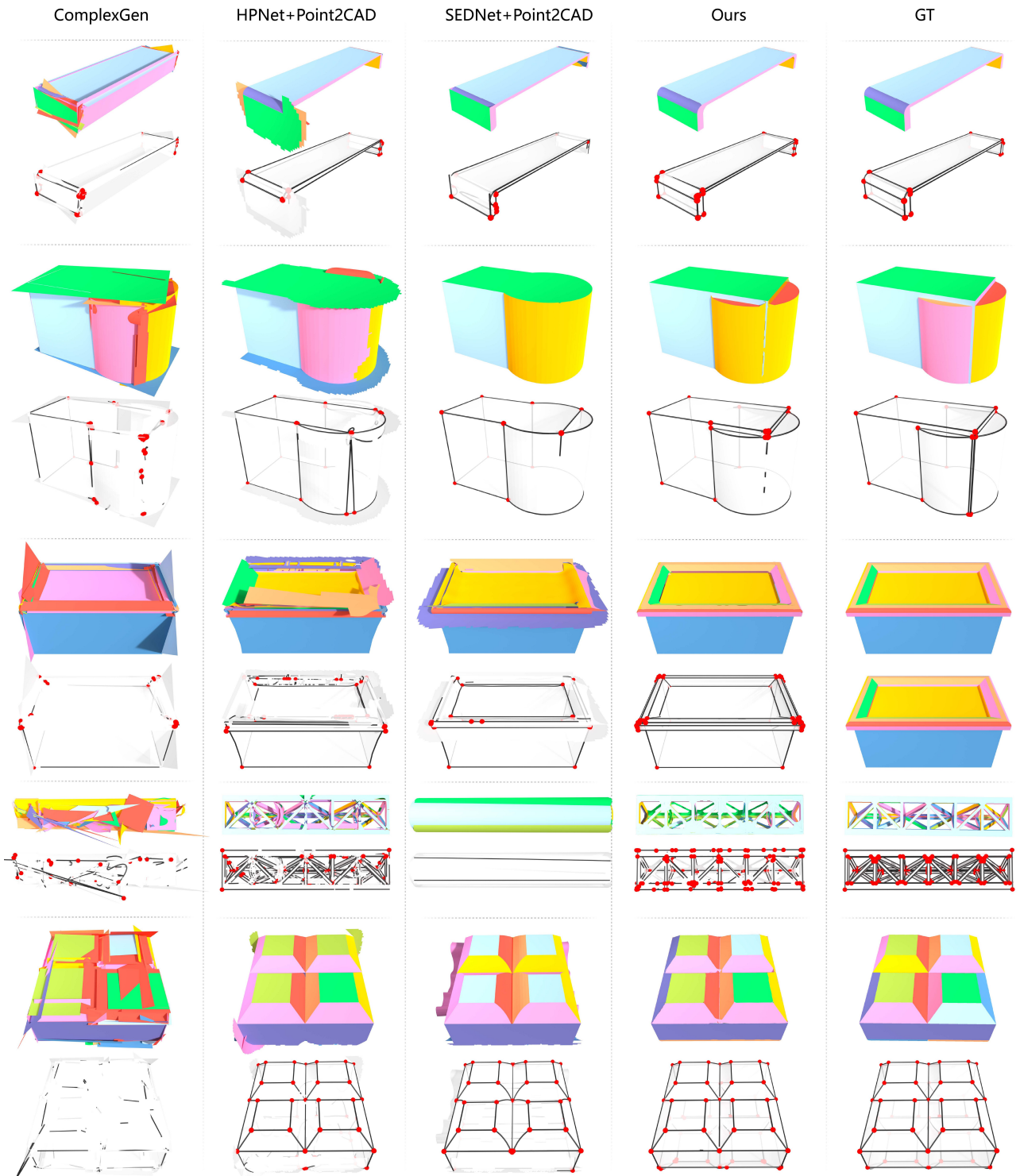


Fig. 5. Qualitative comparisons of 5 **representative** shapes in ABC dataset.

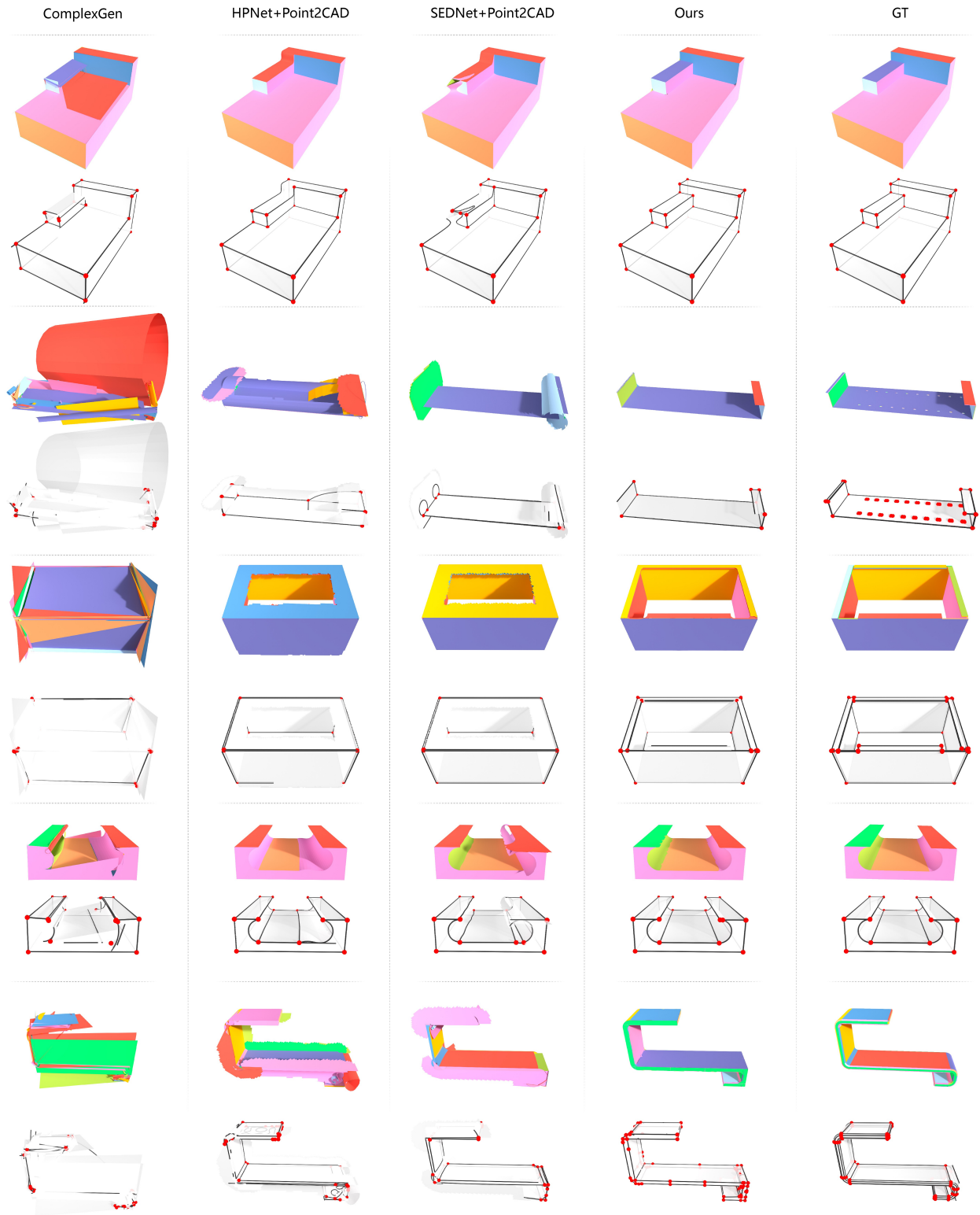


Fig. 6. Qualitative comparisons of 5 randomly sampled shapes in ABC dataset.

Table 3. The **detection score** of vertices(V), curves(C) and surfaces(S) on the ABC dataset [Koch et al. 2019] using different methods. The evaluation process involved a matching process based on the Chamfer Distance, which was calculated between sample points on both the reconstructed and ground truth primitives. To ensure a thorough assessment, we employed a range of thresholds (0.1, 0.05, 0.02, 0.01, and 0.005). These multiple thresholds allowed for a more comprehensive evaluation in calculating Precision, Recall, and the F1-Score. With a chamfer distance threshold of 0.01, we additionally report the number of **good/total** primitives in the table.

Method	Number #			F1 Score \uparrow			Precision \uparrow			Recall \uparrow		
	V	C	S	V	C	S	V	C	S	V	C	S
RANSAC Default	-	-	4.6/10	-	-	0.485	-	-	0.612	-	-	0.462
RANSAC Tuned	-	-	10.4/12.6	-	-	0.722	-	-	0.836	-	-	0.681
ComplexGen	7.9/22.8	16.6/44.9	10.5/25.9	0.5	0.513	0.502	0.521	0.510	0.479	0.505	0.538	0.552
HPNet+Point2CAD	17.7/26.3	23.0/33.8	8.9/12.8	0.671	0.696	0.697	0.749	0.755	0.776	0.649	0.668	0.661
SEDNet+Point2CAD	14.1/21.9	17.1/26.6	6.7/10.5	0.652	0.656	0.646	0.734	0.735	0.742	0.626	0.620	0.605
Ours	29.6/38.5	33.8/41.6	14.2/15.8	0.785	0.790	0.821	0.810	0.850	0.902	0.802	0.766	0.781

Table 4. The **detection error** that directly computed on the input point cloud. Similar to Table 3, we compute the F1-Score, Precision and Recall using multiple thresholds (0.1, 0.05, 0.02, 0.01, 0.005).

Method	F1 Score \uparrow	Precision \uparrow	Recall \uparrow
HPNet	0.614	0.670	0.589
SEDNet	0.588	0.674	0.554
Ours w/o fitting	0.633	0.711	0.602
Ours w/ fitting	0.685	0.757	0.650

and fail to identify the topological relationships among the primitives. In contrast, our method employs the Voronoi diagram as an intermediary representation. The point assignment and the primitive fitting are separated into two individual processes, significantly enhancing both primitive reconstruction and topological accuracy.

In our comprehensive analysis, we extend our evaluation to include a direct assessment of point clouds. This facilitates a more direct comparison of our method with two segmentation-based approaches, HPNet and SEDNet. In Table 4 we show two variants of our method, one with point labels directly segmented from our Voronoi Diagram (*w/o fitting*) and the other with the point labels fitted to the extracted primitives (*w/ fitting*). Our method surpasses HPNet and SEDNet in key metrics such as F1-Score, Precision, and Recall. This further underscores the robustness and adaptability of our approach in diverse evaluation scenarios.

We also conducted a user study to assess the visual fidelity of CAD models reconstructed by different methods. For this study, we selected 20 shapes from our test set, which included 10 shapes from the representative shape set and another 10 from a set of randomly sampled shapes. Participants in the study were presented with the reconstructed CAD models from four different methods, along with their corresponding curves and vertices, and the ground truth for comparison.

The study involved 186 participants who were asked to evaluate and rank the models reconstructed by different methods based on their similarity to the ground truth. This approach allowed us to gauge user perceptions of the quality and accuracy of the reconstructed models. The results of this user study, which provide

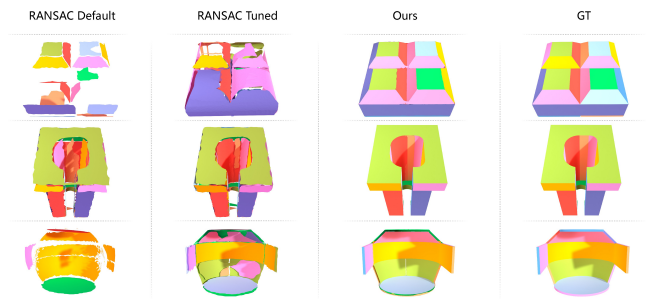


Fig. 7. Qualitative comparisons with RANSAC [Schnabel et al. 2007].

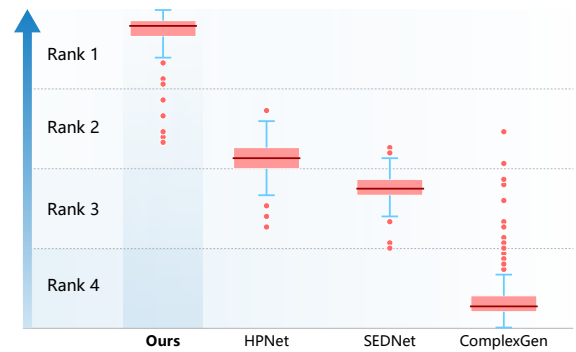


Fig. 8. This is the box plot of our user study. On the Y-axis, we show the average ranking of each method.

insights into the perceived quality of the models generated by each method, are depicted in Fig. 8.

The findings from the study clearly indicate that our method was ranked highest in terms of visual fidelity. This outcome underscores the effectiveness of our approach in producing CAD models that are not only accurate in terms of geometry and topology but also visually appealing to users.

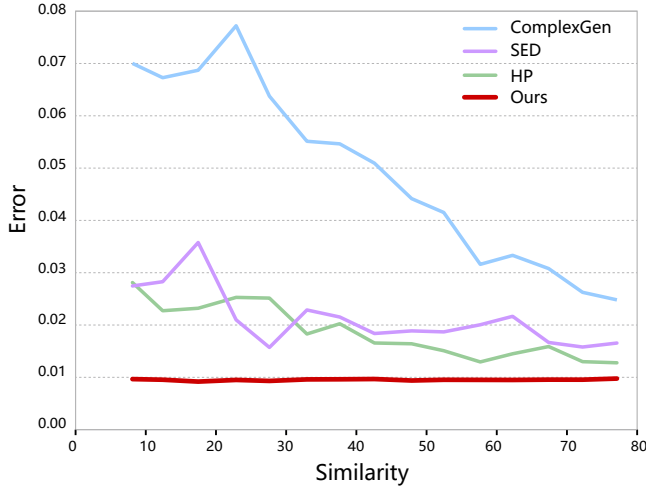


Fig. 9. We plot the generalization ability of all methods on the test set. The similarity between the test model and the training set is sorted, and the corresponding reconstruction error is plotted. The similarity is reported in the range of 0% to 100%, where lower similarity indicates higher chamfer distance and less similarity to the training set. Our method has the best generalization ability of all the methods.

5.3 Generalization Ability

As mentioned above, ComplexGen [Guo et al. 2022] cannot faithfully reconstruct CAD models when the model is complex and less similar to the training set. We conducted a thorough analysis to investigate this issue further and explore the generalization capabilities of different methods. We first identify the model in the training set that is most similar to each model in the test set. We utilized the Chamfer Distance between the corresponding models to quantify this similarity. A lower Chamfer Distance signifies a closer resemblance to the models in the training set, and a higher distance indicates lesser similarity. We organized these similarity metrics in ascending order and plotted them alongside the corresponding geometric errors of each method. Fig. 9 provides a clear illustration of how each method performs in terms of generalization across varying degrees of similarity to the training set.

As demonstrated in Fig. 9, ComplexGen’s performance degrades notably with diminishing shape similarity to the training set, evidenced by an increased reconstruction error. The full information change between the voxel and primitive features in ComplexGen makes the learning process ambiguous, impairing its generalization capacity. In stark contrast, our method maintains consistent performance, irrespective of the model’s similarity to the training dataset. This stability is due, in large part, to our method’s approach to the Voronoi diagram prediction, which simplifies the task to a binary classification problem. This simplification renders our method significantly more robust compared to ComplexGen’s mixed combinatorial and continuous learning process. Furthermore, our approach to predicting the Voronoi diagram relies primarily on local geometric cues, as detailed in Sec. 4.1 This focus on local geometry fosters a more generalizable learning process, enabling our method

to maintain high accuracy and reliability across a wider range of shapes and complexities. This attribute starkly distinguishes our method from others, particularly in scenarios involving diverse and complex CAD models.

Interestingly, we even have a slightly higher error on the most similar shapes (70%-80%). We attribute this to the metric we used to measure the similarity. Chamfer distance from the test shape to training items is not the most ideal choice. For instance, some shapes might have a small Chamfer distance not because they are similar but rather because they share a similar size (e.g., two cubes of differing sizes might have a large chamfer distance even though they have similar appearance).

5.4 Stress Test

We extended our evaluation to more challenging scenarios, including noisy point clouds and real-world scans. For the noisy point cloud tests, we introduced random noise equivalent to 1% of each shape’s diagonal length [Li et al. 2023b]. We directly use the pre-trained models of all methods to reconstruct the noisy shapes. The performance of each method in handling these noisy inputs is presented in Table 5. Our findings indicate that our method still leads in terms of geometric error and detection score, while maintaining comparable topological performance to other methods. This highlights our method’s resilience and accuracy even in the presence of data imperfections.

Regarding real scans, we employed meshes reconstructed from structured light scanners as described in DEF [Matveev et al. 2022]. This allowed us to test our method’s effectiveness on data derived from real-world objects, further extending its applicability. The outcomes of these tests, which showcase the capability of our method to handle real scan data, are visually depicted in Fig. 10. This visual representation underscores our method’s practical utility and robustness in diverse and challenging environments, beyond the controlled conditions of synthetic datasets.

5.5 Limitation

Although our method achieves state-of-the-art performance in CAD model reconstruction, some limitations remain.

Voxelized Voronoi Diagram: The voxelized representation of the Voronoi diagram introduces inherent limitations, such as the resolution constraints of the voxel grid. This can particularly impact thin primitives: in cases where Voronoi boundaries entirely occupy a primitive’s Voronoi cell, both the cell and the corresponding primitive are omitted in subsequent processes. This issue is evident in the first shape of Fig. 6, where a thin plane fails to be recovered, resulting in a zigzag boundary of the yellow nearby plane. Additionally, small holes in the predicted Voronoi boundaries may inadvertently connect Voronoi cells, leading to suboptimal primitive fitting (as shown in Fig. 11).

Noise data: Fig. 10 illustrates that while our method can reconstruct a reasonably accurate B-Rep model to a certain extent, the process of CAD model reconstruction in the presence of noise remains challenging. The complexity stems from the non-uniform

Table 5. The performance of different methods under noisy input. We report the **geometric error** (CD), **detection score** (F1 score) and the **topological error** (F1 score) of different methods.

Method	CD ↓			Detection Score ↑			Topological Error ↑	
	Vertex	Curve	Surface	Vertex	Curve	Surface	FE	EV
ComplexGen	0.1191	0.0660	0.0442	0.460	0.478	0.462	0.546	0.520
HPNet+Point2CAD	0.1078	0.0367	0.0235	0.445	0.492	0.542	0.722	0.657
SEDNet+Point2CAD	0.1044	0.0352	0.0232	0.593	0.614	0.615	0.685	0.629
Ours	0.0539	0.0294	0.0121	0.683	0.683	0.742	0.707	0.673

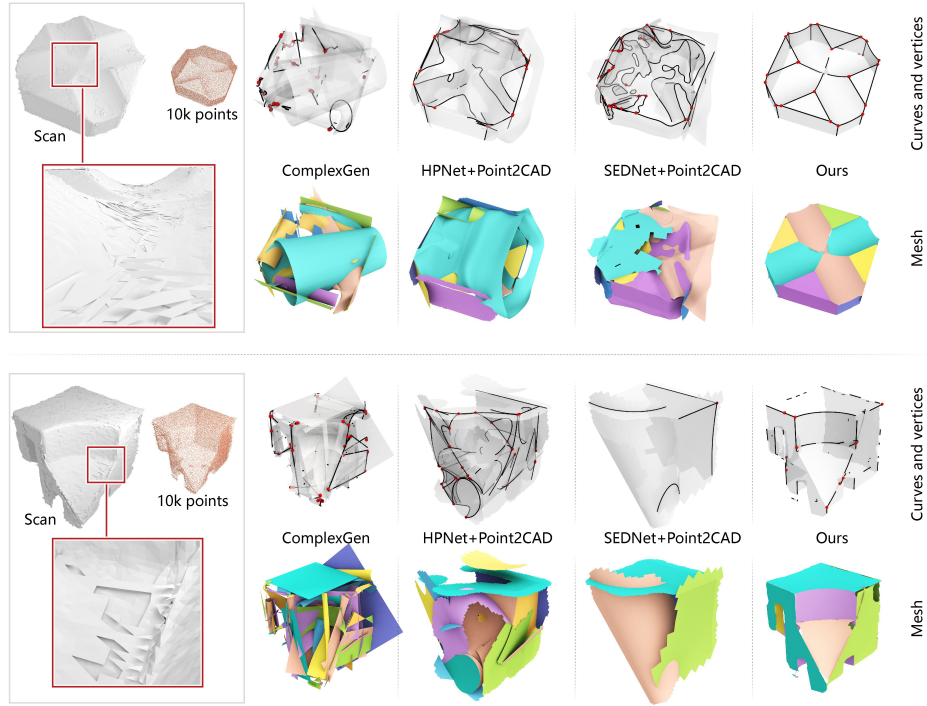


Fig. 10. Qualitative comparison on two real scans. The input models are derived from structured light scanners [Matveev et al. 2022]. These scans are characterized by significant noise and data omissions, including issues like noise accumulation during scanning, self-intersecting triangles, and errors in alignment leading to duplication and overlapping. We standardized the input to 10,000 points for all methods to ensure a fair comparison. The pronounced noise particularly challenges previous methods’ mixed combinatorial and continuous learning processes, resulting in suboptimal segmentation and inaccurate geometric parameters. Despite being trained predominantly on synthetic data, our method demonstrates a superior capacity to manage and interpret noisy data.

nature of noise distribution, which is influenced by a variety of factors including the characteristics of the acquisition device, environmental conditions, and the specifics of the reconstruction algorithm. This diversity in noise sources and patterns makes it challenging to model the noise distribution, thereby complicating the training of a universally robust model capable of handling all types of noise.

B-Spline Fitting: While advancements have been made in fitting B-Spline primitives from unstructured point clouds [Liu et al. 2023; Sharma et al. 2020; Zheng et al. 2012], the process remains unstable and occasionally yields implausible results. The intersection of B-Spline primitives adds further complexity to this process. Consequently, our experiments primarily utilized elementary primitives.

However, B-Spline primitives were still employed in training the network for Voronoi boundary prediction, given that Voronoi diagram construction is independent of primitive type. We also visualize our segmentation results for B-Spline primitives in Fig. 11.

Meshing: Despite the ability of current methods, including ComplexGen [Guo et al. 2022], HPNet [Yan et al. 2021], SEDNet [Li et al. 2023b], and ours, to accurately determine primitives and their topologies, the meshing process remains erratic. A key challenge is that the primitives are topologically but not geometrically interconnected, affecting loop finding and trimming processes. ComplexGen attempts to address this through an optimization framework that adjusts geometric and topological relations simultaneously, but it

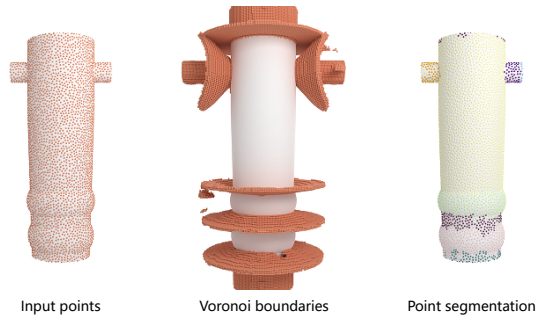


Fig. 11. Although we omitted the BSpline primitive in our mesh extraction process, NVD can still faithfully predict the Voronoi boundaries for **BSpline primitives** and subsequently assign the instance label to each point. We show the input points (*left*), the predicted Voronoi boundaries (*middle*) and the corresponding point segmentation (*right*).

still struggles with the instability of meshing. Developing a stable meshing process remains an unresolved issue in this field.

Shapes violate piece-wise G2 continuity: Our method utilizes training data generated from CAD models that adhere to piece-wise G2 continuity, with discontinuities occurring only at the junctions between primitives. When encountering input shapes that deviate from this continuity, such as partial inputs or surfaces with G2 continuous junctions, our method might struggle to predict Voronoi boundaries accurately. Nevertheless, it can effectively reconstruct the Voronoi boundaries and primitives in the intact regions of partial inputs, thanks to local feature learning (as shown in Fig. 10). In cases of surfaces with G2 continuous transitions, the reconstruction leans towards semantic analysis over geometric detail, necessitating a broader understanding of the shape’s features. To improve performance on such inputs, retraining NVD-Net with a dataset inclusive of shapes with G2 continuous transitions could be beneficial, as these also possess well-defined Voronoi Diagrams.

6 CONCLUSION AND FUTURE WORK

In this work, we introduce a novel pipeline for reconstructing Boundary Representation (B-Rep) CAD models, capable of processing diverse inputs such as point clouds, distance fields, and meshes. The enabler of our approach is the prediction of the Voronoi diagram representing the underlying shape. It serves as a foundation for extracting the geometric primitives and their topological relationships, which are then integrated to reconstruct the CAD models in B-Rep. The Voronoi diagram, being both *unique* and *fixed*, significantly reduces ambiguity in the reconstruction process. Its inherent structure conveniently encapsulates boundaries and connectivity of primitives, thereby simplifying the reconstruction workflow. Through extensive validation, our method has demonstrated superior performance over existing techniques in geometric accuracy, detection precision, topological consistency, and generalizability. The reconstructed B-Rep models are compatible with existing CAD software, enabling seamless integration into the design process; see Fig. 12.

The main limitation of our method is the accuracy of the Voronoi diagram. The voxelization process inherent to our method introduces quantization errors that adversely impact the Voronoi diagram’s precision. This effect is particularly pronounced in the case of tiny structures, which may be lost during voxelization and subsequently affect the extraction of primitives. Future research will investigate alternative representations of the Voronoi diagram, such as implicit functions, to circumvent the quantization errors associated with voxelization.

ACKNOWLEDGMENTS

We thank all the anonymous reviewers for their insightful comments. Thanks also go to Haoxiang Guo, Dani Lischinski, and Ningna Wang for helpful discussions. This work was supported in part by NSFC (62161146005, U21B2023, U2001206), Guangdong Basic and Applied Basic Research Foundation (2023B1515120026), DEGP Innovation Team (2022KCXTD025), Shenzhen Science and Technology Program (KQTD20210811090044003, RCJC20200714114435012, JCYJ20210324120213036), NSERC (611370), ISF (3441/21), SFU Graduate Dean’s Entrance Scholarship, Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), and Scientific Development Funds from Shenzhen University.

REFERENCES

- Pierre Alliez, David Cohen-Steiner, Yiyi Tong, and Mathieu Desbrun. 2007. Voronoi-based variational reconstruction of unoriented point sets. In *Proc. Eurographics Symp. on Geometry Processing*, Vol. 257. 39–48.
- Marco Attene and Giuseppe Patanè. 2010. Hierarchical Structure Recovery of Point-Sampled Surfaces. *Computer Graphics Forum* 29, 6 (2010), 1905–1920.
- Jean-Philippe Bauchet and Florent Lafarge. 2020. Kinetic shape reconstruction. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 39, 5 (2020), 156:1–156:14.
- Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. 2003. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum* 22, 3 (2003), 223–232.
- Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. 2022. Neural Dual Contouring. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 41, 4 (2022), 104:1–104:13.
- Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2020. BSP-Net: Generating Compact Meshes via Binary Space Partitioning. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*. 42–51.
- David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational shape approximation. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 23, 3 (2004), 905–914.
- Jianmin Zheng Daxuan Ren, Jianfei Cai, Haiyong Jiang Jiatong Li, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, and Shuai Yi. 2021. CSG-Stump: A Learning Friendly CSG-Like Representation for Interpretable Shape Parsing. In *ICCV*. 12458–12467.
- David Eberly. 2000. Geometric Tools. <https://www.geometrictools.com/>.
- Pierre-Alain Fayolle and Markus Friedrich. 2023. A Survey of Methods for Converting Unstructured Data to CSG Models. arXiv:2305.01220 [cs.GR]
- Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.
- Rao Fu, Cheng Wen, Qian Li, Xiao Xiao, and Pierre Alliez. 2023. BPNet: B⁺ezier Primitive Segmentation on 3D Point Clouds. In *Proc. Int. Joint Conf. on Artificial Intelligence*. 754–762.
- Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. 2022. ComplexGen: CAD reconstruction by B-rep chain complex generation. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 41, 4 (2022), 129:1–129:18.
- Jingwei Huang, Yanfeng Zhang, and Mingwei Sun. 2021. PrimitiveNet: Primitive Instance Segmentation with Local Primitive Embedding under Adversarial Metric. In *Proc. Int. Conf. on Computer Vision*. 15323–15333.
- Pradeep Kumar Jayaraman, Joseph George Lambourne, Nishkrit Desai, Karl D. D. Willis, Aditya Sanghi, and Nigel J. W. Morris. 2023. SolidGen: An Autoregressive Model for Direct B-rep Synthesis. *Trans. Mach. Learn. Res.* (2023).
- Adrien Kaiser, José Alonso Ybáñez Zepeda, and Tamy Boubekeur. 2019. A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data. *Computer Graphics Forum* 38, 1 (2019), 167–196.

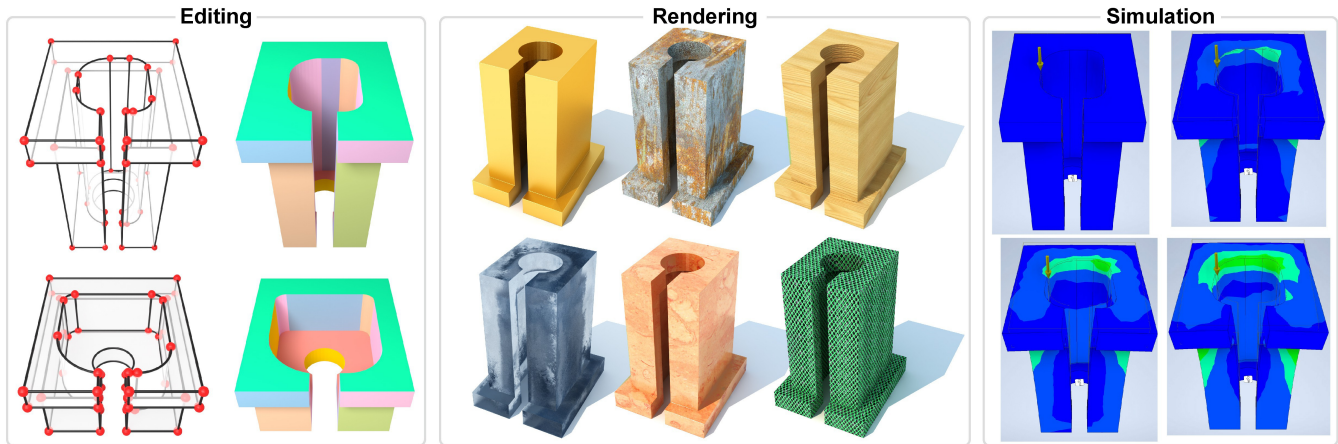


Fig. 12. Three distinct applications utilizing the reconstructed B-Rep model. The reconstructed B-Rep model's compactness, cleanliness, and preservation of sharp features make them exceptionally well-suited for various subsequent uses, including editing, rendering, physical simulation, etc.

- Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. 2020. UCSG-NET-unsupervised discovering of constructive solid geometry tree. In *Proc. Conf. on Neural Information Processing Systems*, Vol. 33. 8776–8786.
- Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A Big CAD Model Dataset for Geometric Deep Learning. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*. 9601–9611.
- Florent Lafarge and Clément Mallet. 2012. Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation. *Proc. Int. Conf. on Computer Vision* 99 (2012), 69–85.
- Joseph G. Lambourne, Karl D.D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. 2021. BRepNet: A topological message passing system for solid models. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*. 12773–12782.
- Joseph G. Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Longfei Zhang, Aditya Sanghi, and Kamal Rahimi Malekshah. 2022. Reconstructing Editable Prismatic CAD from Rounded Voxel Models. In *Proc. SIGGRAPH Asia*. 53:1–53:9.
- Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J. Guibas. 2019a. Supervised Fitting of Geometric Primitives to 3D Point Clouds. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*. 2652–2660.
- Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J. Guibas. 2019b. Supervised Fitting of Geometric Primitives to 3D Point Clouds. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*. 2652–2660.
- Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-Ming Yan. 2023a. SECAD-Net: Self-Supervised CAD Reconstruction by Learning Sketch-Extrude Operations. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*. 16816–16826.
- Yuanqi Li, Shun Liu, Xinran Yang, Jianwei Guo, Jie Guo, and Yanwen Guo. 2023b. Surface and Edge Detection for Primitive Fitting of Point Clouds. In *Proc. SIGGRAPH*. 44:1–44:10.
- Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J Mitra. 2011. Globfit: Consistently fitting primitives by discovering global relations. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 30, 4 (2011), 52:1–52:12.
- Cheng Lin, Lingjie Liu, Changjian Li, Leif Kobbelt, Bin Wang, Shiqing Xin, and Wenping Wang. 2022. SEG-MAT: 3D Shape Segmentation Using Medial Axis Transform. *IEEE Trans. Visualization & Computer Graphics* 28, 6 (2022), 2430–2444.
- Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. 2020. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Analysis & Machine Intelligence* 42, 2 (2020), 318–327.
- Yujia Liu, Anton Obukhov, Jan Dirk Wegner, and Konrad Schindler. 2023. Point2CAD: Reverse Engineering CAD Models from 3D Point Clouds. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition Workshops*.
- Yong-Jin Liu, Chun-Xu Xu, Dian Fan, and Ying He. 2015. Efficient Construction and Simplification of Delaunay Meshes. *ACM Trans. on Graphics* 34, 6 (2015), 174:1–174:13.
- Nissim Maruani, Roman Klokov, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. 2023. VoroMesh: Learning Watertight Surface Meshes with Voronoi Diagrams. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*. 14565–14574.
- Albert Matveev, Ruslan Rakhimov, Alexey Artemov, Gleb Bobrovskikh, Vage Egjazarian, Emil Bogomolov, Daniele Panozzo, Denis Zorin, and Evgeny Burnaev. 2022. Def: Deep Estimation of Sharp Geometric Features in 3D Shapes. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 41, 4 (2022), 108:1–108:22.
- Liangliang Nan and Peter Wonka. 2017. PolyFit: Polygonal Surface Reconstruction from Point Clouds. In *Proc. Int. Conf. on Computer Vision*. 2372–2380.
- Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. 2020. PolyGen: An Autoregressive Generative Model of 3D Meshes. In *Proc. Int. Conf. on Machine Learning*, Vol. 119. 7220–7229.
- Sven Oesau, Florent Lafarge, and Pierre Alliez. 2016. Planar Shape Detection and Regularization in Tandem. *Computer Graphics Forum* 35, 1 (2016), 203–215.
- Ruwen Schnabel, Roland Wahl, and Reinhard Klein. 2007. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum* 26, 2 (2007), 214–226.
- Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomir Měch. 2020. ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds. In *Proc. Euro. Conf. on Computer Vision*, Vol. 12352. 261–276.
- Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. 2023. MeshGPT: Generating Triangle Meshes with Decoder-Only Transformers. arXiv:2311.15475 [cs.CV]
- Martin Skrodzki, Eric Zimmermann, and Konrad Polthier. 2020. Variational shape approximation of point set surfaces. *Computer Aided Geometric Design* 80 (2020), 101875.
- Ningna Wang, Bin Wang, Wenping Wang, and Xiaohu Guo. 2022. Computing medial axis transform with feature preservation via restricted power diagram. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 41, 6 (2022), 188:1–188:18.
- Francis Williams, Jerome Parent-Levesque, Derek Nowrouzezahrai, Daniele Panozzo, Kwang Moo Yi, and Andrea Tagliasacchi. 2020. Voronoinet: General functional approximators with local support. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition Workshops*. 264–265.
- Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl D. D. Willis, and Daniel Ritchie. 2021. Inferring CAD Modeling Sequences Using Zone Graphs. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*. 6062–6070.
- Dong-Ming Yan, Wenping Wang, Yang Liu, and Zhouwang Yang. 2012. Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design* 44, 11 (2012), 1072–1082.
- Siming Yan, Zhenpei Yang, Chongyang Ma, Haibin Huang, Etienne Vouga, and Qixing Huang. 2021. HPNet: Deep Primitive Segmentation Using Hybrid Representations. In *Proc. Int. Conf. on Computer Vision*. 2733–2742.
- Fenggen Yu, Qimin Chen, Maham Tanveer, Ali Mahdavi Amiri, and Hao Zhang. 2023. D2CSG: Unsupervised Learning of Compact CSG Trees with Dual Complements and Dropouts. In *Proc. Conf. on Neural Information Processing Systems*.
- Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. 2022. CAPRI-Net: Learning Compact CAD Shapes with Adaptive Primitive Assembly. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*. 11758–11768.
- Wenni Zheng, Pengbo Bo, Yang Liu, and Wenping Wang. 2012. Fast B-spline Curve Fitting by L-BFGS. *Computer Aided Geometric Design* 29, 7 (2012), 448–462.
- Xiangyu Zhu, Dong Du, Weikai Chen, Zhiyou Zhao, Yinyu Nie, and Xiaoguang Han. 2023. NerVE: Neural Volumetric Edges for Parametric Curve Extraction from Point Cloud. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*. 13601–13610.

Split-and-Fit: Learning B-Reps via Structure-Aware Voronoi Partitioning (Supplementary Materials)

YILIN LIU, Shenzhen University, China and Simon Fraser University, Canada

JIALE CHEN, Shenzhen University, China

SHANSHAN PAN, Shenzhen University, China

DANIEL COHEN-OR, Tel Aviv University, Israel

HAO ZHANG, Simon Fraser University, Canada and Amazon, Canada

HUI HUANG[†], Shenzhen University, China

CCS Concepts: • **Computing methodologies** → **Shape inference**;

Additional Key Words and Phrases: Neural Voronoi diagram, CAD modeling, boundary representation

ACM Reference Format:

Yilin Liu, Jiale Chen, Shanshan Pan, Daniel Cohen-Or, Hao Zhang, and Hui Huang. 2024. Split-and-Fit: Learning B-Reps via Structure-Aware Voronoi Partitioning (Supplementary Materials). *ACM Trans. Graph.* 43, 4, Article 108 (July 2024), 27 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

This supplementary material presents the training details of NVD-Net and a pseudocode describing the subsequent shape extraction algorithm. We also include a qualitative comparison of four methods on 30 representative shapes and 30 randomly sample shapes in the test set (Sec. C).

A PRIMITIVE EXTRACTION

Our primitive extraction method begins by segmenting the input shape into Voronoi cells, aiming to simplify the complex task of primitive fitting by localizing the process within each cell, see Sec. A.1. This segmentation facilitates a more manageable and efficient primitive fitting process as each cell typically contains fewer primitives than the entire shape, significantly improving performance over traditional global fitting methods like RANSAC [Fischler and Bolles 1981; Schnabel et al. 2007]. Similar to SEDNet [Li et al. 2023b] and Point2CAD [Liu et al. 2023], we additionally recover curves and vertices through primitive intersections. We show the pseudocode of our primitive extraction in Algorithm 1. The process culminates in the generation of a comprehensive B-Rep model, detailed through

[†]Corresponding author: Hui Huang (hhzhiyan@gmail.com)

Authors' addresses: Yilin Liu, whatsevenlyl@gmail.com, Shenzhen University, China and Simon Fraser University, Canada; Jiale Chen, chenjiale0303@gmail.com, Shenzhen University, China; Shanshan Pan, psshappystar@gmail.com, Shenzhen University, China; Daniel Cohen-Or, cohenor@gmail.com, Tel Aviv University, Israel; Hao Zhang, hao.r.zhang@gmail.com, Simon Fraser University, Canada and Amazon, Canada; Hui Huang, hhzhiyan@gmail.com, College of Computer Science & Software Engineering, Shenzhen University, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2024 Association for Computing Machinery.

0730-0301/2024/7-ART108 \$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

Table 6. Summary of important notations.

Notation	Description
\mathbb{B}	Boolean domain $\{0, 1\}$
$N_{v/e/f}$	element numbers for vertices/curves/surfaces
$M = (V, E, F, \partial, \mathcal{P})$	a B-Rep chain complex with vertices V , curves E , surfaces F , topological embedding $\partial = \{FF, FE, EE, EV, FV\}$, and geometric embedding \mathcal{P}
$FF \in \mathbb{B}^{N_f \times N_f}$	the adjacency of surfaces and surfaces
$FE \in \mathbb{B}^{N_f \times N_e}$	the adjacency of surfaces and curves
$EE \in \mathbb{B}^{N_e \times N_e}$	the adjacency of curves and curves
$EV \in \mathbb{B}^{N_e \times N_v}$	the adjacency of curves and vertices
$FV \in \mathbb{B}^{N_f \times N_v}$	the adjacency of surfaces and vertices
P_i	the internal points in each Voronoi cell
ϵ_1	0.001, threshold of Fitting_Primitives
ϵ_2	0.02, threshold of Build_Surfaces_Adjacency
ϵ_3	0.05, threshold of Build_Curves_Adjacency

Algorithm 1: Overview of our primitive extraction process

```

input : Voronoi cells  $N_v$ 
output: B-Rep model  $M(V, E, F, \partial, \mathcal{P})$ 
1  $V, E, F \leftarrow \emptyset$ ;
2  $S_{\text{Type}} \leftarrow \{\text{plane, sphere, cylinder, cone, torus}\}$ ;
3  $C_{\text{Type}} \leftarrow \{\text{line, circle, ellipse}\}$ ;
4  $F, E \leftarrow \text{Fitting\_Primitives}(N_v)$ ;
5  $\mathbf{FF} \leftarrow \text{Build\_Surfaces\_Adjacency}(N_v, F)$ ;
6  $E, \mathbf{FE} \leftarrow \text{Curve\_Extraction}(F, E, \mathbf{FF})$ ;
7  $\mathbf{EE} \leftarrow \text{Build\_Curves\_Adjacency}(F, E, \mathbf{FE})$ ;
8  $V, \mathbf{EV}, \mathbf{FV} \leftarrow \text{Vertex\_Extraction}(E, \mathbf{EE})$ ;

```

vertices (V), curves (E), and surfaces (F), bound together by their geometric and topological relationships, see Sec. A.2.

A.1 Hole Filling and Cell Construction

We identify Voronoi cells from the predicted Voronoi boundaries using region growing. Due to holes commonly appearing in these boundaries, we perform hole-filling to refine them before reconstructing the cells. This involves traversing each voxel in the space

Algorithm 2: Fitting_Primitives()

input : Voronoi cells N_v
output : Surfaces F and Curves E

```

1  $E, F \leftarrow \emptyset$ ;
2 Error  $\epsilon \leftarrow \text{inf}$ ;
3 for each  $N_i$  in  $N_v$  do
4   Surface  $f, \epsilon \leftarrow \text{Plane\_Fitting}(N_i)$ ;
5   if  $\epsilon < \epsilon_1$  then
6     // find the best curve in CType using points in  $N_i$ 
7     Curve  $e, \epsilon \leftarrow \text{Curve\_Fitting}(N_i)$ ;
8     if  $\epsilon < \epsilon_1$  then
9        $E \leftarrow E \cup e$ ;
10      continue;
11    else
12       $F \leftarrow F \cup f$ ;
13  else
14    // find the best surface in SType using points in  $N_i$ 
15    Surface  $f, \epsilon \leftarrow \text{Surface\_Fitting}(N_i)$ ;
16    if  $\epsilon < \epsilon_1$  then
17       $F \leftarrow F \cup f$ ;
18    else
19      // Use RANSAC to fit multiple primitives
20      Surface set  $F_s \leftarrow \text{RANSAC}(N_v)$ ;
21       $F \leftarrow F \cup F_s$ ;

```

Algorithm 3: Build_Surfaces_Adjacency()

input : Voronoi cells N_v and Surfaces F
output : Surfaces adjacency \mathbf{FF}

```

1  $\mathbf{FF}[N_f][N_f] \leftarrow 0$ ;
2 for each  $N_i$  in  $N_v$  do
3   for each  $N_j$  in adjacent cells do
4     if  $\text{is\_SType}(N_i)$  and  $\text{is\_SType}(N_j)$  then
5        $\epsilon \leftarrow \text{Distance}(P_i, P_j)$ ;
6       if  $\epsilon < \epsilon_2$  then
7          $\mathbf{FF}[i][j] \leftarrow 1$ ;

```

along the gradient direction to gather a set of voxels and their classification results. If half or more of these are part of the Voronoi boundary, we also classify the rest as boundary cells. With these refined boundaries, we apply standard region growing to group the voxels into distinct Voronoi cells.

A.2 Primitive Fitting

We provide an overview of our primitive fitting process in Algorithm 1 and the chosen parameters in Table. 6. We first define two distinct sets of geometric primitives. For curves, our consideration extends to lines, circles, and ellipses, categorized under *CType*; for surfaces, we encompass planes, spheres, cylinders, cones, and tori,

Algorithm 4: Curve_Extraction()

input : Surfaces F , Curves E and Surfaces adjacency \mathbf{FF}
output : Curves E and Surface-Curve adjacency \mathbf{FE}

```

1 for each  $f_i$  in  $F$  do
2   for each  $f_j$  in adjacent surfaces do
3     Common points  $P \leftarrow P_i \cap P_j$ ;
4     Compute intersection curves  $E_s$ ;
5     // find the curve with minimal distance to  $P$ 
6     Curve  $e \leftarrow \text{Min\_Distance}(E_s, P)$ ;
7      $E \leftarrow E \cup e$ ;
8   update Surface-Curve adjacency  $\mathbf{FE}$ ;
9 remove duplication in  $E$  and update  $\mathbf{FE}$ ;

```

Algorithm 5: Build_Curves_Adjacency()

input : Surfaces F , Curves E and Surfaces adjacency \mathbf{FE}
output : Curves adjacency \mathbf{EE}

```

1  $\mathbf{EE}[N_e][N_e] \leftarrow 0$ ;
2 for each  $e_i$  in  $E$  do
3   for each  $e_j$  in  $E$  do
4     for each  $f_k$  in  $F$  do
5       if  $\mathbf{FE}[k][i] = 1$  and  $\mathbf{FE}[k][j] = 1$  then
6          $\epsilon \leftarrow \text{Distance}(P_i, P_j)$ ;
7         if  $\epsilon < \epsilon_3$  then
8            $\mathbf{EE}[i][j] \leftarrow 1$ ;

```

Algorithm 6: Vertex_Extraction()

input : Curves E and Curves adjacency \mathbf{EE}
output : Vertices V , Curve-Vertex adjacency \mathbf{EV} and Surface-Vertex adjacency \mathbf{FV}

```

1 for each  $e_i$  in  $E$  do
2   for each  $e_j$  in adjacent curves do
3     Compute intersection vertices  $V_s$ ;
4     // find the vertex with minimal distance to  $e_i$  and  $e_j$ 
5     Vertex  $v \leftarrow \text{Min\_Distance}(V_s, e_i, e_j)$ ;
6      $V \leftarrow V \cup v$ ;
7     update Curve-Vertex adjacency  $\mathbf{EV}$ ;
8     update Surface-Vertex adjacency  $\mathbf{FV}$ ;
9 remove duplication in  $V$  and update  $\mathbf{EV}, \mathbf{FV}$ ;

```

collectively referred to as *SType*. We use least-square fitting to obtain all these primitives¹. Details of the primitive fitting process are described as follows (Algorithm 2):

- **Sequential Fitting:** We perform a sequential fitting process for each Voronoi cell. Initially, we attempt to fit the simplest primitive, plane, and evaluate the fitting error using a predefined threshold ϵ_1 . Since a cell of a curve may also be recognized as a plane, we proceed to curve fitting within the

¹<https://www.geometrictools.com/Samples/Mathematics.html>

cell if the plane is successfully fitted, trying each curve type in $CType$ and selecting the one with the minimal fitting error. If no satisfactory curve fitting is achieved (minimal fitting error $> \epsilon_1$), we explore surface fitting by iterating through $SType$. A similar error-based selection criterion is applied to determine the best-fitting surface.

- **Fallback Mechanism:** For cells where a single primitive fitting is unfeasible, we employ a traditional RANSAC algorithm to detect combinations of primitives, ensuring robustness in handling complex geometries.

Once we have done the primitive fitting, subsequent steps delve into defining topological relationships and extracting finer geometric details. Details of each step are described as follows:

- **Surfaces Adjacency (Algorithm 3):** We consider surfaces to be adjacent if they belong to neighbouring Voronoi cells and the distance between their points is less than a predefined threshold ϵ_2 . This adjacency relationship is stored in the surfaces adjacency matrix (**FF**). This dual condition ensures a robust method of determining adjacency, where spatial proximity and Voronoi adjacency work in tandem.
- **Curve Extraction (Algorithm 4):** The intersection of surfaces is computed to refine the curve elements (E)². Since two surfaces may intersect along multiple curves, we identify the correct boundary curve by finding common points shared by the surfaces. This process ensures that the extracted curves precisely represent the shape's edges in the B-Rep model.
- **Curves Adjacency (Algorithm 5):** We define curves as adjacent if they are connected to the same surface and the distance between their internal points is less than a predefined threshold ϵ_3 . This adjacency relationship is stored in the curves adjacency matrix (**EE**).
- **Vertex Extraction (Algorithm 6):** Similar to the curve extraction, we extract vertices (V) through the intersection of curves, completing the boundary representation (B-Rep) model construction.

B NETWORK AND TRAINING DETAILS

Network: The backbone of the approach is a simple UNet architecture, which comprises four down-sampling and four up-sampling layers. The architecture is detailed as follows:

Conv(1, 4, 16) – *BatchNorm* – *ReLU*–
DownBlock(16, 32) – *DownBlock*(32, 64)–
DownBlock(64, 128) – *DownBlock*(128, 256)–
UpBlock(256, 128) – *Block*(256, 128)–
UpBlock(128, 64) – *Block*(128, 64)–
UpBlock(64, 32) – *Block*(64, 32)–
UpBlock(32, 16) – *Block*(32, 16) – *Block*(16, 1) – *Sigmoid*,

where *Conv*(*kernelsize*, *in*, *out*) denotes a 3D convolutional layer. *Block* consists of a *Conv*(3, *in*, *out*), a *BatchNorm*, a *ReLU* layer, a *Conv*(3, *out*, *out*), a *BatchNorm* and a *ReLU* layer. *DownBlock* consists of a *Block*(*in*, *out*) and a *MaxPool* layer. *UpBlock* consists

of a *Upsample*, a *Conv*(3, *in*, *out*) and a *ReLU* layer. All the *UpBlock* has a skip concatenation from the corresponding *DownBlock* layer.

Data: For training data, we use the same split as the previous work [Guo et al. 2022; Li et al. 2023b; Yan et al. 2021]. We use the shape number from 0-800000 in ABC dataset for training, 800000-900000 for validation and 900000-1000000 for testing. For each shape, we use Poisson disk sampling to sample 10000 points and feed it into NDC [Chen et al. 2022] to generate the UDF field and corresponding gradient direction as the network input. As for the ground truth label of the Voronoi diagram, we first voxelize the unit square bounding box of the shape into a 256^3 grid. Based on the annotation provided by the ABC dataset, we compute and compare the nearest primitive for each voxel. By definition, if two adjacent voxels have different nearest primitives, there is a Voronoi boundary between them. We label both voxels as the voxelized Voronoi boundaries and train the network to predict this label. To learn local features, we split the voxel grid into 32^3 local patches and train the network to predict the Voronoi boundaries for each local patch. We also filter out patches far from the actual shape (i.e., the distance to the shape is larger than 0.3).

Training: We use Adam optimizer with a learning rate 0.0001 and a batch size 16. The weight of focal loss is set to 0.75 since most of the voxels have negative labels. We also augment the data by randomly flipping the input UDF field and the corresponding ground truth label.

C ADDITIONAL RESULTS

We show the rest of **representative** shapes in Figs. 13, 14, 15, 16, 17, 18 and **randomly** selected shapes in Figs. 19, 20, 21, 22, 23.

²https://dev.opencascade.org/doc/overview/html/occt_user_guides__modeling_algos.html

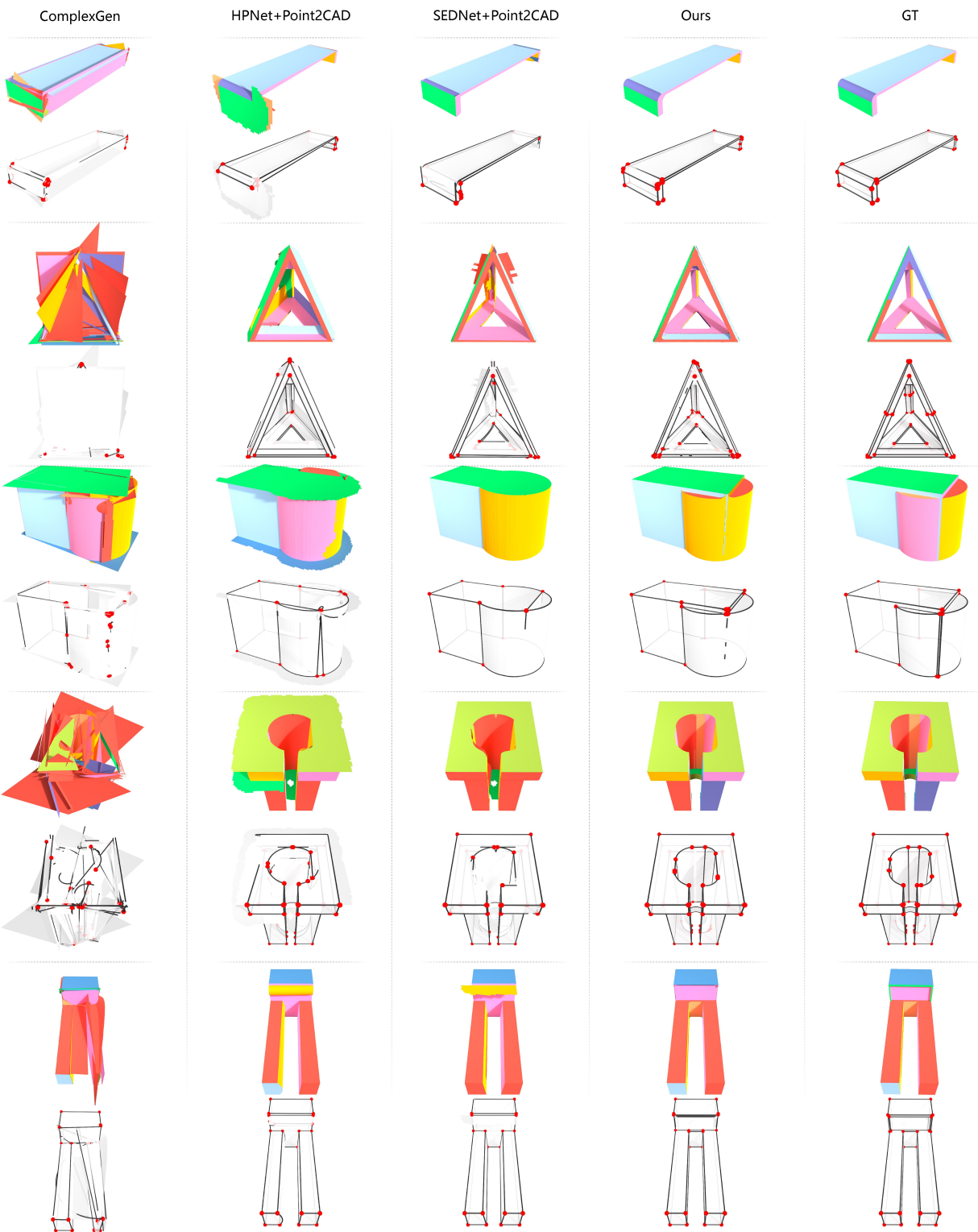


Fig. 13. Qualitative comparisons of **representative** cases (1/6).

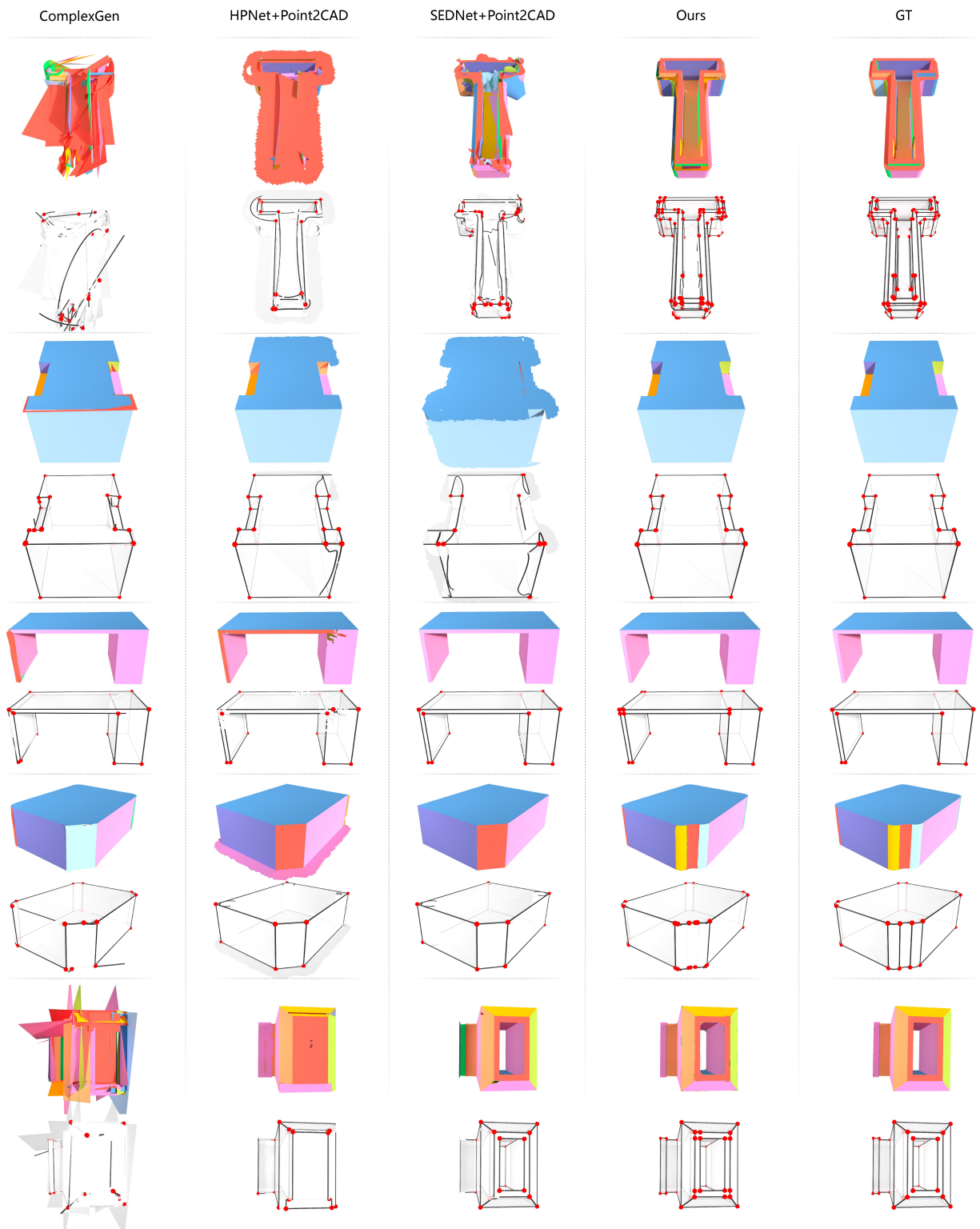


Fig. 14. Qualitative comparisons of **representative** cases (2/6).

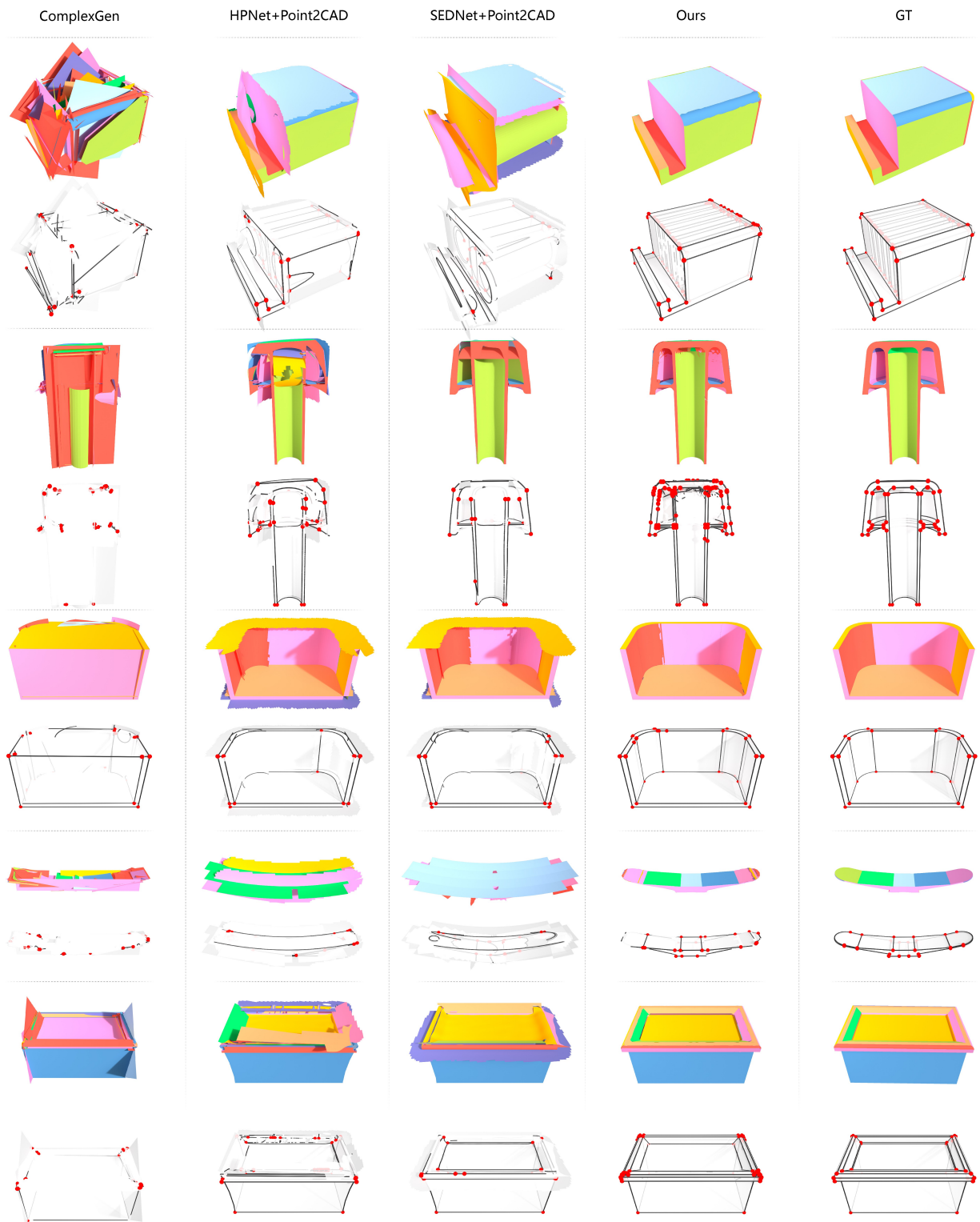


Fig. 15. Qualitative comparisons of **representative** cases (3/6).



Fig. 16. Qualitative comparisons of **representative** cases (4/6).

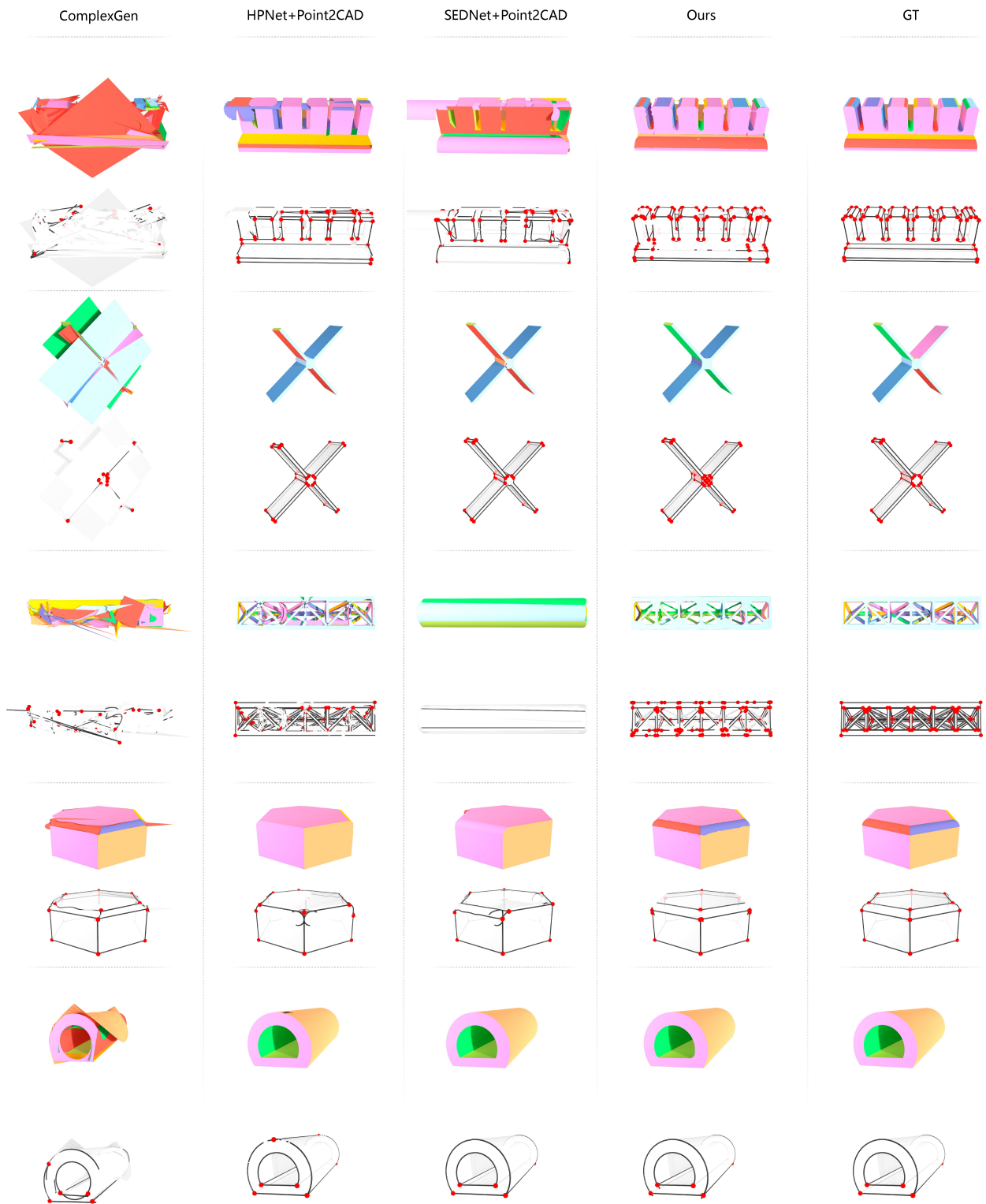


Fig. 17. Qualitative comparisons of **representative** cases (5/6).

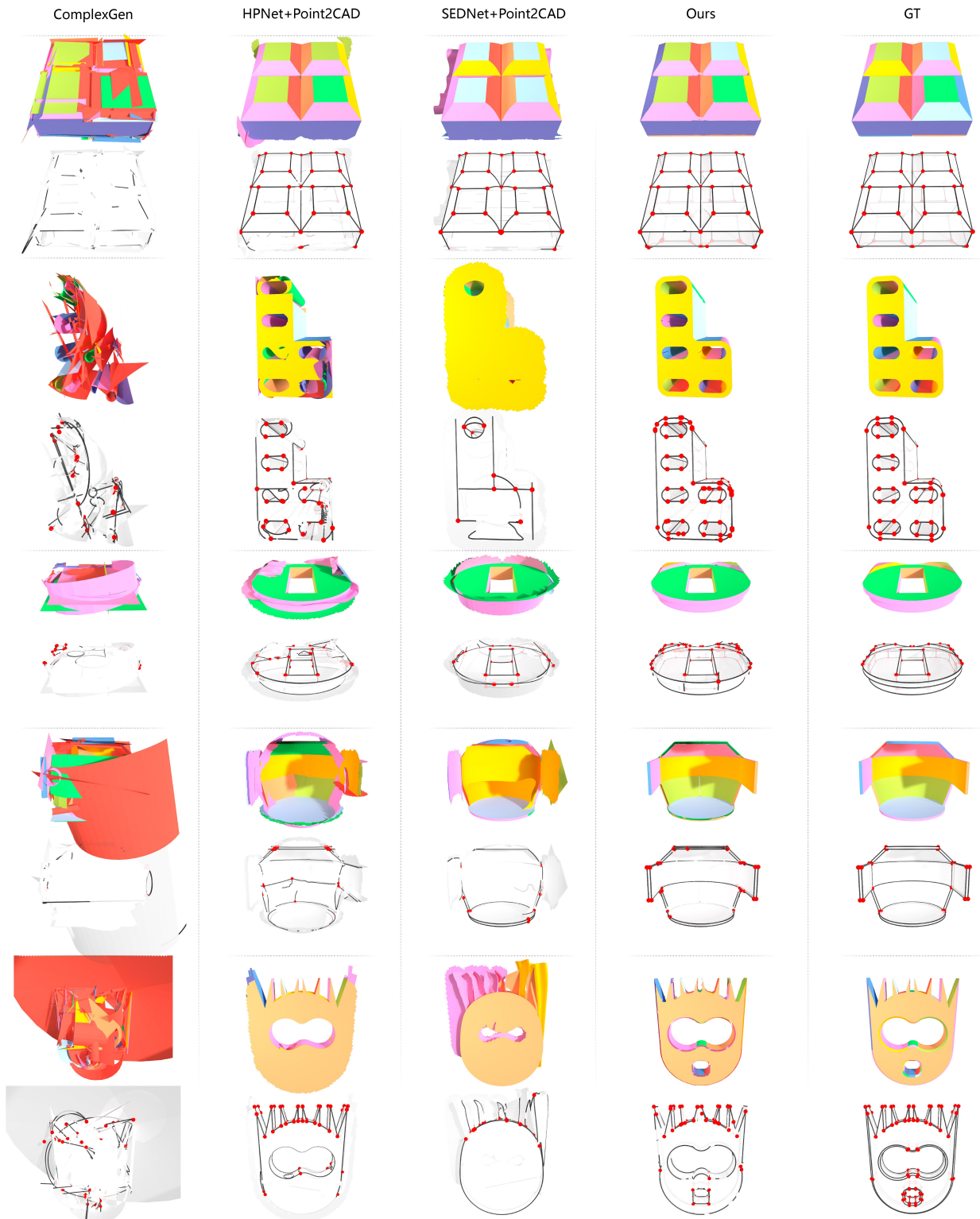


Fig. 18. Qualitative comparisons of **representative** cases (6/6).

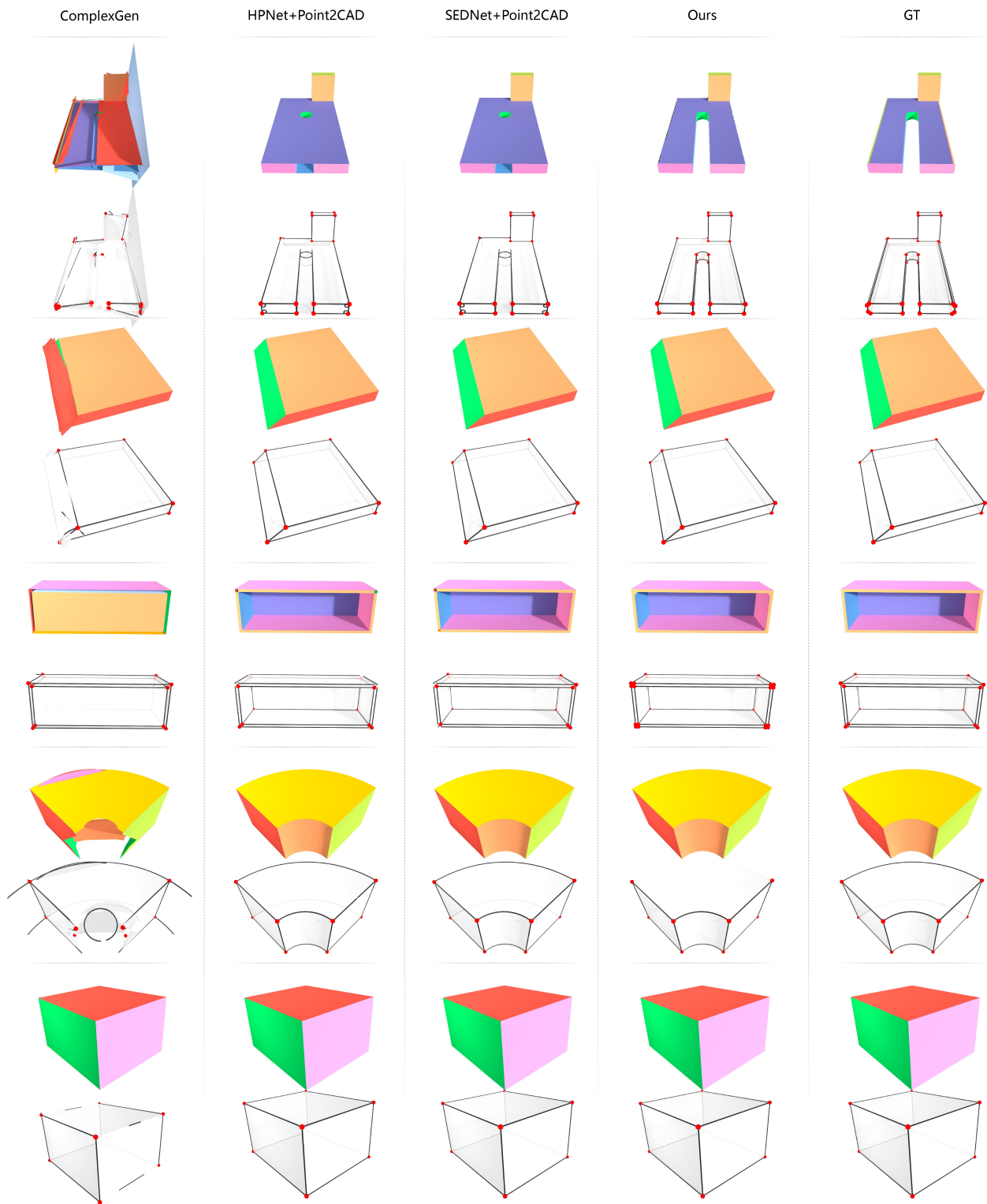


Fig. 19. Qualitative comparisons of **randomly** selected cases (1/5).

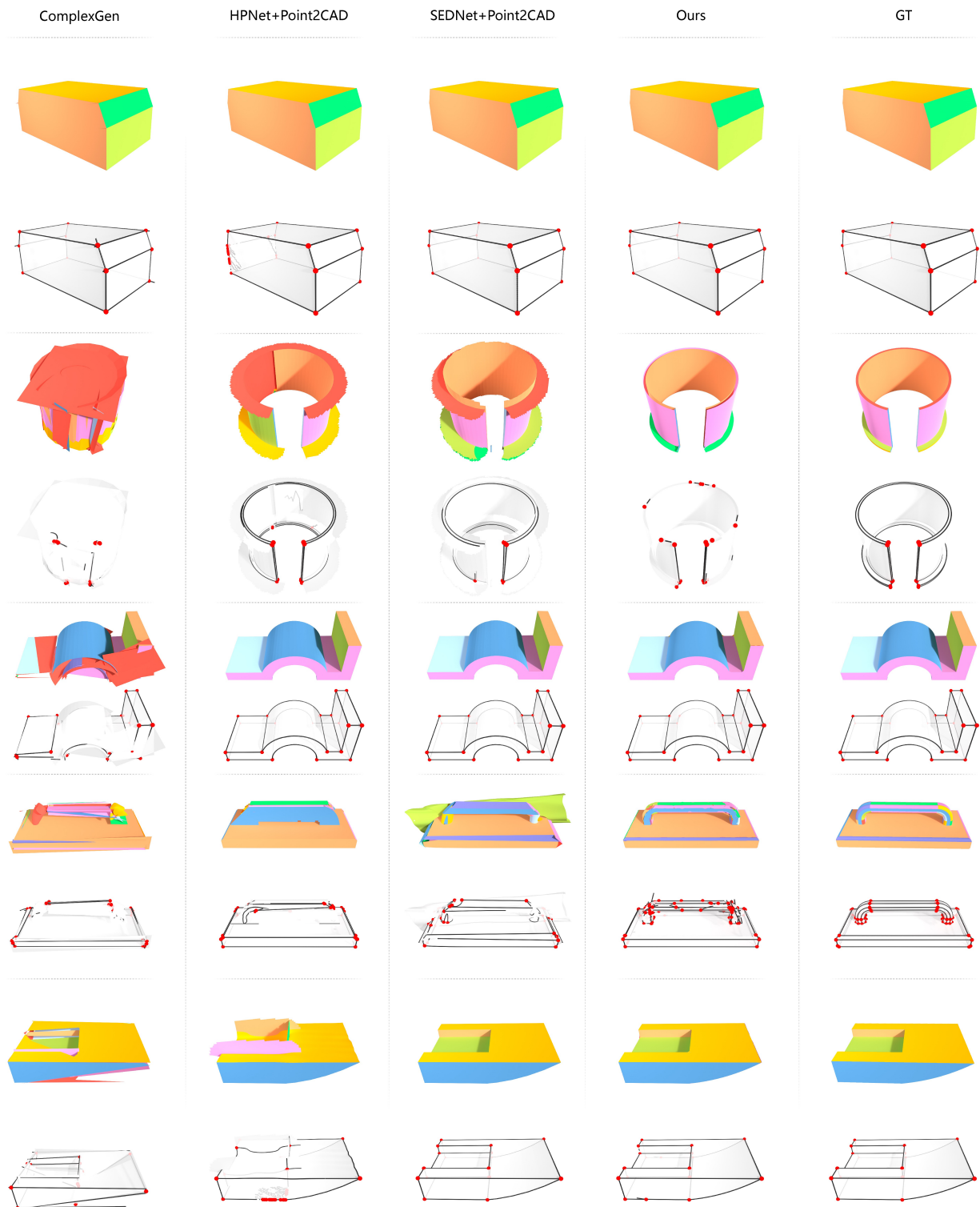


Fig. 20. Qualitative comparisons of **randomly** selected cases (2/5).

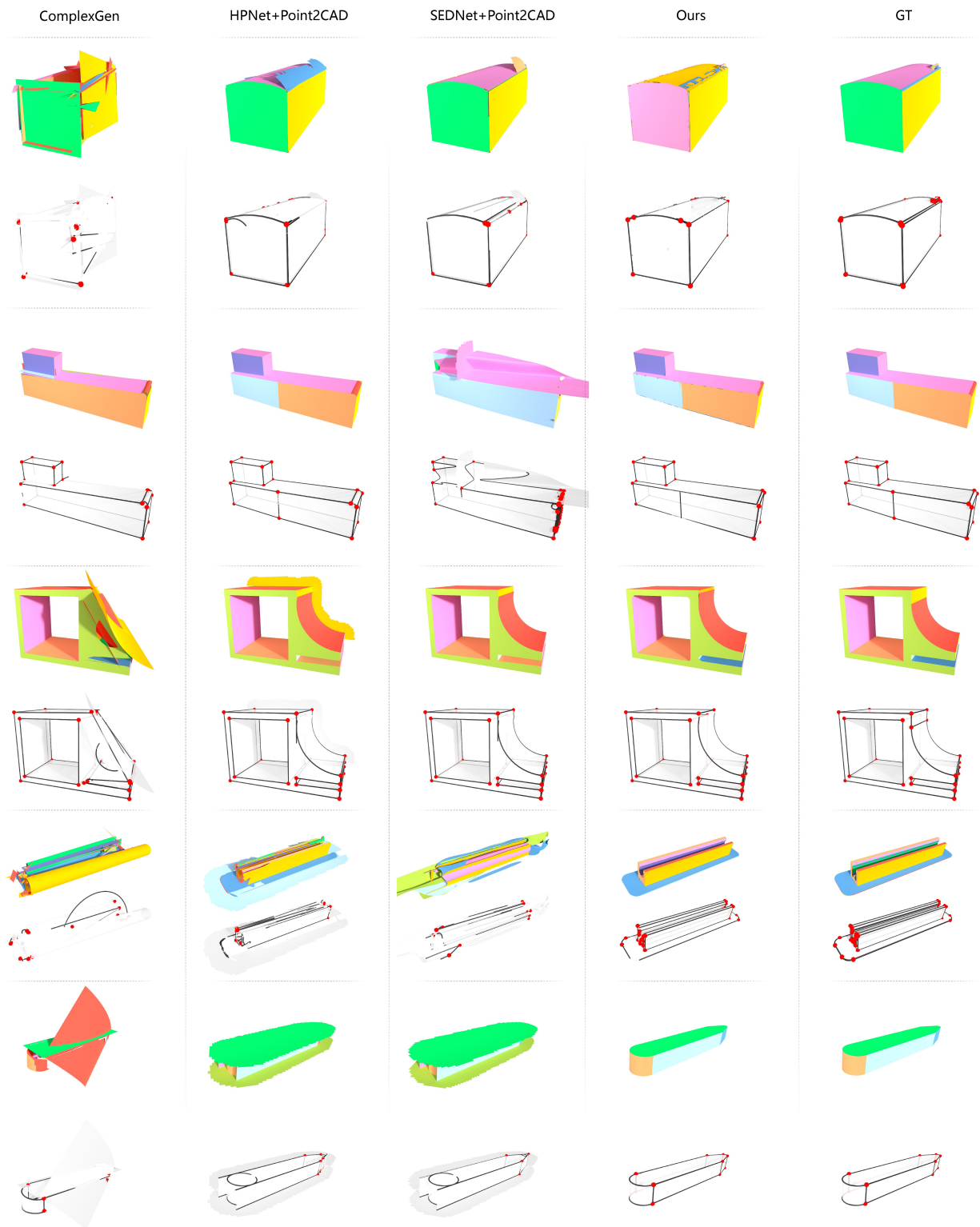


Fig. 21. Qualitative comparisons of **randomly** selected cases (3/5).

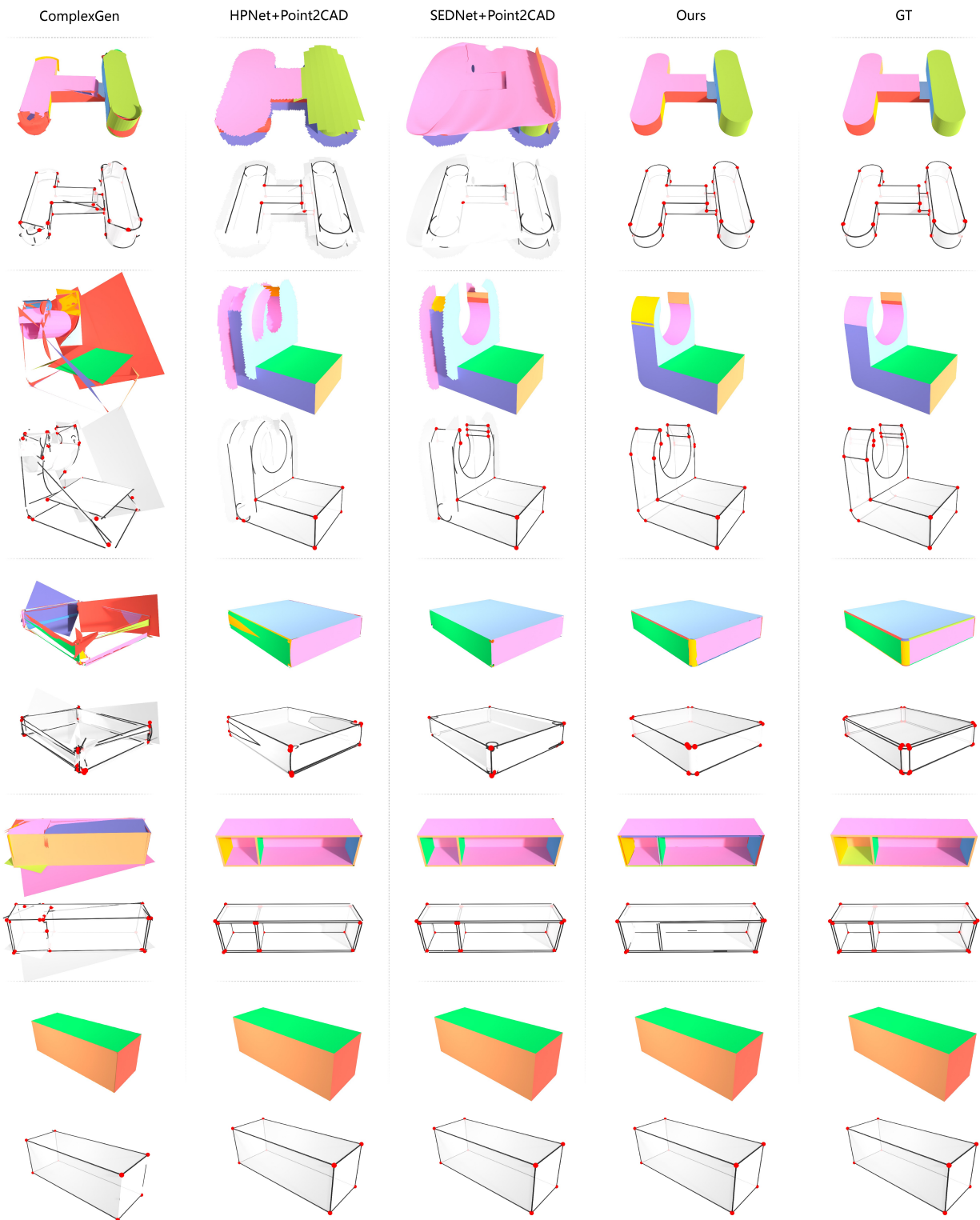


Fig. 22. Qualitative comparisons of **randomly** selected cases (4/5).

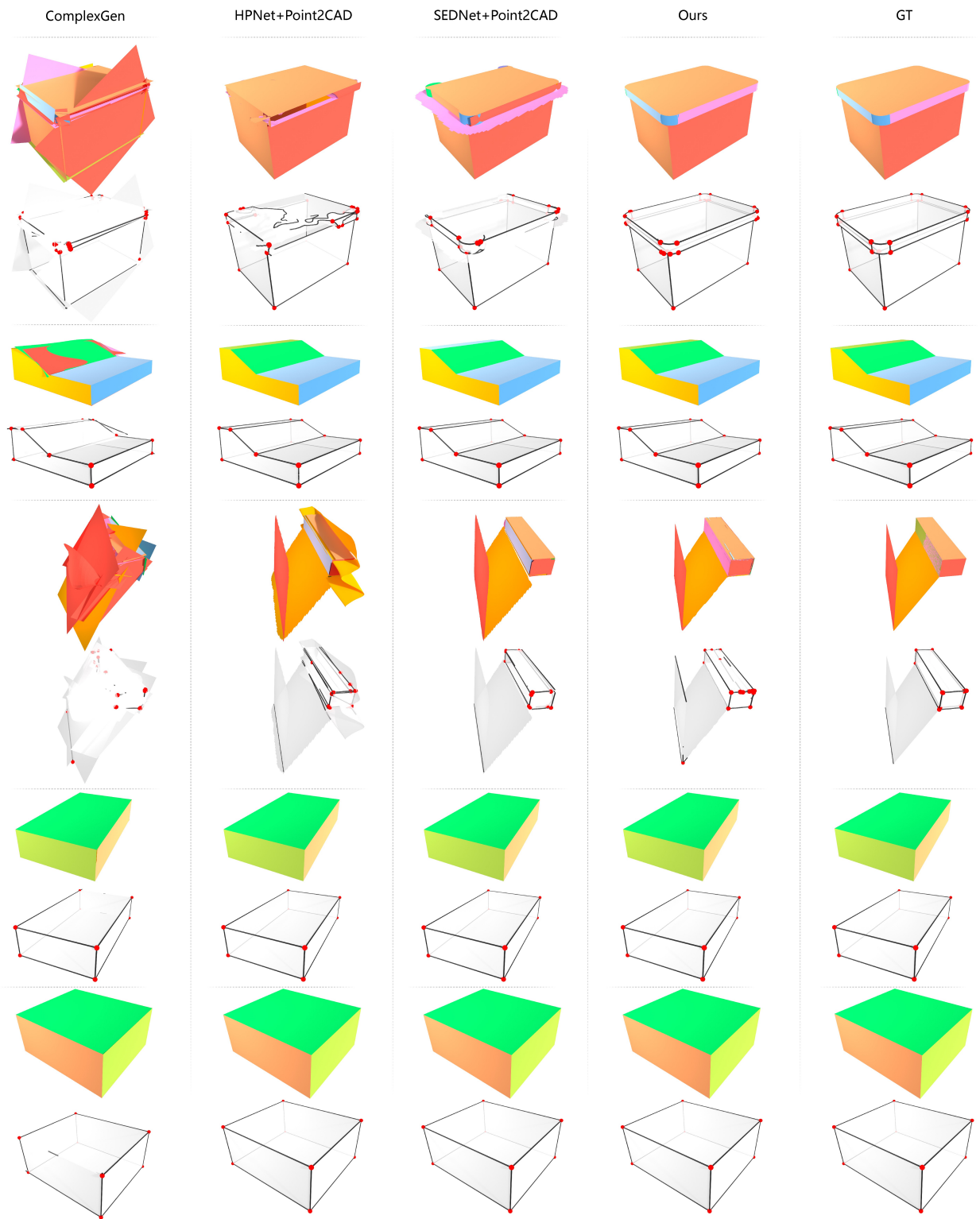


Fig. 23. Qualitative comparisons of **randomly** selected cases (5/5).