Structure–Aware Surface Reconstruction via Primitive Assembly

Jingen Jiang[1,2,3,4*] Mingyang Zhao[1,5*] Shiqing Xin[4] Yanchao Yang[6]
Hanxiao Wang[1,3] Xiaohong Jia[2,3] Dong-Ming Yan[1,3†]
[1]*MAIS & NLPR, CASIA* [2]*KLMM, AMSS, CAS*
[3]*UCAS* [4]*Shandong University* [5]*BAAI* [6]*The University of Hong Kong*

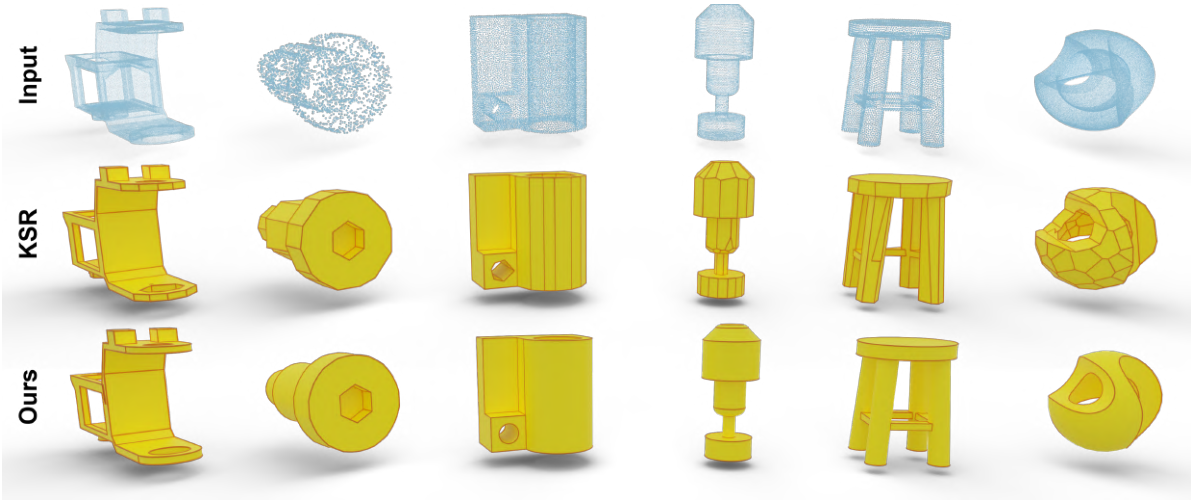


Figure 1: Surface reconstruction for measurement point clouds. In contrast to piecewise approximations (KSR), our method not only delivers manifold and watertight surfaces but also retains the primitive structures (planes, cylinders, spheres, cones, and tori) faithfully. Also, our method enables the recovery of the sharp geometry of objects (*e.g.*, feature lines) as a byproduct.

Abstract

We propose a novel and efficient method for reconstructing manifold surfaces from point clouds. Unlike previous approaches that use dense implicit reconstructions or piecewise approximations and overlook inherent structures like quadrics in CAD models, our method faithfully preserves these quadric structures by assembling primitives. To achieve high-quality primitive extraction, we use a variational shape approximation, followed by a mesh arrangement for space partitioning and candidate primitive patches generation. We then introduce an effective pruning mechanism to classify candidate primitive patches as active or inactive, and further prune inactive patches to reduce the search space and speed up surface extraction significantly. Finally, the optimal active patches are computed by a binary linear programming and assembled as manifold and watertight surfaces. We perform extensive experiments on a wide range of CAD objects to validate its effectiveness.

1. Introduction

The problem of converting unorganized point clouds into surface meshes, also known as *surface reconstruction*, has various applications in 3D vision, computer graphics, and architecture geometry. In recent decades, although many powerful algorithms have been devoted to pursuing faithful and accurate reconstructions [19, 21, 34, 28, 27, 49], it remains a challenging task due to the inherent ill-posed nature [11, 36, 5].

The popular Poisson reconstruction methods [24, 25] implicitly generate *dense* smooth surfaces for point clouds, while others mainly focus on giving a compact approximation to objects by piecewise *planar shapes* or *polygonal* meshes [7, 13, 3, 35]. However, man-made objects in real-world scenarios are typically composed of non-planar quadratic surfaces or *quadric structures*, such as the *Computer-Aided Design* (CAD) models (see Fig. 1 as an example). Therefore, reconstructing manifold surface meshes from measurement data meanwhile retaining their intrinsic primitive structures in principle shall generate higher-quality approximation results than solely relying on poly-

*The authors contribute equally to this work.
†Corresponding author (yandongming@gmail.com).

gons.

We argue that a method that solves the aforesaid problem ought to satisfy the following objectives: 1) *Structure-awareness*. The reconstruction surfaces should faithfully keep the structural features of the input point cloud, for instance, the common primitives in CAD, meanwhile correctly recovering the detailed geometry, and giving a high-fidelity approximation (*e.g.*, the minimal Hausdorff distance) to the target object. 2) *Manifold guarantees*. The output surface meshes should be oriented 2-manifold and watertight [35]. 3) *Efficiency*. The developed algorithm should be scalable to large-scale point clouds with an acceptable computational complexity [3].

In this work, we develop a novel and efficient framework that takes the underlying geometric structures of point clouds into account when performing shape reconstruction. Our method ensures high-quality outputs and aims to achieve the above objectives. The core of our algorithm lies in a faithful reconstruction of intrinsic primitive structures presented in CAD models, as opposed to the direct surface approximation based on piecewise planar primitives. We consider *five* representative types of primitives commonly found in modern CAD systems, including four quadrics (*planes, cylinders, cones, spheres*) and *tori*. We adopt a variational shape approximation to optimize the extracted primitives followed by mesh arrangement techniques to attain primitive patches. A patch is a manifold that consists of a set of connected triangle meshes. As the size of patches is typically smaller than that of facets, our method enables a more stable and efficient reconstruction process than plane-based ones. Moreover, we propose a *patch pruning* mechanism to further speed up the consecutive surface extraction process. This mechanism distinguishes patches as either *active* or *inactive*, allowing us to perform fast local collision detection instead of a time-consuming global processing. Lastly, we select an optimal set of patches from the *active* candidate patches to assemble a reconstruction surface that respects the underlying structure of the object. To this end, we optimize a *Binary Linear Programming* (BLP) under hard constraints, as done in [35], to guarantee that the resulting surface is both manifold and watertight.

We perform extensive experiments to validate the effectiveness of our algorithm and compare it with state-of-the-art approaches using a variety of CAD models. Experimental results evidence that our method outperforms competitors in terms of both approximation precision and reconstruction efficiency, while also preserving the intrinsic primitive structures of the input point clouds with much better fidelity. To summarize, the main technical contributions include:

- An efficient structure-aware reconstruction method that produces accurate surface models while preserving the underlying intrinsic primitive structures of the input point clouds.
- A patch pruning technique that avoids the time-consuming global collision detection and accelerates the surface extraction by a large margin.
- A patch-induced binary linear programming for assembling manifold and watertight surfaces from candidate active patches, which significantly reduces the search space and helps achieve efficient and accurate surface reconstruction.

2. Related Work

In this section, we briefly review the works that are most relevant to ours. Please refer to [5, 23] for a more comprehensive discussion.

Geometric primitive extraction. Segmenting 3D point cloud models into a collection of geometric primitives such as planes and cylinders is helpful in most reconstruction approaches. Popular extraction frameworks include *region growing* [31, 39] and *RANSAC* [43, 52]. The former strategy takes the neighboring structure of models into consideration and propagates candidate primitives on their *k-nearest neighbor* graph until the fitting error exceeds a preset tolerance. Instead, RANSAC adopts a *hypothesis and verification* mechanism to recover primitives that embrace the most inliers. In order to make full use of the *a priori* structure of the input model, Li *et al*. [29] started with primitive extraction by RANSAC, followed by the refinement subject to spatial relationships such as parallelism and orthogonality. This idea was further extended to extract planar primitives simultaneously [32] or sequentially [37]. Recently, Yu *et al*. [49] proposed a new algorithm for extracting planar primitives, which has powerful potential for plane-assembly reconstruction. In our work, we leverage an efficient region growing strategy [39] for primitive extraction followed by a refinement via *variational shape optimization*.

Assembly-based reconstruction. The core idea of assembly-based methods is to integrate all extracted primitives into a hybrid mesh. *Connectivity analysis* and *slicing* are two principal assembly manners. Connectivity strategies [8, 42, 46] consider neighboring relationships between primitives and encode them into an adjacency graph, which is adapted to regularize output meshes. Slicing strategies [6, 7, 3, 33, 35, 13] extend the extracted primitives and slice them to construct a spatial partition for candidate facets. Connectivity strategies are typically efficient but suffer from linkage errors of adjacent graphs generated by incomplete inputs. Arikan *et al*. [1] remedied the incomplete part by a user-guided interactive method that is labor-intensive especially for large-scale measurement data, whereas Lafarge *et al*. [26] provided an automatic solution that first consolidated the point clouds and then executed a min-cut optimization to extract dense triangle meshes.

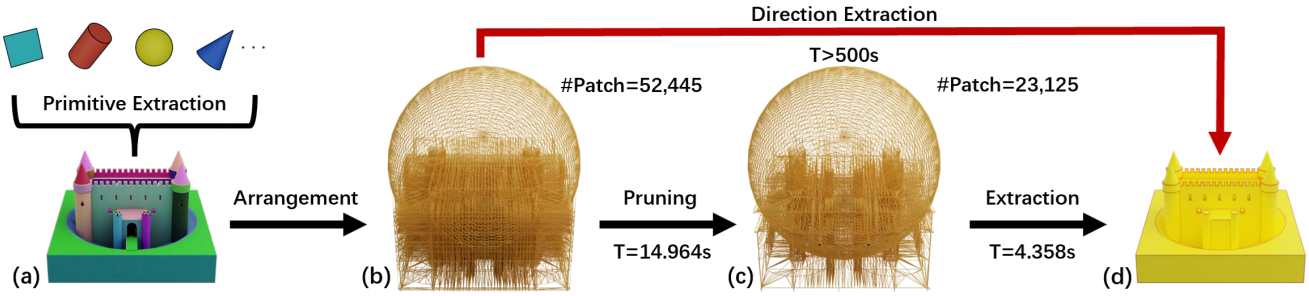


Figure 2: Overview of the proposed method. (a) Point cloud after primitive extraction; (b) Accomplished space partitioning; (c) Active primitive patches after patch pruning; (d) The reconstructed surface.

Alternatively, slicing strategies deal with incomplete models by means of extending primitives. Nan *et al*. [35] performed an exhaustive partition to compensate for all possible missing and thus guaranteed a manifold output via BLP. Nevertheless, this manner typically leads to high computational complexity for the subsequent surface extraction. To circumvent this shortcoming, Fang *et al*. [13] proposed a hybrid method that combines the connectivity analysis with the slicing process to make a limited partition, while Bauchet *et al*. [3] adopted a kinetic data structure [16, 15] to monitor the growing of convex piecewise planar polygons until they collide, which also results in a limited partition. Unlike previous approaches, we propose a simper partition by considering five types of primitives simultaneously rather than merely planar shapes. Moreover, we develop a propagation strategy inspired by Bauchet *et al*. [3] to further simplify and speed up the surface extraction.

Learning-based reconstruction. With the advancements in deep learning, there have been several notable works that utilize learning-based techniques to reconstruct meshes. Voxel-inspired methods, such as those presented in references [38] and [10], train a network of models to convert voxels into implicit surface representations. To attain more accurate representations, SIREN [45] and DIGS [4] used *signed distance functions* as implicit surface representations, and then extracted meshes via marching cubes [30]. Points2Surf [12] employed a patch-based learning framework to directly attain meshes from point clouds without the need for normals, while DSE [40] adopted 2D Delaunay triangulations to ensure manifold meshes and BSP-Net [9] output compact polygonal meshes with arbitrary topology. However, most of these methods depend on implicit surface representations, which may not explicitly capture the *underlying structures* such as quadric primitives of objects. In addition, it is hard to perform model editing tasks for surfaces generated by implicit reconstructions.

3. Methodology

The proposed method consists of three major steps as illustrated in Fig. 2. Given an input point cloud $\mathcal{P} = \{p_j \in \mathbb{R}^3\}_{j=1}^{N}$ (Fig. 2(a)), we first partition $\mathcal{P}$ by a set of proxy meshes defined by the extracted geometric primitives and then apply a mesh arrangement operation to get candidate manifold patches (Fig. 2(b)) in Section 3.1. Later, a fast propagation mechanism presented in Section 3.2 is used to prune patches that are inactive (Fig. 2(c)) to accelerate the subsequent surface extraction procedure. Finally, we deliver a manifold and watertight output surface (Fig. 2(d)) by formulating a patch-induced optimization under the BLP framework in Section 3.3.

3.1. Space Partitioning

The goal of this step is to generate a finite space partitioning by arranging proxy meshes of geometry primitives extracted from the input point cloud $\mathcal{P}$.

Primitive extraction. We first utilize an efficient region growing strategy introduced in [39] to attain the rough segmentation components $\mathcal{S} = \{s_i \subset \mathbb{R}^3\}_{i=1}^{M}$ and the corresponding primitives $\mathcal{Q} = \{q_i \subset \mathbb{R}^3\}_{i=1}^{M}$ from $\mathcal{P}$, where q_i is the supporting primitive of s_i inversely, each point $p_j \in s_i$ is the supporting point of q_i. Let $\mathcal{L} = (\mathcal{S}, \mathcal{Q})$ be the initial layout of the extracted primitives, we further refine $\mathcal{L}$ via the variational shape approximation based on the ℓ^2 and the $\ell^{2,1}$ metrics [11, 47], which aims to optimize the following expression

$$\min_{\mathcal{L}} \mathcal{F}(\mathcal{L}) = \sum_{i=1}^{M} \sum_{p_j \in s_i} dist(p_j, q_i), \tag{1}$$

where $dist(p_j, q_i)$ stands for the hybrid metric that measures the cost of the point $p_j \in s_i$ with respect to the corresponding primitive $q_i \in \mathcal{Q}$ in terms of both the spatial ℓ^2 distance and the normal $\ell^{2,1}$ deviation. Fig. 2(a) presents the primitive extraction result based on five types of primitives that we are currently considering. Noted that the proposed framework is not constrained by the above detected primitive types. Instead, it can be integrated with powerful detection algorithms that support a wider variety of primitive types, enabling more complex reconstructions with better accuracy.

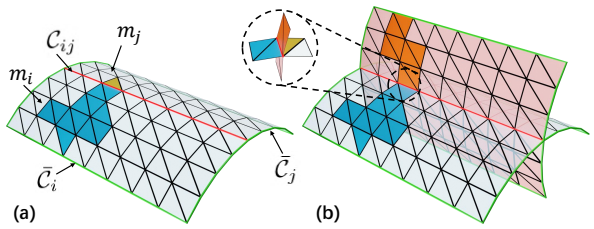


Figure 3: Visual description of patches and (non–) boundary curves. (a) $\mathcal{C}_{ij}$ is the non-boundary curve generated by patches m_i and m_j, while $\bar{\mathcal{C}}_i$ and $\bar{\mathcal{C}}_j$ are the boundary curves of m_i and m_j, separately. (b) For each non-boundary curve, there are four patches around it.

Proxy mesh arrangement. After the optimization of the extracted primitives, we generate the discrete approximation to all primitives via proxy meshes. These meshes are trimmed by the bounding box of $\mathcal{P}$ and then all meshes are processed by a mesh arrangement [50]. The mesh arrangement results are a set of candidate *primitive patches* $\mathcal{M} = \{m_i \subset \mathbb{R}^3\}_{i=1}^{K}$ as illustrated in Fig. 2(b), which partition the current working space into a series of cells that form a closed manifold without self-intersections. Here each patch m_i represents a subset of the input meshes and is the maximal connected set of triangle meshes as illustrated in Fig. 3. Due to the connectivity of patches, each m_i may embrace several *non-boundary* curves $\mathcal{C}_{ij}$ (colored red in Fig. 3(a)) that are shared with other primitive patches, which are defined as

$$\mathcal{C}_{ij} = m_i \bigcap m_j \,(i \neq j)\,, m_i, m_j \in \mathcal{M}, \tag{2}$$

whereas at most one *boundary* curve $\bar{\mathcal{C}}_i$ (colored green in Fig. 3(a)). For example, there are in total of four neighboring patches that share the same $\mathcal{C}_{ij}$ in Fig. 3(b). Since we perform space partitioning based on primitive patches rather than discretizing each of them as multiple piecewise planes, the developed algorithm indeed is several orders of magnitude faster than previous plane-induced approaches. The efficiency superiority of our method is also quantitatively validated by the following experiments.

3.2. Pruning of Candidate Primitive Patches

It is known that the computational complexity of the ultimate surface extraction in shape reconstruction regarding the size of patches is exponential [44]. To expedite the subsequent optimization process, we further propose a *pruning* mechanism which effectively circumvents the time-consuming and memory-intensive global collision detection to classify current primitive patches as *active* and *inactive*, from which we prune inactive ones to reduce the search space for surface extraction. As illustrated in Fig. 2, the significant time gap between the two routes(black and red) demonstrates the effectiveness of our pruning mechanism. The proposed procedure is composed of two steps, initialization and pruning, as presented subsequently.

Algorithm 1: Pseudo-code of the propagation process

Input: Priority queue Q of representative triangles.
Output: Active primitive patches $\tilde{\mathcal{M}}$.

```
Initialize propagation cost of each triangle as c = 0.
while Q ≠ ∅ do
    (t, c) ← Q.pop() and t̂ ← t.
    for each triangle edge e ∈ t̂ do
        for triangle t_e ∩ t = e do
            if prim(t_e) == prim(t) & t_e is
              inactive & t_e is not blocked then
                Compute the height h_e of t_e on e.
                Q.push((t_e, c + h_e)).
```

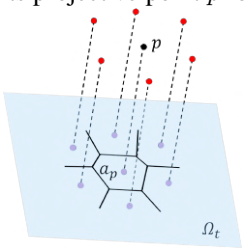
Initialization. To describe the point distribution of the input point cloud as fidelity as possible, we first select a set of representative triangle meshes denoted as $\mathcal{I}$ from $\mathcal{M}$ by means of the triangle coverage $Cover(t)$which is defined as

$$Cover(t) = \frac{Valid(t)}{Area(t)}, \tag{3}$$

where $Area(t)$ stands for the area of a triangle mesh t and $Valid(t)$ is formulated as

$$Valid(t) = \sum_{p \in \mathcal{P}_t} 1(\|p - p'\|_2 < \epsilon) a_p, \tag{4}$$

where $\mathcal{P}_t \subset \mathcal{P}$ encompasses the measurement points that their projections are attached to t, $\|p - p'\|_2$ denotes the ℓ^2 Euclidean distance from p to its projective point p' on t, 1 is the indicator function, and a_p is the corresponding geodesic Voronoi area of p, which is calculated as in the inset: p and its k-nearest neighbors (*i.e.*, hard red points) are first projected to the supporting plane Ω_t where its attached triangle mesh t lies on, then a_p is equivalent to the area of the 2D Voronoi cell [2]. t is termed *representative* if its $Cover(t)$ surpasses a user-specified value. We populate all representative meshes of $\mathcal{I}$ into a *priority queue* Q with an ascending order.



Pruning. Next, representative triangle meshes are propagated to enable a *limited* partition to speedup the final surface extraction. To this end, all triangles of Q are initially labeled as inactive prior to entering the loop of the priority queue. For each round during the loop, we first pop the top element, for instance, (t_i, c_i), out of the priority queue and prompt t_i to be an active state $\hat{t}_i$. The propagation cost c_i of a triangle mesh t_i is defined as the height

h_i from its vertex to the base edge shared with another adjacent triangle. Sequentially, we propagate all the triangles that share both the same edge and the same supporting primitive with t_i and pop the update elements out of Q. For instance, for edges that are not located on non-boundary curves, there are at most three update elements (t_{i1}, c_i+h_1), $(t_{i2}, c_i + h_2)$, $(t_{i3}, c_i + h_3)$, whereas, for edges that coincide non-boundary curves, we perform a fast *local* collision detection to check whether there exist active meshes, if yes then we evoke barricades and stop the propagation process. The above propagation proceeds until Q becomes empty. We summarize the proposed propagation process in Algorithm 1 to make it more understandable. Consequently, a primitive patch is *active* only if its triangles are all active, otherwise, it is still *inactive*. We use $\tilde{\mathcal{M}}$ to represent the set of active patches. The patches in Fig. 2(b) after pruning are shown in Fig. 2(c). As observed, the size of candidate patches has been significantly reduced compared with Fig. 2(b).

Like [3], we also allow for the propagation crossing primitive patches $\mathcal{G}$ times (with $\mathcal{G} = 2$ in default) at most. This can deal with small parts that may have been missed during the primitive detection. However, in contrast to previous global collision detection manners [3], our local strategy preserves much better the propagation efficiency.

3.3. Surface Extraction

Finally, we adopt a *Binary Linear Programming* (BLP) analogous to [35] to extract the surface model from $\tilde{\mathcal{M}}$. Specifically, the state of patches in $\tilde{\mathcal{M}}$ is encoded as $\mathcal{X}_{\tilde{M}} = \{x_{\tilde{m}} | \tilde{m} \in \tilde{\mathcal{M}}, x_{\tilde{m}} \in \{0, 1\}\}$, where the binary discrete variable $x_{\tilde{m}}$ indicates whether the patch $\tilde{m}$ is selected $(x_{\tilde{m}} = 1)$ or not $(x_{\tilde{m}} = 0)$. To ensure the resulting surface manifold and watertight, we constrain the curves of the selected patches to be all non-boundary, and thus the two patches sharing the same non-boundary curve are either selected at the same time or both not selected. Consequently, our objective function is defined as

$$\min_{\mathcal{X}_{\tilde{\mathcal{M}}}} \quad \mathbf{U}(\mathcal{X}_{\tilde{\mathcal{M}}}) + \lambda \mathbf{B}(\mathcal{X}_{\tilde{\mathcal{M}}}),$$
$$s.t. \begin{cases} \sum_{\tilde{m}_i \cap \tilde{m}_j \neq \emptyset, i \neq j} (x_{\tilde{m}_i} + x_{\tilde{m}_j}) = 0 \ or \ 2, \tilde{m}_i, \tilde{m}_j \in \tilde{\mathcal{M}} \\ x_{\tilde{m}_i} \in \{0, 1\}, \quad i = 1, 2, 3 \ldots |\tilde{\mathcal{M}}|. \end{cases} \tag{5}$$

where $\mathbf{U}(\mathcal{X}_{\tilde{\mathcal{M}}})$ and $\mathbf{B}(\mathcal{X}_{\tilde{\mathcal{M}}})$ are penalty terms regarding the patches and the neighboring patch pairs, respectively, and λ is a balance coefficient.Moreover$\mathbf{U}(\mathcal{X}_{\tilde{\mathcal{M}}})$ encourages the selection of active patches that are with high fidelity in the form of

$$\mathbf{U}(\mathcal{X}_{\tilde{M}}) = \sum_{\tilde{m} \in \tilde{\mathcal{M}}} x_{\tilde{m}} \cdot \left(\frac{\mathcal{A}_{\tilde{m}} - \bar{\mathcal{A}}_{\tilde{m}}}{\mathcal{A}} - \frac{|\mathcal{P}_{\tilde{m}}|}{|\mathcal{P}|} \right). \tag{6}$$

Here $\mathcal{A}_{\tilde{m}}$ and $\bar{\mathcal{A}}_{\tilde{m}}$ denote the area and the coverage area of the patch $\tilde{m}$, respectively. The coverage area of $\tilde{m}$ is the summation of the coverage area of all its triangles. $\mathcal{A}$ is a normalization factor equivalent to the sum of the area of all patches. $\mathcal{P}_{\tilde{m}}$ is the set of points whose projections are locating in $\tilde{m}$. On the other hand, $\mathcal{B}(\tilde{\mathcal{X}})$ is designed to control the model complexity. Specifically, it put penalties to the non-boundary curve length of two patches:

$$\mathcal{B}(\mathcal{X}_{\tilde{\mathcal{M}}}) = \frac{1}{L} \sum_{i,j} |\mathcal{C}_{ij}| \cdot x_{\tilde{m}_i} \cdot x_{\tilde{m}_j}, \tag{7}$$

where $|\mathcal{C}_{ij}|$ denotes the length of the curve shared by two active patches $\tilde{m}_i$ and $\tilde{m}_j$ with different supporting primitives and L is the sum of the length of all curves.Fig. 2(d) presents the final reconstruction surface by BLP, which not only generates a manifold and watertight shape but also successfully preserves the intrinsic primitive structures.

4. Experimental Evaluations and Discussions

Implementation details and metrics. In this section, we execute extensive experiments to validate the advantages of the developed method in terms of both reconstruction precision and efficiency and make various comparisons with state-of-the-art approaches. Like [3], the precision is quantitatively evaluated by the *symmetric mean Hausdorff distance* (SMH) between the measurement point clouds and the reconstruction meshes (values are normalized by the diagonal length of the models' bounding box and multiplied by a factor of 10^2), while the efficiency is assessed by the running time. Our algorithm is implemented in C++ using the geometry processing library *libigl* [22]. We adopt the mathematical programming solver Gurobi [18] to optimize Eq. 5. All reconstruction results are rendered by Blender [14] for better visualization. Our experiments are conducted on a desktop with Intel(R) Core i7-7700K CPU@4.2GHz and 48GB RAM.

Reconstruction effect. Our algorithm has demonstrated good reconstruction results from point clouds to manifold meshes on various CAD objects as illustrated in Fig. 4. The input point clouds presented in Fig. 4(a)–(h) are either sampled from real-world CAD models or courtesy by publicly available benchmark datasets [41, 51]. As observed, there are multiple explicit intrinsic primitive structures in each model. Instead of approximating them directly by piecewise planar primitives, our algorithm is capable of reconstructing the intrinsic primitives faithfully even for quite challenging or complicated models. For instance, the rich spheral and conic structures existing in the `Minarets` model of Fig. 4(b) require careful treatments, yet our method is still able to achieve highly accurate reconstructions while preserving all structural details successfully. At the same time,

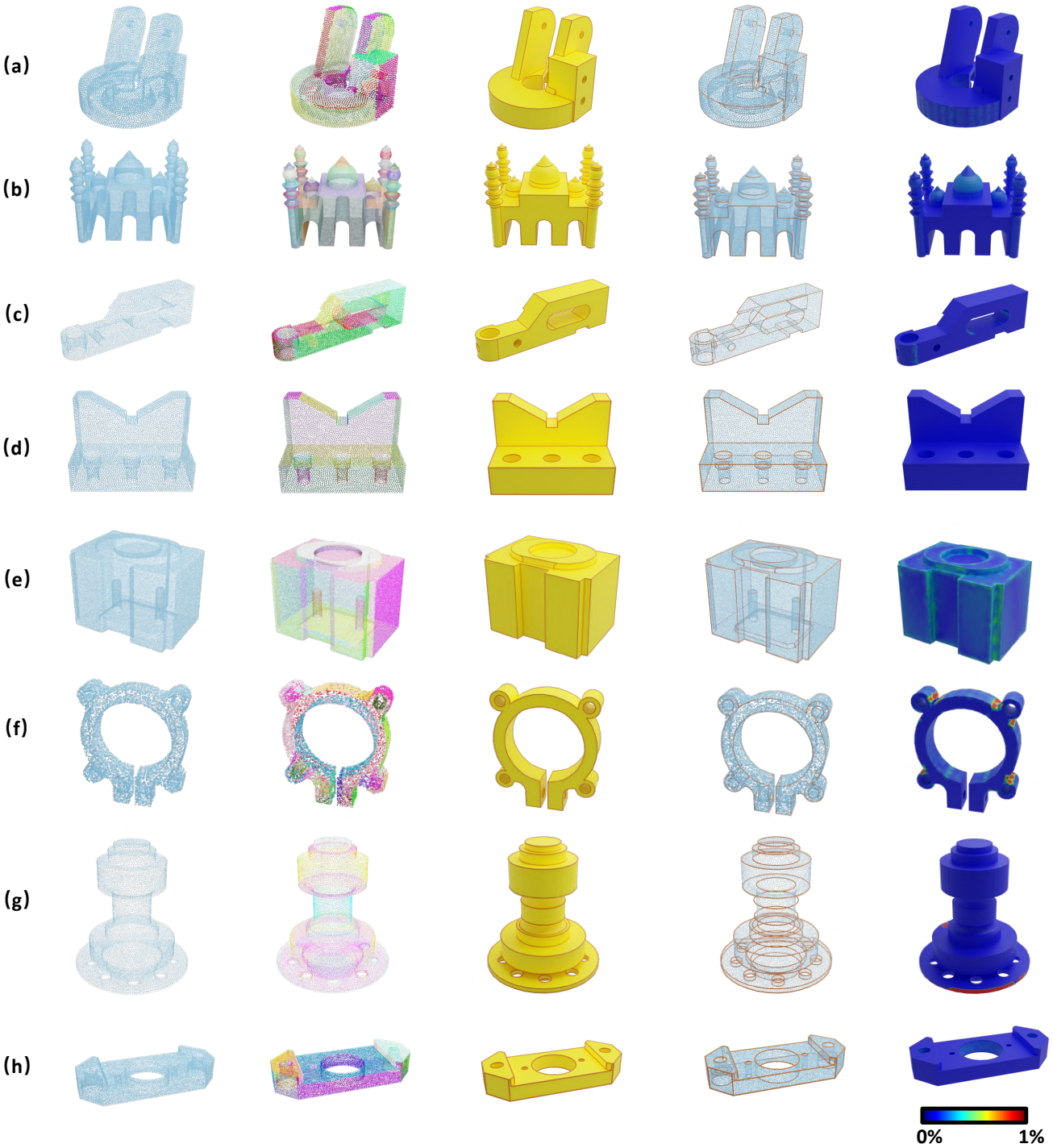


Figure 4: Reconstruction results by the proposed method. From left to right: Input measurement point clouds, extracted geometric primitives (different colors stand for different primitives), output reconstruction models, recovered sharp geometry along with the original point clouds, and colored models showing the error distribution regarding the Hausdorff distance. Our method produces high-quality reconstructions and preserves the intrinsic primitive structures as well as the sharp features well.

our method produces high-quality sharp feature lines attained by patch intersections, as shown in the forth column of Fig. 4. This is an important advantage for many practical applications such as sketch-style rendering, as it ensures that the reconstructed object retains its original shape and does not appear distorted or blurry. The fifth column of

Table 1: Statistical results of the quantity and the runtime (in second) during reconstruction of the CAD models in Fig. 4.

Metrics \ Models	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
# Points of the input point cloud	10k	593k	10k	10k	74k	7k	16k	14k
# Extracted primitives	28	53	21	20	34	32	29	27
# Generated candidate patches	3646	6024	1030	723	4970	417	750	1092
# Active patches after pruning	2245	3369	741	417	2586	342	472	667
Patch decrease (%)	38.426	44.073	28.058	42.324	47.968	17.986	37.067	38.919
# Determined active patches of the output model	605	915	191	93	533	99	86	114
Mesh arrangement (s)	7.853	21.752	2.463	1.727	5.236	2.935	5.431	1.421
Patch pruning (s)	0.927	15.421	0.792	0.476	3.043	0.488	1.131	0.726
Surface extraction (s)	1.288	2.086	0.659	0.091	1.112	0.157	0.314	0.137

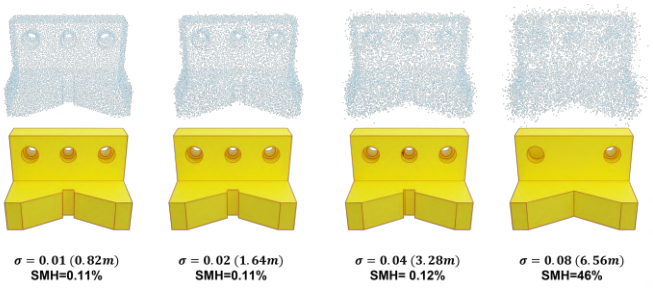


Figure 5: Robustness test against noise disturbance. Top: Input point clouds with increasing noise. Bottom: Our reconstruction results along with the noise level σ and the reconstruction error SMH.

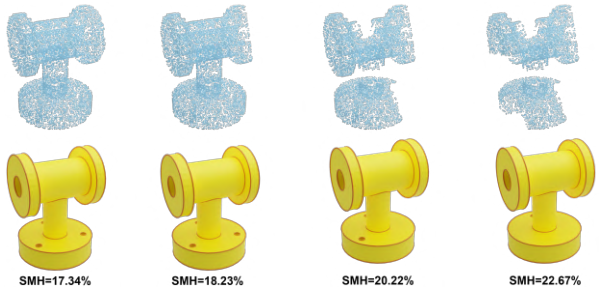


Figure 6: Robustness test against occlusion data. Top: Input point clouds with increasing occlusions. Bottom: Our reconstruction results along with the reconstruction error.

Fig. 4 visualizes the reconstruction error of our algorithm with respect to the Hausdorff distance, where our method consistently produces accurate approximations for all objects, demonstrating its stability and effectiveness.

Statistical results. Table 1 summarizes the quantitative results of our proposed method for the models in Fig. 4. Our algorithm significantly reduces candidate patches, especially for complex models like Fig. 4(b),(d),(e), where the patch decrease ratio exceed 40%. In terms of efficiency, patch pruning and surface extraction outperform mesh arrangement, and further improvements can be made by exploring a more efficient intersection mechanism.

Robustness assessment. We also assess the ability of the proposed algorithm against *noise disturbance* and *occlusion data*. As depicted in Fig. 5, we increase the standard deviation σ of Gaussian noise from $0.01(0.82m)$ to $0.04(3.28m)$, where values in parentheses are real intensity relative to the bounding sphere's radius of the model. Our method consistently generates accurate reconstructions in terms of both *topological structures* and *approximation distance* (SMH), suggesting its high resilience to noise. Even when $\sigma = 0.08(6.56m)$, which has caused significant topological alteration of the original point cloud, our method still maintains the overall structures of the object. Additionally, we verify the robustness of our method to occlusion data. Fig. 6 reports our reconstruction results after progressively erasing point clouds. As observed, our method shows remarkable resistance to occlusions and has successfully compensated the missing structures. Conversely, this is also quite helpful to complete the imperfect point clouds by sampling points in the reconstruction models.

Table 2: Comparisons with state-of-the-art approaches, where "w" and "w/o" indicate with and without the patch pruning, separately.

Metrics	iPSR[20]	SIREN[45]	DSE[40]	KSR[3]	Ours	
					w	w/o
SMH (%)	4.23	41.79	3.15	169.44	2.45	2.45
Time (s)	201.589	1208.124	303.865	570.468	45.5	56.777

Comparisons. Subsequently, we conduct a comparative analysis of our method with representative state-of-the-art approaches, including densely implicit reconstruction methods iPSR [20], DSE [40], and SIREN [45], as well as the plane-assembly approach KSR [3]. The quantitative comparison results in terms of reconstruction accuracy and efficiency on the `Cheese` model is presented in Table 2. The results demonstrate that our method achieves competitive reconstruction accuracy with implicit approaches iPSR and DSE, and outperforms SIREN and KSR by a large margin. Additionally, our algorithm performs significantly faster than competitors, even without patch pruning. The qualitative reconstruction results presented in Fig. 7 show that benefiting from the introduced primitive assembly, our

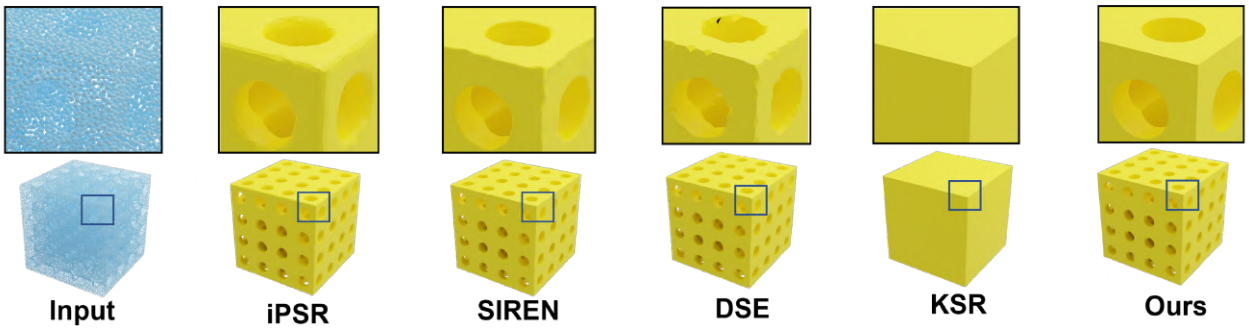


Figure 7: Qualitative comparisons with state-of-the-art approaches. Our method enables the shape reconstruction faithfully and successfully recovers the sharp geometry (*e.g.*, feature lines and corners) from unorganized point clouds.

Table 3: Quantitative comparisons on the ABC300 dataset.

Metrics \ Methods	iPSR[20]	SIREN[45]	DSE[40]	KSR[3]	ComplexGen[17]	CAPRI-Net[48]	Ours
SMH (%)	2.71	24.89	4.06	26.43	24.24	15.87	4.81
Time (s)	24.064	1565.217	32.800	4.208	553.089	183.844	3.728

method successfully recovers sharp features from unorganized point clouds, which is a salient advantage compared to implicit methods.

We also conduct quantitative comparisons with several predecessors (additionally including CAPRI-Net [48] and ComplexGen [17]) on 300 CAD models from the ABC dataset (abbreviated as ABC300). Table 3 demonstrates that our method achieves consistently competitive reconstruction accuracy with implicit approaches such as iPSR and DSE (while also providing advantages for sharp feature recovery and model editing), and significantly surpasses KSR, CAPRI-Net, and CompleGen. Furthermore, our method is considerably faster than its competitors.

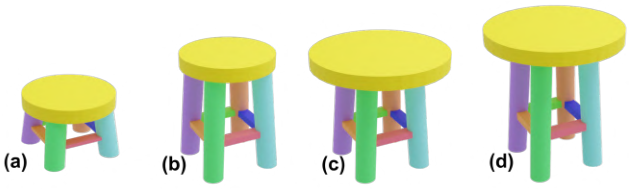


Figure 8: Model editing based on our structure-aware reconstruction algorithm. (a) The original reconstruction object. (b)–(d) The newly edited results.

Balance parameter. We investigate the impact of λ on surface extraction by varying its values to different settings (see Fig. 9(a)). It can be seen that increasing λ brings the reduction of the complexity of the output model. However, this also degrades the reconstruction accuracy. For example,when λ is set to 0.7, the inner hollows of the input point clouds disappear.

Application to model editing. Our reconstruction method can be employed for flexible model editing by taking advantage of the structure-aware merits. As shown in Fig. 8, since each quadratic structure of the object is preserved along with their detailed geometric parameters, one can modify the reconstruction shape by simply specifying various parameters. For instance, by re-defining the radius or height of the cylindrical primitives in Fig. 8(a), we can resize them to our desirable shapes (Fig. 8(b)–(d)).

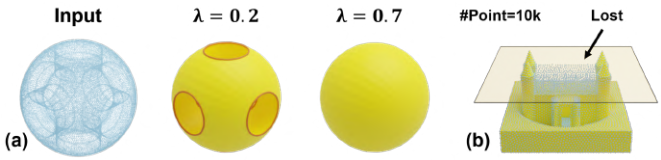


Figure 9: (a) Impact of parameter λ. As λ is increased, the complexity of the output model decreases. (b) In sparse input scenarios, small primitives can be easily lost, leading to failures.

Limitations. Our approach primarily focuses on the rapid and accurate reconstruction of representative primitive structures in CAD models. However, as shown in Fig 9(b), our reconstruction quality may deteriorate for inputs lacking explicit primitives. Additionally, inadequate primitive extraction due to severe occlusion or noise can degrade the quality of the reconstruction results.

5. Conclusion and Future Work

We propose an accurate and fast shape reconstruction pipeline for converting disordered 3D point clouds into manifold, watertight surfaces. Our method preserves representative intrinsic primitive structures in CAD systems through primitive assembly, avoiding dense implicit reconstructions and extending piecewise planar approximations. We also develop an effective patch pruning mechanism, greatly expediting surface extraction by reducing patch intersections. Additionally, our method is highly resilient to noise and can reliably recover sharp object geometry. Extensive experiments show our algorithm's superiority over competing methods.

In our future work, we plan to expand the current primitive types to general primitives, and even spline surfaces, in order to efficiently process more complex CAD models.

Acknowledgements

This work was partially funded by the National Key Research and Development Program (2021YFB1715900), the CAS Project for Young Scientists in Basic Research (YSBR-034), the National Natural Science Foundation of China (62172415, 62272277, 12022117), and the HKU-100 Research Award.

References

[1] Murat Arikan, Michael Schwärzler, Simon Flöry, Michael Wimmer, and Stefan Maierhofer. O-snap: Optimization-based snapping for modeling architecture. *ACM Trans. Graph.*, 32(1):6:1–6:15, 2013. 2

[2] Gavin Barill, Neil G Dickson, Ryan Schmidt, David IW Levin, and Alec Jacobson. Fast winding numbers for soups and clouds. *ACM Trans. Graph.*, 37(4):43:1–43:12, 2018. 4

[3] Jean-Philippe Bauchet and Florent Lafarge. Kinetic shape reconstruction. *ACM Trans. Graph.*, 39(5):156:1–156:14, 2020. 1, 2, 3, 5, 7, 8

[4] Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. Digs: Divergence guided shape implicit neural representation for unoriented point clouds. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 3

[5] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. *Comput. Graph. Forum*, 36(1):301–329, 2017. 1, 2

[6] Alexandre Boulch, Martin de La Gorce, and Renaud Marlet. Piecewise-planar 3d reconstruction with edge and corner regularization. *Comput. Graph. Forum*, 33(5):55–64, 2014. 2

[7] Anne-Laure Chauve, Patrick Labatut, and Jean-Philippe Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1261–1268, 2010. 1, 2

[8] Jie Chen and Baoquan Chen. Architectural modeling from sparsely scanned range data. *Int. J. Comput. Vis.*, 78(2):223–236, 2008. 2

[9] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 3

[10] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5939–5948, 2019. 3

[11] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. *ACM Trans. Graph.*, 23(3):905–914, 2004. 1, 3

[12] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. Points2surf learning implicit surfaces from point clouds. In *Eur. Conf. Comput. Vis.*, pages 108–124, 2020. 3

[13] Hao Fang and Florent Lafarge. Connect-and-slice: an hybrid approach for reconstructing 3d objects. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 13490–13498, 2020. 1, 2, 3

[14] Blender Foundation. Blender - a 3D modelling and rendering package. http://www.blender.org. 5

[15] Leonidas Guibas. Kinetic data structures. In *Handbook of Data Structures and Applications*, pages 377–388. 2018. 3

[16] Leonidas Guibas, Lyle Ramshaw, and Jorge Stolfi. A kinetic framework for computational geometry. In *IEEE Annual Symposium on Foundations of Computer Science*, pages 100–111, 1983. 3

[17] Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Trans. Graph.*, 41(4):129:1–129:18, 2022. 8

[18] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. https://www.gurobi.com. 5

[19] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proc. ACM SIGGRAPH*, pages 71–78, 1992. 1

[20] 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):128:1–128:13, 2022. 7, 8

[21] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph.*, 28(5):1–7, 2009. 1

[22] Alec Jacobson and Daniele Panozzo. Libigl: Prototyping geometry processing research in c++. In *SIGGRAPH Asia 2017 courses*, pages 1–172. 2017. 5

[23] Adrien Kaiser, Jose Alonso Ybanez Zepeda, and Tamy Boubekeur. A survey of simple geometric primitives detection methods for captured 3d data. *Comput. Graph. Forum*, 38(1):167–196, 2019. 2

[24] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Comput. Graph. Forum (Proceedings of SGP)*, page 61–70, 2006. 1

[25] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13, 2013. 1

[26] Florent Lafarge and Pierre Alliez. Surface reconstruction through point set structuring. *Comput. Graph. Forum*, 32(2pt2):225–234, 2013. 2

[27] Minglei Li, Liangliang Nan, Neil Smith, and Peter Wonka. Reconstructing building mass models from UAV images. *Comput. Graph.*, 54:84–93, 2016. 1

[28] Minglei Li, Peter Wonka, and Liangliang Nan. Manhattan-world urban reconstruction from point clouds. In *Eur. Conf. Comput. Vis.*, pages 54–69, 2016. 1

[29] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph.*, 30(4):52:1–52:12, 2011. 2

[30] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proc. ACM SIGGRAPH*, volume 21, pages 163–169. 3

[31] David Marshall, Gabor Lukacs, and Ralph Martin. Robust segmentation of primitives from range data in the presence of geometric degeneracy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(3):304–314, 2001. 2

[32] Aron Monszpart, Nicolas Mellado, Gabriel J Brostow, and Niloy J Mitra. Rapter: rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.*, 34(4):103:1–103:12, 2015. 2

[33] Claudio Mura, Oliver Mattausch, and Renato Pajarola. Piecewise-planar reconstruction of multi-room interiors with arbitrary wall arrangements. *Comput. Graph. Forum*, 35(7):179–188, 2016. 2

[34] Liangliang Nan, Andrei Sharf, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Smartboxes for interactive urban reconstruction. *ACM Trans. Graph.*, 29(4):93:1–93:10, 2010. 1

[35] Liangliang Nan and Peter Wonka. Polyfit: Polygonal surface reconstruction from point clouds. In *IEEE Int. Conf. Comput. Vis.*, pages 2353–2361, 2017. 1, 2, 3, 5

[36] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. In *IEEE Int. Conf. Robot. Autom. Mechatron*, pages 225–230, 2013. 1

[37] Sven Oesau, Florent Lafarge, and Pierre Alliez. Planar shape detection and regularization in tandem. *Comput. Graph. Forum*, 35(1):203–215, 2016. 2

[38] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 165–174, 2019. 3

[39] Tahir Rabbani, Frank Van Den Heuvel, and George Vosselmann. Segmentation of point clouds using smoothness constraint. *ISPRS J. PhotoGramm.*, 36(5):248–253, 2006. 2, 3

[40] Marie-Julie Rakotosaona, Paul Guerrero, Noam Aigerman, Niloy J. Mitra, and Maks Ovsjanikov. Learning delaunay surface elements for mesh reconstruction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 22–31, June 2021. 3, 7, 8

[41] Chiara Romanengo, Andrea Raffo, Yifan Qie, Nabil Anwer, and Bianca Falcidieno. Fit4cad: A point cloud benchmark for fitting simple geometric primitives in cad objects. *Comput. Graph.*, 102:133–143, 2022. 5

[42] Falko Schindler, Wolfgang Wörstner, and Jan-Michael Frahm. Classification and reconstruction of surfaces from point clouds of man-made objects. In *IEEE Int. Conf. Comput. Vis. Worksh.*, pages 257–263, 2011. 2

[43] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Comput. Graph. Forum*, 26(2):214–226, 2007. 2

[44] Gerard Sierksma and Yori Zwols. *Linear and integer optimization: theory and practice*. 2015. 4

[45] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Int. Conf. Neur. Info. Process. Sys.*, volume 33, pages 7462–7473, 2020. 3, 7, 8

[46] Marc Van Kreveld, Thijs Van Lankveld, and Remco C Veltkamp. On the shape of a set of points and lines in the plane. *Comput. Graph. Forum*, 30(5):1553–1562, 2011. 2

[47] Dong-Ming Yan, Wenping Wang, Yang Liu, and Zhouwang Yang. Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design*, 44(11):1072–1082, 2012. 3

[48] Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. Capri-net: Learning compact cad shapes with adaptive primitive assembly. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11768–11778, 2022. 8

[49] Mulin Yu and Florent Lafarge. Finding good configurations of planar primitives in unorganized point clouds. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6367–6376, 2022. 1, 2

[50] Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. Mesh arrangements for solid geometry. *ACM Trans. Graph.*, 35(4):39:1–39:15, 2016. 4

[51] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 5

[52] Marco Zuliani, Charles S Kenney, and BS Manjunath. The multiransac algorithm and its application to detect planar homographies. In *IEEE Int. Conf. Image Process.*, volume 3, pages III–153, 2005. 2