Hierarchical Space Partition for Surface Reconstruction

Supplementary Material

In this supplement, we present implementation details, analyze parameter sensitivity and ablation effects, demonstrate the method's generalizability, and discuss its limitations.

6. Implementation Details

6.1. Pseudo-code of Planar Visibility Calculations

This section mainly shows how to calculate the visibility ratio of planes. We model α-shapes as uniformly luminous objects and the keypoints as the point lights. We begin by introducing the notations used in this computation. Let:

- $\mathcal{P} = (P_1, \ldots, P_m)$ be the set of planar primitives;
- $\mathcal{A} = (A_1, \ldots, A_m)$ be the set of α-shape polygons;
- $\mathcal{K} = (K_1, \ldots, K_m)$ be the set of keypoints of plane P_i, treated as point light sources;
- $\mathcal{D} = (\mathbf{d}_1, \ldots, \mathbf{d}_{50})$ be a sequence of 50 unit direction vectors generated by Fibonacci sampling on the unit sphere, representing the directions of sampled light rays.
- $\mathcal{T}$ is a hierarchical collision detection structure built from $\mathcal{A}$, where each node is bounded by an Axis-Aligned Bounding Box (AABB).
- $\mathcal{N} = \{\mathbf{n}_1, \ldots, \mathbf{n}_m\}$ be the set of unit surface normals, where each $\mathbf{n}_i$ is the outward-pointing normal to the supporting plane P_i. Further, for a highly visible plane, we expect the normal to point to the visible half of the facet — the side where light does not intersect the model. Since normals are ambiguous, the final direction is determined by votes from scattered rays. Specifically, for the highly visible plane primitive P_i, let $\mathbf{n}_i$ denote the original normal direction. The number of votes supporting $\mathbf{n}_i$ as the final direction is N_{ni}^{+}, and those supporting $-\mathbf{n}_i$ are N_{ni}^{-}.
- N_h represents the number of highly visible planes, which is 0 at the beginning. N_h^{pre} represents N_h corresponding to the last iteration.
- $\mathcal{V} = (v_1, \ldots, v_m)$ be the output visibility ratio of planes.
- $\mathcal{Q} = (Q_1, \ldots, Q_m)$ be the output original visible labels of each planes. If the visibility ratio v_i is larger than 0.5, the plane P_i is defined as original highly visible (Q_i = 1), otherwise, barely visible (Q_i = 0).

The calculation of a planar visibility ratio is an iterative optimization process. A ray is considered visible if it satisfies one of the following two conditions: i) the ray does not intersect the model $\mathcal{T}$; ii) if the plane where the ray first intersects the model $\mathcal{T}$ has been determined to be highly visible, and the ray lies on the visible side of this plane, then the ray is also considered visible. We iterate on the above process until the number of strong visible planes no longer increases.

Algorithm 1 Plane visibility calculations

Require : $\mathcal{A} \leftarrow (A_1, \ldots, A_m)$
Require : $\mathcal{K} \leftarrow (K_1, \ldots, K_m)$
Require : $\mathcal{D} \leftarrow (\mathbf{d}_1, \ldots, \mathbf{d}_{50})$
Require : $\mathcal{N} \leftarrow \{\mathbf{n}_1, \ldots, \mathbf{n}_m\}$
Require : $N_h \leftarrow 0, N_h^{pre} \leftarrow 0$
Ensure : $\mathcal{V} \leftarrow (0, 0, \ldots, 0)$
Ensure : $\mathcal{Q} \leftarrow (0, 0, \ldots, 0)$

procedure PLANE VISIBILITY CALCULATIONS
  $\mathcal{T} \leftarrow$ new `AABB_Tree` from input geometry $\mathcal{A}$
  repeat
    $N_h^{pre} \leftarrow N_h$
    for $P_i \in \mathcal{P}$ where $L_i = 0$ **do**
      $C_i \leftarrow 0, \quad C_n \leftarrow 0$
      for $k_{ij} \in K_i$ **do**
        for $d_t \in \mathcal{D}$ **do**
          Emit ray r_t from k_{ij} in direction d_t
          if r_t does not intersect in $\mathcal{T}$ **then**
            $C_n \leftarrow C_n + 1$, COUNTS($\mathbf{n}_i, d_t$)
          else
            Let p_k be the first intersection plane of r_t with $\mathcal{T}$
            if $Q_k = 1 \wedge \mathbf{n}_k \cdot d_t < 0$ **then**
              $C_n \leftarrow C_n + 1$, COUNTS($\mathbf{n}_i, d_t$)
            else
              $C_i \leftarrow C_i + 1$
            end if
          end if
        end for
      end for
      if $C_i < C_n$ **then**
        $Q_i \leftarrow 1$, $N_h \leftarrow N_h + 1$
        if $N_{ni}^{+} < N_{ni}^{-}$ **then**
          $\mathbf{n}_i \leftarrow -\mathbf{n}_i$
        end if
      end if
      $v_i \leftarrow C_i/(C_i + C_n)$
    end for
  until $N_h^{pre} = N_h$
end procedure

procedure COUNTS($\mathbf{n}_i, d_t$)
  if $\mathbf{n}_i \cdot d_t > 0$ **then**
    $N_{ni}^{+} \leftarrow N_{ni}^{+} + 1$
  else
    $N_{ni}^{-} \leftarrow N_{ni}^{-} + 1$
  end if
end procedure

6.2. Pseudo-code of Missing Planes Recovery

The algorithm of this part is mainly divided into two steps. The first is the selection of singular line segments, and then is the planes fitting based on the singular line segments. Let:

- $\mathcal{P} = (P_1, \ldots, P_m)$ be the set of planar primitives;
- $\mathcal{L} = (L_1, \ldots, L_m)$ be the set of intersection line sets, where each L_i contains the intersection lines between P_i and every other plane, i.e. $L_{ij} = P_i \cap P_j$ for $j \neq i$.
- $\mathcal{S} = (S_1, \ldots, S_m)$ be the collection of boundary segment sets for each plane primitive, where:
  - $S_i = (s_{i1}, \ldots, s_{in_i})$ represents the boundary segments of plane primitive P_i (with n_i segments).
  - $\mathbf{n}_{ij}$ denotes the normal vector of segment s_{ij}.
  - I_{ij} is the set of interior points of segment s_{ij}.
  - c_{ij} denotes the centroid of the line segment s_{ij}, which is calculated as the average position of all points contained in I_{ij}.
  - f_{ij} is the initial plane of segment s_{ij}, determined by point c_{ij} and normal vector $\mathbf{n}_{ij}$.
  - $N_{ij} = \{s_{pq}\}$ as the neighborhood set of boundary segment s_{ij}, where two segments are mutual neighbors if their corresponding plane primitives P_i and P_p are second-order adjacent (see Section 3.3).
  - $d_{ij} = 0$ or 1. $d_{ij} = 0$ indicates that the line segment s_{ij} has not been visited yet, while $d_{ij} = 1$ means that the line segment has already been visited.
- r_d: distance threshold, r_a: angle threshold
- $\mathcal{G}$ be the output set of singular segments (see Section 3.3).
- $\mathcal{N}$ be the output recovered missing planes.

6.3. Pseudo-code of Hierarchical Space Partition

We design a hierarchical space partition strategy based on a kinetic data structure. Unlike the previously described scheme, we divide the planes into three different categories based on visibility (highly visible, barely visible, and invisible), corresponding to three levels. Planes at the current level will partition further based on the outcome only after planes at the immediately higher level have completed their spatial partition (i.e. hierarchical partition). At the same time, each plane is given a different growth speed according to its visibility. This hierarchical strategy produces lightweight and semantically meaningful 3D partitions, with substantially reduced algorithmic complexity compared to other methods. Let:

- $\mathcal{C}_h = (C_{h1}, \ldots, C_{hh})$ be the set of convex polygons of highly visible planes (planes on level 1). Let V_1 be the set of non-frozen vertices on level 1.
- $\mathcal{C}_b = (C_{b1}, \ldots, C_{bb})$ be the set of convex polygons of barely visible planes (planes on level 2). Let V_2 be the set of non-frozen vertices on level 2.
- $\mathcal{C}_m = (C_{m1}, \ldots, C_{mm})$ be the set of convex polygons of invisible planes (planes on level 3). Let V_3 be the set of non-frozen vertices on level 3.
- s_i be the growth speed of vertex v_i, which is positively correlated with its planar visibility ratio (see Section 3.5).
- $\mathcal{M}$ be the output partition of polyhedra.

Algorithm 2 Missing planes recovery

Require : $\mathcal{P} \leftarrow (P_1, \ldots, P_m)$
Require : $\mathcal{L} \leftarrow (L_1, \ldots, L_m)$
Require : $\mathcal{S} \leftarrow (S_1, \ldots, S_m)$
Require : r_d: distance threshold, r_a: angle threshold
Ensure : $\mathcal{G} \leftarrow \emptyset$
Ensure : $\mathcal{N} \leftarrow \emptyset$

procedure SINGULAR SEGMENTS SELECTION
  for $P_i \in \mathcal{P}$ **do**
    for $s_{ik} \in S_i$ **do**
      for $L_{ij} \in L_i$ **do**
        if $r_d < \text{dist}(c_{ik}, L_{ij}) \vee r_a < \text{angle}(s_{ik}, L_{ij})$ **then**
          Add s_{ik} to $\mathcal{G}$
        end if
      end for
    end for
  end for
end procedure

procedure PLANE FITTING
  Sort the segments in $\mathcal{G}$ in descending order of length.
  Initialize $d_j \leftarrow 0$ for all $s_j \in \mathcal{G}$
  for $s_{ik} \in \mathcal{G}$ where $d_{ik} = 0$ **do**
    Initialize empty queue Q
    count $\leftarrow 1$
    $f_{fit} \leftarrow f_{ik}$
    $Q.push(s_{ik})$
    $d_{ik} \leftarrow 1$
    while Q is not empty **do**
      $s_{current} \leftarrow Q.front()$
      $Q.pop()$
      neighbors $\leftarrow N_{current}$
      for each $s_j \in$ neighbors where $d_j = 0$ **do**
        if $\text{dist}(I_j, f_{fit}) < r_d \wedge \text{angle}(\mathbf{n}_j, f_{fit}) < r_a$ **then**
          $Q.push(s_j)$
          $d_j \leftarrow 1$
          count $\leftarrow$ count $+ 1$
          update_plane_model(f_{fit}, s_j)
        end if
      end for
    end while
    if count > 1 **then**
      $\mathcal{N} \leftarrow \mathcal{N} \cup \{f_{fit}\}$
    end if
  end for
end procedure

Algorithm 3 Hierarchical space partition

Require : $\mathcal{C}_h \leftarrow (C_{h1}, \ldots, C_{hh})$
Require : $\mathcal{C}_b \leftarrow (C_{b1}, \ldots, C_{bb})$
Require : $\mathcal{C}_r \leftarrow (C_{r1}, \ldots, C_{rr})$
Require: For each vertex v_i from all polygons, $s_i :=$ growth speed of v_i
Ensure : $\mathcal{M} \leftarrow \emptyset$

procedure HIERARCHICAL SPACE PARTITION
    Compute the bounding box of all convex polygons $\{\mathcal{C}_h, \mathcal{C}_b, \mathcal{C}_m\}$
    Set the bounding box as level_0 of the partition (space_0)
    Generate the initial non-frozen vertices sets $\{\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3\}$
    for $i \leftarrow 1$ **to** 3 **do**
        while $\mathcal{V}_i$ is not empty **do**
            Get the highest priority vertex v_j from $\mathcal{V}_i$
            Grow v_j with speed s_j based on space_$(i-1)$
            Determine the collision case (see Figure 6)
            Update $\mathcal{V}_i$ with sliding and/or frozen vertices
            Update the level_i of the partitions (space_i)
        end while
    end for
    Assemble adjacent facets of level_3 of the partitions (space_3) into polyhedra $\mathcal{M}$
end procedure

7. Parameter Sensitivity Analysis

In the proposed multi-stage reconstruction pipeline, several key parameters are involved, including (i) the weighting coefficients for plane categorization, (ii) the distance and angular thresholds used for missing plane recovery, and (iii) the weights applied during surface extraction. In this section, we analyze the sensitivity of these parameters and show that most of them are robust across diverse scenes, with only a few requiring scene-specific tuning.

7.1. Weighting Parameter for Plane Categorization

The weighting parameter w balances the influence between a node's own visibility evidence and the contextual information from its neighboring nodes during plane classification. Larger values of w place greater emphasis on neighborhood consistency, while $w = 0$ degenerates the classification to a purely local decision based solely on the node's visibility ratio. In contrast, overly large values (e.g., $w = 1$) may over-smooth the labeling by suppressing local evidence, which can negatively affect classification quality.

To evaluate the impact of w, we perform experiments on a validation set consisting of 10 indoor scenes selected from the ScanNet++V2 dataset. As ground-truth visibility labels (i.e., highly vs. barely visible planes) are not available, we adopt a semantic-based proxy derived from the dataset annotations. Specifically, planes belonging to structural elements (e.g., walls, floors, and ceilings) are treated as highly visible, while planes associated with non-structural objects (e.g., tables and chairs) are regarded as barely visible. This mapping serves as an empirical approximation rather than a strict correspondence; therefore, we evaluate semantic–visibility consistency instead of classification accuracy.

Let $\mathcal{P} = \{P_i\}_{i=1}^{N}$ denote the set of extracted planar regions. Each plane P_i is associated with a predicted visibility label $V_i \in \{\text{highly visible}, \text{barely visible}\}$ and a semantic label S_i. We define two semantic category sets:

$$\mathcal{S}_{\text{struct}} = \{\text{wall, floor, ceiling, } \ldots\}, \tag{1}$$

$$\mathcal{S}_{\text{non-struct}} = \{\text{table, chair, } \ldots\}, \tag{2}$$

A plane is considered consistent if its predicted visibility agrees with its semantic category. This is quantified by the indicator

$$M_i = \mathbb{I}[f(V_i) = g(S_i)], \tag{3}$$

where $f(\cdot)$ and $g(\cdot)$ map visibility and semantic labels to binary values, respectively. The overall consistency score is then computed as

$$C = \frac{1}{N} \sum_{i=1}^{N} M_i. \tag{4}$$

This consistency metric captures the empirical alignment between visibility-based plane classification and widely observed semantic priors in indoor environments, and is used as an auxiliary measure to assess the stability of the proposed classification with respect to w.

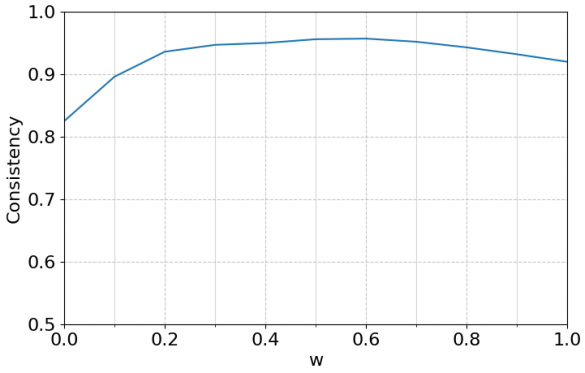

Figure 11. Impact of the parameter w. The consistency metric drops noticeably when w approaches the extremes, while higher consistency is achieved when w lies in the range of 0.4 to 0.7.

As shown in Figure 11, when w varies from 0 to 1, the consistency score decreases at both extremes. Visual inspection indicates that these regions correspond to noticeable classification errors. In contrast, the consistency remains relatively high and stable when $w \in [0.4, 0.7]$. Based on this observation, we set $w = 0.5$ in all experiments.

7.2. Thresholds for Missing Plane Recovery

The angle and distance thresholds in this module are used to extract singular segments, i.e., segments that cannot be sufficiently explained by existing planar structures, and to guide the fitting of corresponding missing planes. A larger threshold tends to classify more segments as being supported by existing planes, resulting in fewer singular segments and consequently fewer recovered missing planes. Conversely, an excessively small threshold may introduce a large number of false singular segments, leading to erroneous plane fitting and increased reconstruction error.

The angle threshold is determined following the strategy proposed in [12, 17], while the distance threshold is empirically set in accordance with common practices in point-cloud-based plane fitting literature [26, 40]. Given that our test datasets exhibit relatively uniform point density and consistent structural characteristics, the selected thresholds demonstrate stable performance across different scenes.

To evaluate parameter sensitivity, we adopt a synthetic masking-and-recovery protocol on a dedicated validation set consisting of 50 models randomly selected from the ScanNet++ V2 dataset. Due to the absence of ground-truth annotations for missing planes in real-world scans, we simulate partial observations by randomly removing a subset of existing planar structures along with their associated point sets. These removed planes serve as the ground truth for recovery, denoted as $\mathcal{G} = \{G_j\}_{j=1}^{M}$. Given the incomplete scans, our algorithm produces a set of recovered planes $\mathcal{R} = \{R_i\}_{i=1}^{K}$, which are then matched against $\mathcal{G}$ for quantitative evaluation.

To accommodate varying geometric complexity across different planes, we employ an adaptive matching criterion. A recovered plane is regarded as a true positive if both its angular deviation and spatial distance to a ground-truth plane fall within twice the maximum fitting residual of the ground-truth plane's supporting points. A one-to-one matching constraint is enforced to prevent duplicate assignments. Performance is evaluated using Precision (TP/$|\mathcal{R}|$), Recall (TP/$|\mathcal{G}|$), and the F_1-score:

$$\mathrm{F}_1 = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}. \tag{5}$$

As shown in Figure 12, when the angular threshold is set too small, the F_1-score of plane recovery decreases significantly. This is primarily because an excessive number of singular edges are extracted, leading to the reconstruction of many erroneous planes. Conversely, overly large angular thresholds suppress the detection of genuine missing planes, resulting in reduced recall. Empirically, the best performance is achieved when the angular threshold lies between 8° and 12°, and the distance threshold is set to approximately 4 to 6 times the average plane fitting residual, yielding consistently higher F_1-scores.

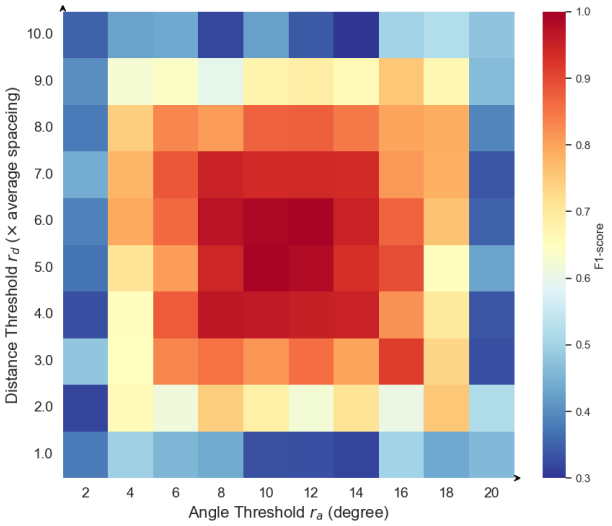

Figure 12. Impact of the angular threshold and the distance threshold.

7.3. Weighting Parameter for Surface Extraction

This parameter primarily governs the trade-off between model compactness and geometric fidelity. When assigned an excessively large value (e.g., greater than 0.9), the reconstruction tends to suffer from noticeable structural loss and increased geometric errors. Conversely, overly small values (e.g., below 0.2) lead to redundant fine-scale details and the emergence of zigzag artifacts on the surface. As illustrated in Figure 13 and Table 3, selecting the parameter within the range of 0.4 to 0.7 achieves a favorable balance between surface simplicity and reconstruction accuracy.

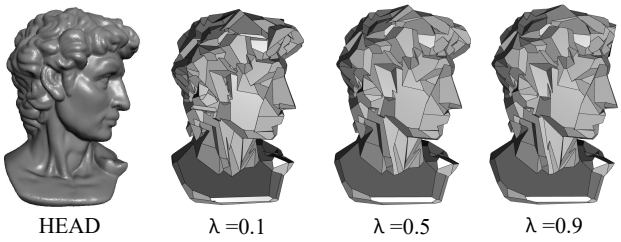

Figure 13. Impact of the parameter λ. When λ is too small, more erroneous and redundant surfaces are preserved (e.g., $\lambda = 0.1$). In contrast, too large values of λ lead to missing surface patches (e.g., $\lambda = 0.9$). Table 3 presents more detailed values.

λ	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
#f	838	821	805	792	781	773	720	695	631
e	0.412	0.396	0.382	0.371	0.372	0.375	0.379	0.381	0.567

Table 3. Effect of λ. #f is the number of output facets, and e is the mean Hausdorff error (MHE) from input points to the output model.

ID	*Components*		*Correctness*		*Conciseness*			*Efficiency*
	MP-Recovery	H-Partitions	MHE ↓	RMSE ↓	$P^{Avg.}$ ↓	$F^{Avg.}$ ↓	$RH^{Avg.}$ ↓	T(s) ↓
1	✗	✗	1.955	0.041	279	385	0.0060	**231**
2	✓	✗	1.073	0.031	329	523	0.0042	320
3	✗	✓	1.930	0.039	**251**	**359**	0.0053	309
4(ours)	✓	✓	**1.066**	**0.025**	315	501	**0.0036**	327

Table 4. Ablation study of MP-Recovery and H-Partitions on dataset *Arch-100*. **MP-Recovery**: Missing Planes Recovery; **H-Partitions**: Hierarchical Partitions. Correctness is quantified using Mean Hausdorff Error (*MHE*) and Root Mean Squared Error (*RMSE*), reflecting how well the reconstructed polygonal planes align with the input points. Conciseness is evaluated based on the number of vertices ($P^{Avg.}$) and facets ($F^{Avg.}$) in the reconstructed mesh, as well as a composite metric ($RH^{Avg.}$) that combines the Hausdorff distance and a simplification ratio. Computational efficiency is assessed via runtime (*T*). The best results are highlighted in **bold**.

8. Ablation study

To validate the effectiveness of the proposed Missing Planes Recovery and Hierarchical Partitions, we conduct ablation studies on the dataset *Arch-100*, as reported in Table 4. As shown in the results, removing Missing Planes Recovery leads to a significant degradation in accuracy. This is primarily attributed to the fact that certain locally missing details cannot be accurately reconstructed. On the other hand, the absence of Hierarchical Partitions results in increased complexity in spatial partitioning and reduced model conciseness.

9. Flexibility

To further demonstrate the flexibility and generalization capability of our method, we conduct additional experiments on a set of free-form objects. It is worth noting that the point clouds of these objects are obtained using different acquisition techniques. Specifically, models such as Horse, Ignatius, and Capron are reconstructed from point clouds generated by multi-view stereo (MVS), while Bunny and Hand are acquired using laser scanning.

Despite the diversity in object shapes and data sources, our approach consistently produces accurate, high-quality re-

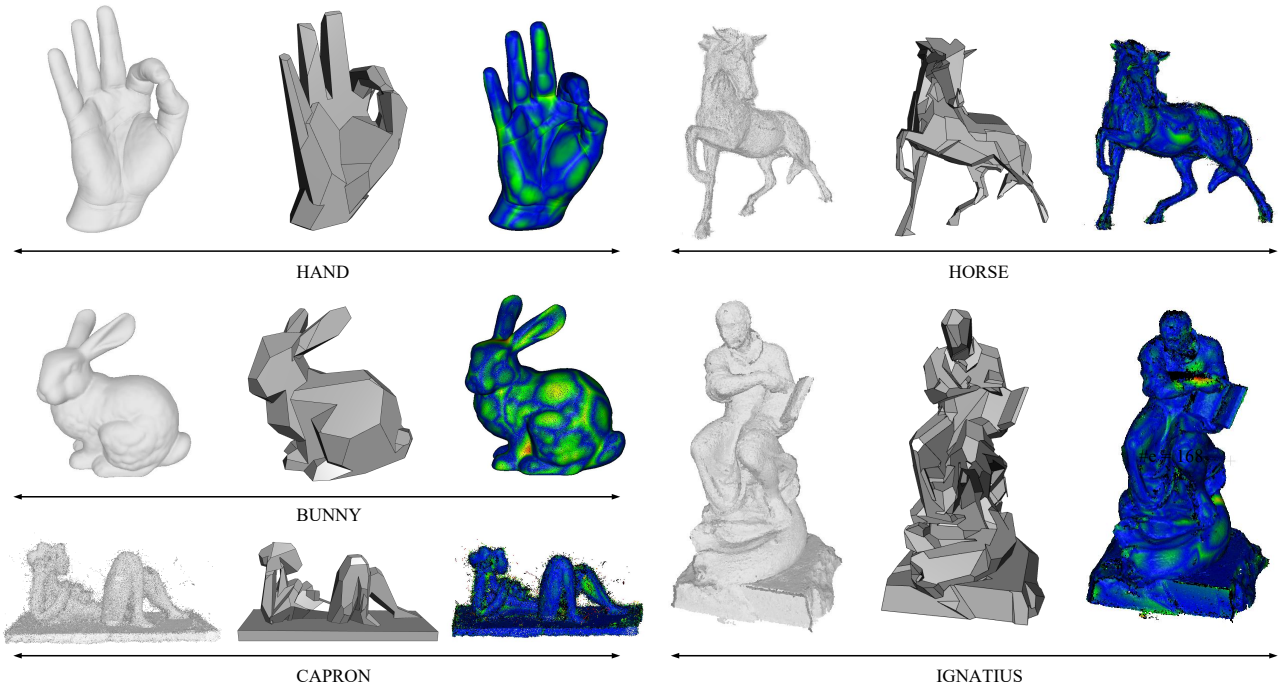

Figure 14. Reconstruction results on free-form objects. For each instance, we present the input point cloud with aligned normals (left), the generated watertight polygonal mesh (center), and a heatmap visualizing the RMSE (root-mean-square error) between the reconstructed model and the input point cloud (right).

		BUNNY	HAND	CAPRON	HORSE	IGNATIUS
	Type	Laser	Laser	MVS	MVS	MVS
	#i	146K	369K	168K	788K	1.4M
	#s	101	75	152	275	298
Ours	e	0.463	**0.411**	**0.259**	**0.237**	**0.201**
	#f	121	95	**151**	365	**430**
KSR	e	**0.454**	0.423	0.282	0.249	0.212
	#f	**111**	**92**	156	**351**	443

Table 5. Reconstruction results on free-form surface objects. #i is the number of input points, #s is the number of detected planes, #f is the number of facets of the reconstruction model, and e is the mean Hausdorff error (MHE) from input points to the output model. The best results are highlighted in **bold**.

constructions, yielding compact spatial partitions and concise mesh representations, as shown in Figure 14 and Table 5. It should be noted, however, that for these free-form objects, the missing plane recovery module rarely detects valid missing planes in practice, as the underlying planar structure assumption is largely violated.

10. Limitations

Because our missing plane fitting relies on the normal information of boundary segments, if the normal difference is too large, it cannot be recovered, for example, we cannot restore the missing top surface of a quadrangular frustum. Moreover, as the reconstruction framework is designed to produce watertight models, it is not suitable for completely open environments like a tennis court.