
CAGE: Continuity-Aware edGE Network Unlocks Robust Floorplan Reconstruction

Yiyi Liu¹ Chunyang Liu¹ Bohan Wang¹ Weiqin Jiao²

Bojian Wu³ Lubin Fan³ Yuwei Chen⁴ Fashuai Li^{5,*} Biao Xiong^{1,*}

¹Wuhan University of Technology ²University of Twente ³Independent Researcher

⁴Hangzhou Institute for Advanced Study, University of Chinese Academy of Sciences

⁵The Advanced Laser Technology Laboratory of Anhui Province

*Corresponding Authors: lifashuai@gmail.com, b.xiong@whut.edu.cn

Abstract

We present **CAGE** (*Continuity-Aware edGE*) network, a robust framework for reconstructing vector floorplans directly from point-cloud density maps. Traditional corner-based polygon representations are highly sensitive to noise and incomplete observations, often resulting in fragmented or implausible layouts. Recent line grouping methods leverage structural cues to improve robustness but still struggle to recover fine geometric details. To address these limitations, we propose a *native* edge-centric formulation, modeling each wall segment as a directed, geometrically continuous edge. This representation enables inference of coherent floorplan structures, ensuring watertight, topologically valid room boundaries while improving robustness and reducing artifacts. Towards this design, we develop a dual-query transformer decoder that integrates perturbed and latent queries within a denoising framework, which not only stabilizes optimization but also accelerates convergence. Extensive experiments on Structured3D and SceneCAD show that **CAGE** achieves state-of-the-art performance, with F1 scores of 99.1% (rooms), 91.7% (corners), and 89.3% (angles). The method also demonstrates strong cross-dataset generalization, underscoring the efficacy of our architectural innovations. Code and pretrained models are available on our project page: <https://github.com/ee-Liu/CAGE.git>.

1 Introduction

Reconstructing indoor scenes into compact, editable vector floorplans is a longstanding goal in computer vision and robotics [1; 2]. A vector floorplan is a structured 2D representation of interior geometry, composed of lines or polygons that delineate walls, room boundaries, and architectural elements. Unlike raster maps, vector floorplans are resolution-independent, support precise geometric reasoning, and integrate seamlessly with CAD tools and building information models (BIM). Their joint encoding of spatial topology and geometry makes them ideal for downstream applications such as building lifecycle management, AR/VR simulation, and autonomous navigation [3; 4; 5; 6].

Modern floorplan reconstruction uses diverse inputs including RGB images [7], panoramic views [8], CAD drawings [9], and 3D point clouds [10]. Among these, projecting point clouds to 2D density maps provides an optimal balance of geometric accuracy and computational efficiency [11; 12], making it popular in learning-based approaches. However, real-world scans often suffer from severe occlusions, clutter, and incompleteness that obscure critical structural elements. Additionally, the discretization process in density map creation blurs corner features and wall segments. These challenges significantly complicate accurate vector floorplan recovery. The resulting incomplete or noisy representations demand robust algorithms capable of handling such imperfect data.

Notably, a critical factor affecting reconstruction quality lies in the choice of floorplan representation. Recent work has explored a range of methods. For example, HEAT [13] detects discrete corners and

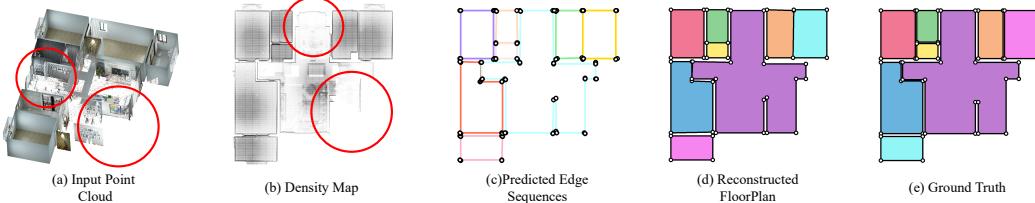


Figure 1: Floorplan reconstruction using our **CAGE** network. Given an input point cloud (a), we project it into a 2D density map (b), predict two-level edge sequences (c), and reconstruct a vector floorplan by intersecting the predicted edges (d). Note that, our edge-based formulation enables the recovery of regular, topologically valid polygons even in severely occluded regions (highlighted with red circles). See Figure 6 for comparisons with corner-based and other SOTA methods.

connects them into edges and polygons, yet it often suffers from gaps caused by missed detections. RoomFormer [12] models floorplans as sequences of corners using a two-level query transformer, yet a single missing corner can distort the entire layout. As a result, corner-based methods are highly sensitive to noise. SLIBO-Net [14] introduces a slicing-box representation but relies heavily on the Manhattan-world assumption. FRI-Net [15] reconstructs rooms by compositional line grouping, its implicit neural representations often lead to over-smoothing of fine structures. It is clear that current forms still struggle to capture global structure while keeping local precision.

To address these challenges, we propose **CAGE** (*Continuity-Aware edGE*) network, a robust framework for reconstructing vector floorplans from density maps. Our **CAGE** adopts an edge-centric formulation, modeling each wall as a directed, geometrically continuous edge. This design improves robustness to incomplete or noisy data by removing reliance on precise corner localization, while enabling global reasoning. To complement this representation, we introduce a dual-query transformer decoder that integrates perturbed and latent edge queries within a denoising framework, enhancing training stability and accelerating convergence. As decoding progresses, edge predictions are iteratively refined, yielding clean and watertight polygons. Evaluations on Structured3D [16] and SceneCAD [17] demonstrate that **CAGE** achieves state-of-the-art performance and generalizes well across datasets. Our main contributions are as follows:

- **Continuity-Aware Edge Representation:** We introduce an edge-based polygon formulation that enhances robustness to incomplete and noisy data, while preserving directional and structural consistency.
- **Dual-Query Transformer Decoder:** Our approach leverages perturbed and latent queries within a denoising framework to stabilize training, refine predictions, and accelerate convergence.
- **Strong Performance and Generalization:** We achieve state-of-the-art performance on multiple benchmarks and generalize effectively across datasets, validating the effectiveness of our continuity-aware edge-based formulation.

2 Related Work

Approaches to floorplan reconstruction can be broadly categorized into three groups: traditional methods based on classical geometric or machine learning techniques, stage-wise learning-based pipelines that combine hand-crafted and learned components, and fully end-to-end learning methods.

2.1 Traditional Methods

Early work in floor- and room-layout recovery relied on low-level vision cues or geometric heuristics. Some methods extract planes from point clouds and optimize line placements to assemble floorplans [18; 19], while others infer structure from panoramic images [20; 8] or CAD drawings [9; 21; 22]. Techniques such as RANSAC-based plane fitting and piecewise merging have been used to generate compact surface meshes that are later flattened into floorplans [23]. Graph-based formulations often cast the reconstruction as shortest-path or global optimization problems, producing watertight layouts but requiring hand-designed potentials [24; 25]. More recent pipelines focus on handling cluttered LiDAR scans via global optimization over curved surfaces [26] or decomposing scenes into walls and objects for lightweight BIM generation [27]. From a single panorama, Pano2CAD estimates Manhattan room geometry and object pose [28], while successors like PanoFormer [29]

support curved wall recovery via tessellated spherical surfaces. Recent work also explores LoD4 building modeling by fusing interior and exterior cues from SfM images [30]. Traditional methods require small amounts of training data while heavily relying on the design of handcrafted features.

2.2 Stage-wise Learning-based Methods

Several methods adopt a hybrid pipeline where learning components are combined with post-processing optimization. FloorNet [31] detects corners in a top-view raster and assembles them via integer programming, while HEAT [13] replaces the solver with a transformer-based corner-pair classifier. Floor-SP [11] and MonteFloor [32] segment rooms using Mask-RCNN [33], refine initial contours through shortest path or Monte Carlo Tree Search [34; 35], and encourage geometric consistency by sharing corners and walls. PolyGraph [36] generates wall points using a cross-guided neural network, forms initial triangles, and applies post-processing to refine polygonal layouts. FloorUSG [19] integrates 2D plane instances with 3D geometry to lift RGB features into structured floorplans. ArrangementNet [10] detects walls and partitions space using wall lines, then leverages an extended GCN to model collinearity and coplanarity among surface patches, improving segmentation in complex scenes. SLIBO-Net [14] introduces a slicing box representation with geometric regularization and post-processing for capturing fine local details. While these stage-wise systems achieve strong performance, they are often sensitive to missing primitives and rely on extensive post-optimization.

2.3 End-to-End Learning-based Methods

Recent advances favor end-to-end pipelines that directly predict vector floorplan using transformer-based decoders. RoomFormer [12] introduces a transformer architecture with hierarchical queries to jointly predict multiple corner sequences. Although effective, it suffers from unordered predictions due to random query initialization and incomplete polygons caused by missed corners. PolyRoom [37] enhances this framework by segmenting rooms with Mask-RCNN [33] and initializing RoomFormer-style queries from detected contours. PolyDiffuse [38] formulates floorplan generation as a conditional task and introduces a guided set diffusion model for room representation optimization. These corner-based methods treat floorplan reconstruction as discrete corner regression, often neglecting global shape coherence. FRI-Net [15] addresses this by learning room-wise latent codes and decoding them into lines, which are grouped into polygons using a BSP-Net-inspired [39] grouping strategy. Recent methods have progressed from corner-based to line-grouping approaches, highlighting the importance of structural information. However, they still fail to balance robustness with geometric precision. We address this with edge-centric representation that fundamentally reformulates the problem, enabling accurate, topologically valid reconstructions in an end-to-end trainable framework.

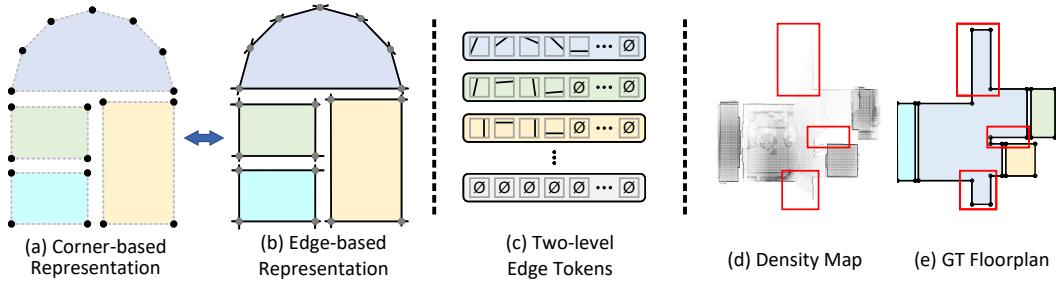


Figure 2: Edge-based Floorplan Representation. (a) Corner-based: polygons defined by sequential vertices; (b) Edge-based: walls represented as directed edges with geometric continuity; (c) Tokenization: polygons represented as sequences of edge tokens at two levels; (d) Density map generated from point cloud; (e) Ground-truth floorplan. While edge and corner representations are mathematically dual, the edge-based formulation offers greater robustness to noise and occlusion (highlighted in red).

3 Edge-based Floorplan Representation

We propose an explicit edge-based representation for modeling floorplan, addressing the key limitations of prior methods, see Figure 2(a)(b). We represent each room polygon as an ordered sequence of directed edges, where each edge is defined by a pair of endpoints in normalized 2D space, as illustrated in Figure 2(b). Let a floorplan contain at most M rooms, and each room contain at most N

edges. We represent the m -th room as:

$$R_m = \{e_m^n\}_{n=1}^{N_m}, \quad \text{where } e_m^n = (\mathbf{p}_{m1}^n, \mathbf{p}_{m2}^n) \quad (1)$$

Each edge $e_m^n \in \mathbb{R}^4$ connects two points $\mathbf{p}_{m1}^n, \mathbf{p}_{m2}^n \in [0, 1]^2$. The edge order encodes the polygon's directional traversal. To support batched training and variable-length polygons, we augment each edge with a binary validity label:

$$t_m^n = (\mathbf{p}_{m1}^n, \mathbf{p}_{m2}^n, c_m^n), \quad c_m^n \in \{0, 1\} \quad (2)$$

Here, $c_m^n = 0$ denotes an invalid (padded) edge, and each t_m^n serves as an edge token, as illustrated in Figure 2(c). If all edges in a polygon are invalid, the entire room is treated as padding. While edge and corner representations are dual, edges are empirically more stable in sparse or noisy conditions, since they encode directionality explicitly, promoting angular regularity and structural coherence. Unlike corner-based methods that require precise vertex localization, edge-based design allows endpoints to lie along wall segments. The flexibility simplifies spatial queries and enables the model to infer global structure even from partial observation.

We aggregate edge tokens into a fixed-length set:

$$T_m = \{t_m^n\}_{n=1}^N, \quad \text{where } \max N_m \leq N \quad (3)$$

and define the full floorplan as:

$$\mathcal{F} = \{T_m\}_{m=1}^M \quad (4)$$

This edge-based representation models room layouts as fixed-length sequences of spatially grounded edge tokens, which align well with transformer architectures, facilitate spatial attention over relevant image regions, and support stable training through bipartite matching.

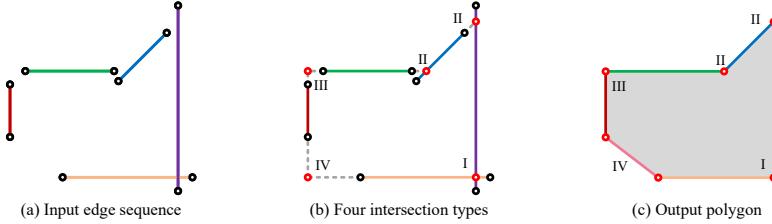


Figure 3: Illustration of edge-to-polygon conversion. (a) Input edge sequence; (b) Four types of edge intersections based on geometric proximity; (c) Final reconstructed polygon.

To convert edge sequences into closed polygons, we resolve edge intersections into vertices. As illustrated in Figure 3, we identify four types of pairwise edge intersections. Types I–III involve intersection points that lie on or near the edges and are retained as valid polygon vertices. In Type IV, where the intersection is far from any endpoints, we instead connect the nearest edge endpoints to preserve continuity. As all edges are ordered and directed, this conversion remains deterministic and robust even under noisy predictions.

4 Method

4.1 Overview

Figure 4 illustrates the overall architecture of our proposed **CAGE** framework for floorplan reconstruction from 3D point clouds. The input point cloud is first projected onto the XY-plane to generate a 2D density map, which is processed by a convolutional image backbone to extract multi-scale visual features. The output feature maps are flattened and augmented with positional encodings to form a unified token sequence, which is passed to a transformer encoder. We employ multi-scale deformable attention [40] in the encoder to efficiently aggregate both local and global spatial context across all feature levels.

The encoded features are processed by our novel transformer decoder composed of stacked attention layers. The decoder operates on a set of learnable edge queries, which are refined across layers through self-/cross-attention mechanisms. Each query predicts a directed edge denoted by two endpoints and is assigned a binary label indicating whether it constitutes a valid segment, as described in Sec. 3. To improve the robustness and training stability, we incorporate a dual-query design that includes latent queries for final predictions and perturbed queries for denoising supervision. The following subsections will illustrate our novel dual-query decoding mechanism and loss formulation.

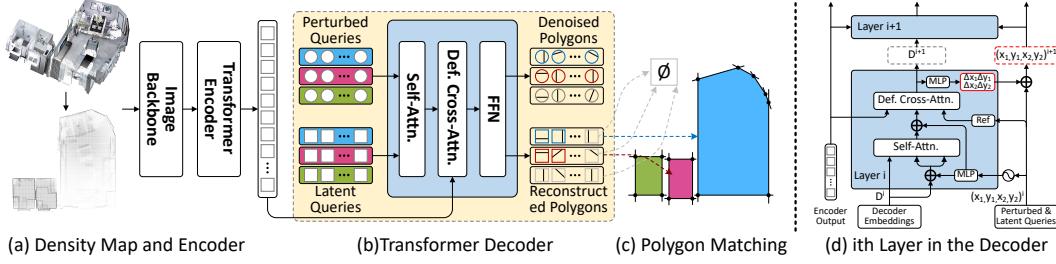


Figure 4: Architecture of the **CAGE** network. (a) The input point cloud is projected into a density map and processed by an image backbone and transformer encoder. (b) The transformer decoder receives two types of queries (perturbed and latent) and predicts edges defined by two endpoints, which may not correspond to polygon vertices. (c) A feed-forward network assigns class labels, with polygon matching for supervision. (d) Detailed architecture of decoder, showing the progressive refinement of edge queries with the incorporation of the novel designs of perturbation and denoising.

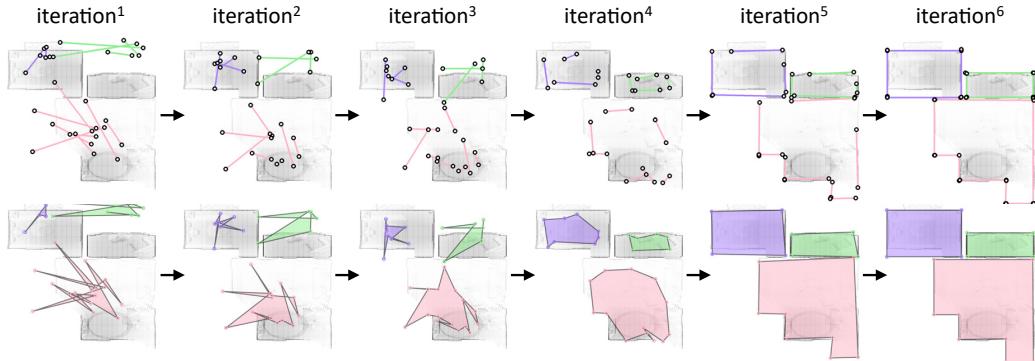


Figure 5: Polygon evolution during decoding. Top: edge query predictions refined over six decoder layers. Bottom: corresponding polygon reconstruction becomes more accurate and complete.

4.2 Iterative Polygon Refinement by Dual Query

To enable precise and stable polygon prediction, we design a dual-query transformer decoder that refines edge representations over multiple decoding layers. As shown in Figure 4(d), the decoder jointly operates on two sets of edge queries: perturbed queries used for denoising-based supervision, and latent queries responsible for final polygon prediction. The denoising strategy encourages the model to recover clean edge structures from deliberately corrupted inputs, improving its robustness to noise and accelerating convergence. This dual-query design strengthens learning stability, enhances generalization, and maintains full end-to-end differentiability.

As described in Sec. 3, we represent each room as a sequence of edges and flatten all edges into query tensor as $Q \in \mathbb{R}^{M \times N \times 2}$, where M is the maximum number of polygons and N is the maximum number of edges per polygon. Each query predicts an edge defined by two endpoints. The decoder layers refine these queries iteratively, as illustrated in Figure 5, where edge geometry and corresponding polygons become increasingly accurate across decoding steps.

Each layer receives positional queries derived from polygon coordinates via sinusoidal encoding, along with content queries attending to multi-scale encoder features. Self-attention allows intra-polygon edge refinement and inter-polygon context exchange. Cross-attention modules associate each edge query with spatially grounded encoder features, guided by its predicted endpoint coordinates. This supports effective reasoning over both local structure and global context. A binary classifier identifies edge validity, allowing the model to handle variable polygon structures.

Following Deformable DETR [40], we refine edge coordinates through iterative offset prediction. Each decoder layer outputs coordinate offsets that adjust the endpoints of every edge query, improving geometric precision across layers. Latent queries are randomly initialized and optimized for prediction, while perturbed queries are derived by adding controlled noise to ground-truth edge coordinates. These two query types are processed jointly in the decoder.

To support denoising supervision, we adopt a training objective inspired by DN-DETR [41]. The decoder receives perturbed queries generated by adding controlled noise to ground-truth edges and optionally flipping their class labels. Let the perturbed query set be $\mathbf{q} = \{q_0, \dots, q_{MN-1}\}$, and the latent query set be $\mathbf{Q} = \{Q_0, \dots, Q_{MN-1}\}$. The decoder operates on both sets under an attention mask \mathbf{A} that prevents leakage between perturbed instances:

$$\mathbf{o} = D(\mathbf{q}, \mathbf{Q}, \mathbf{F} \mid \mathbf{A}) \quad (5)$$

where \mathbf{F} denotes encoder features.

Each perturbed edge query is generated by applying controlled noise to a ground-truth edge during training. Specifically, for each edge, its two endpoints are randomly displaced to simulate input uncertainty. The displacements are constrained relative to the spatial extent of the density map: the horizontal shift Δx and vertical shift Δy are bounded by $|\Delta x| < \lambda w/2$ and $|\Delta y| < \lambda h/2$, where w and h denote the width and height of the input density image, respectively. If a perturbed endpoint falls outside the image boundary, it is clamped to remain within the valid range. The scalar λ is a hyperparameter that determines the magnitude of perturbation. Additionally, to introduce label noise, the binary class label of each edge is randomly flipped with probability γ . This denoising strategy encourages the decoder to recover clean edge structures from noisy inputs. During inference, only the latent queries are used to generate final polygons.

4.3 Loss Functions

To train the **CAGE** network in an end-to-end manner, we formulate a loss that supervises both the structure and geometry of predicted polygons. The decoder outputs a fixed number M of polygons, each consisting of up to N directed edges. Since the number and order of edges vary across ground-truth annotations, we adopt a two-level matching strategy that aligns predictions with targets.

Polygon Matching. Let the predicted edge set be $\hat{S} = \{\hat{E}_m = (\hat{e}_m^1, \dots, \hat{e}_m^N)\}_{m=1}^M$, where each edge $\hat{e}_m^n = (\hat{c}_m^n, \hat{p}_{m1}^n, \hat{p}_{m2}^n)$ includes a binary confidence score and two endpoint coordinates. Ground-truth polygons are denoted as $S = \{E_m\}_{m=1}^{M_{\text{gt}}}$, each padded to length N and further extended with mock polygons for matching. We compute a bipartite matching between predicted and ground-truth polygons using the Hungarian algorithm, minimizing the following assignment cost:

$$\hat{\sigma} = \arg \min_{\sigma} \sum_{m=1}^M \mathcal{D}(E_m, \hat{E}_{\sigma(m)}) \quad (6)$$

where the matching cost \mathcal{D} includes both classification error and edge regression:

$$\mathcal{D}(E_m, \hat{E}_{\sigma(m)}) = \mathbb{1}_{\{m \leq M_{\text{gt}}\}} \left[\lambda_{\text{cls}} \sum_{n=1}^N \left| c_m^n - \hat{c}_{\sigma(m)}^n \right| + \sum_{n=1}^N \sum_{k=1}^2 \left\| p_{mk}^n - \hat{p}_{\sigma(mk)}^n \right\|_1 \right] \quad (7)$$

To account for the cyclic nature of polygons, we evaluate distances over all valid rotations of the ground-truth sequence and choose the minimal one.

Loss Components. After matching, we supervise the predicted polygons using three terms: classification, edge regression, and rasterization. For latent queries, losses are applied only to matched predictions. For *perturbed queries*, losses are directly applied without the matching step, as their correspondence to ground-truth edges is already known.

The classification loss uses binary cross-entropy:

$$\mathcal{L}_{\text{cls}}^m = -\frac{1}{N} \sum_{n=1}^N \left[c_n^m \log(\hat{c}_n^{\hat{\sigma}(m)}) + (1 - c_n^m) \log(1 - \hat{c}_n^{\hat{\sigma}(m)}) \right] \quad (8)$$

The edge regression loss is the ℓ_1 distance between predicted and ground-truth endpoints:

$$\mathcal{L}_{\text{edge}}^m = \frac{1}{N_m} \mathbb{1}_{\{m \leq M_{\text{gt}}\}} \sum_{n=1}^{N_m} \sum_{k=1}^2 \left\| p_{mk}^n - \hat{p}_{\sigma(mk)}^n \right\|_1 \quad (9)$$

The rasterization loss measures the overlap between polygon masks using the Dice loss [42]:

$$\mathcal{L}_{\text{ras}}^m = \mathbb{1}_{\{m \leq M_{\text{gt}}\}} \cdot \text{Dice} \left(R(P_m), R(\hat{P}_{\hat{\sigma}(m)}) \right) \quad (10)$$

where $R(\cdot)$ denotes a differentiable polygon rasterizer [43]

For perturbed queries, we apply denoising losses without matching. These include binary cross-entropy for classification and ℓ_1 regression for edge localization, denoted as $\mathcal{L}_{\text{cls_DN}}$ and $\mathcal{L}_{\text{edge_DN}}$.

The total training loss is the weighted sum of all components:

$$\mathcal{L} = \sum_{m=1}^M (\lambda_{\text{cls}} \mathcal{L}_{\text{cls}}^m + \lambda_{\text{edge}} \mathcal{L}_{\text{edge}}^m + \lambda_{\text{ras}} \mathcal{L}_{\text{ras}}^m + \lambda_{\text{cls_DN}} \mathcal{L}_{\text{cls_DN}}^m + \lambda_{\text{edge_DN}} \mathcal{L}_{\text{edge_DN}}^m) \quad (11)$$

5 Experiments

5.1 Experiments Setting

Dataset and Evaluation Metrics. We evaluate our method on two large-scale indoor datasets, including Structured3D [16] and SceneCAD [17]. Structured3D is a photo-realistic synthetic dataset comprising 3,500 houses with diverse Manhattan and non-Manhattan layouts. SceneCAD provides 3D room layout annotations for real-world RGB-D scans from ScanNet [44], with each sample containing a single room. Following previous work [13; 12], we split Structured3D into 3,000 training, 250 validation, and 250 test samples. For SceneCAD, we use the provided splits of 828 training and 127 validation samples. Following [32; 13], we process registered multi-view RGB-D panoramas into point clouds, and project them vertically into 256×256 pixel density images. Each pixel value is normalized to $[0, 1]$ by dividing the number of projected points by the maximum count per image. In line with prior work [13; 12; 36], we evaluate performance using Precision, Recall, and F1 scores at the Room, Corner, and Angle levels.

Baselines. We compare our method with eight state-of-the-art approaches: Floor-SP [11], HEAT [13], RoomFormer [12], SLIBO-Net [14], FRI-Net [15], PolyRoom [37], PolyGraph [36], and PolyDiffuse [38]. Floor-SP [11] uses Mask-RCNN for room segmentation followed by classical optimization for vectorization. HEAT [13] detects corners and infers connectivity with a neural network. RoomFormer [12] employs a transformer to jointly predict multiple corner sequences. SLIBO-Net [14] generates slicing boxes and room centroids via transformer decoding, refined through post-processing. FRI-Net [15] encodes room-wise latent features, decodes them into lines, and assembles polygons using a BSP-Net-inspired [39] grouping strategy. PolyRoom [37] combines Mask-RCNN-based segmentation with RoomFormer-style reconstruction initialized from detected contours. PolyGraph [36] predicts wall points and triangulations and then refines results by postprocessing. PolyDiffuse [38] formulates reconstruction as a conditional generation task, using a diffusion model to refine floorplans from initial proposals iteratively.

Implementation Details. Our model is implemented in PyTorch and trained on a single NVIDIA RTX 4090 GPU. We use Swin Transformer V2 [45] as the default image backbone, while ResNet-50 [46] and Swin Transformer V1 [47] are used in ablation studies. The transformer architecture follows a standard 6-layer encoder-decoder configuration. We apply denoising training with dual-query supervision and optimize using Adam [48]. Further model configurations, training schedules, and hyperparameters are provided in supplementary material A.

5.2 Comparisons to State-of-the-Art Methods

Table 1: Quantitative comparison on the Structured3D [16] test set. The running time is averaged over the whole test set. PD denotes PolyDiffuse [38] post-optimization. *: Params are calculated from the officially released models.

Method	Venue	Backbone	Params(M)*	t (s)	Room			Corner			Angle		
					Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
HEAT [13]	CVPR22	ResNet-50	48.9	0.11	96.9	94.0	95.4	81.7	83.2	82.5	77.6	79.0	78.3
RoomFormer [12]	CVPR23	ResNet-50	40.6	0.01	97.9	96.7	97.3	89.2	85.3	87.2	83.0	79.6	81.3
SLIBO-Net [14]	NeurIPS23	ResNet-50	—	0.17	99.1	97.8	98.4	88.9	82.1	85.4	87.8	81.2	84.4
PolyRoom [37]	ECCV24	ResNet-50	42.2	0.02	98.9	97.7	98.3	94.6	86.1	90.2	89.3	81.4	85.2
FRI-Net [15]	ECCV24	ResNet-50	58.6	0.09	99.5	98.7	99.1	90.8	84.9	87.8	89.6	84.3	86.9
PolyGraph [36]	TVC25	ResNet-50	—	0.04	95.7	97.9	96.7	92.4	82.2	88.3	89.2	79.4	85.4
CAGE(Our)	—	SwinV2-L	211.3	0.01	99.6	98.7	99.1	95.0	88.6	91.7	92.5	86.4	89.3
RoomFormer+PD [38]	NeurIPS23	ResNet-50	—	—	98.7	98.1	98.4	92.8	89.3	91.0	90.8	87.4	89.1
FRI-Net+PD [15]	ECCV24	ResNet-50	—	—	99.6	98.6	99.1	94.2	88.2	91.1	91.9	86.7	89.2
Ours+PD	—	SwinV2-L	—	—	99.7	98.7	99.1	95.1	89.9	91.7	91.6	87.5	89.2

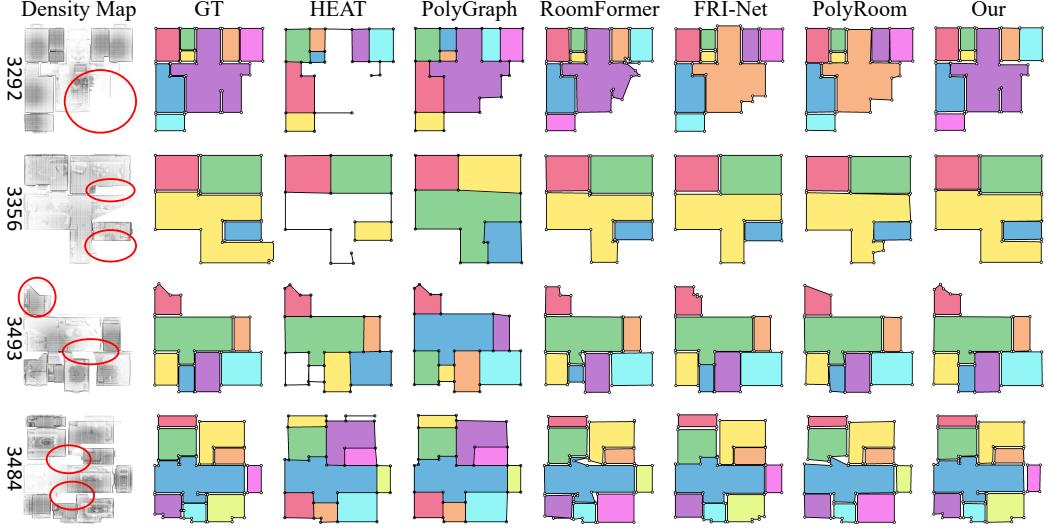


Figure 6: Qualitative comparisons on Structured3D [16]. Scene IDs are shown on the left. Red circles highlight challenging regions.

Quantitative Evaluation. Quantitative comparisons against state-of-the-art floorplan reconstruction methods on the Structured3D [16] dataset are presented in Table 1. We compare our method with HEAT [13], SLIBO-Net [14], PolyRoom [37], FRI-Net [15], and PolyGraph [36]. For completeness, we also include post-optimization results using PolyDiffuse [38]. Without any post-processing, our method achieves the best performance across all geometric levels, reaching 99.1% Room F1, 91.7% Corner F1, and 89.3% Angle F1. Although FRI-Net attains comparable performance in Room Recall and Room F1, **CAGE** surpasses it by a notable margin of +3.9 points in Corner F1 and +2.4 points in Angle F1, demonstrating superior accuracy in fine-grained corner localization and angular estimation. After applying PolyDiffuse for post-optimization, our method further consolidates its leading position across all three evaluation levels, outperforming both RoomFormer and FRI-Net. Interestingly, the post-optimization with PolyDiffuse brings only marginal improvements to our model, indicating that **CAGE** already captures the scene structure effectively without reliance on heavy post-processing. In contrast, PolyDiffuse, as a diffusion-based model, requires substantial additional training time. Furthermore, despite not explicitly optimizing for corner predictions, our method maintains competitive performance in Corner Recall. Regarding efficiency, **CAGE** achieves the fastest inference speed, along with RoomFormer (both 0.01 second per image), benefiting from its lightweight architecture without the need for additional refinement stages. Overall, these results demonstrate that **CAGE** strikes an excellent balance between reconstruction accuracy and computational efficiency.

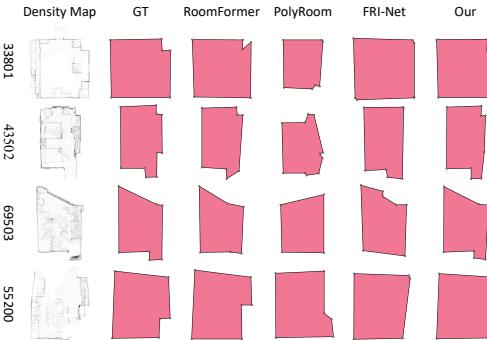


Figure 7: Qualitative evaluations on SceneCAD [17].

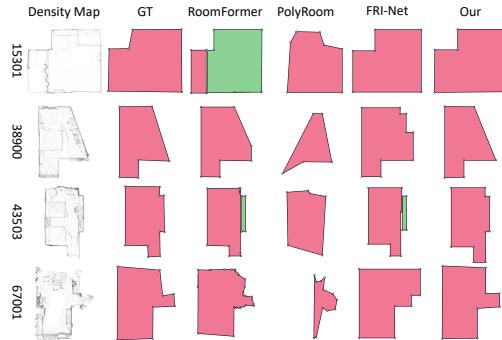


Figure 8: Cross-dataset evaluation (training on Structured3D [16] and testing on SceneCAD [17]).

On the SceneCAD [17] dataset, we further evaluate our method against Floor-SP [11], HEAT [13], RoomFormer [12], FRI-Net [15], and PolyRoom [37]. Given that SceneCAD primarily consists of single-room layouts, we assess reconstruction quality using IoU for room shape. As shown in Table 2, **CAGE** consistently achieves either the best or second-best performance across nearly all evaluation metrics. Specifically, our method attains the highest scores in Room IoU, Corner Recall, Angle Recall, and Angle F1, while securing second-best results in other metrics. In comparison, PolyRoom achieves the highest Corner Precision and Corner F1 but performs slightly lower on remaining metrics. Overall, **CAGE** demonstrates strong and balanced performance across all aspects of room layout reconstruction on the SceneCAD dataset.

Qualitative Evaluation. Visual comparisons on Structured3D [16] are shown in Figure 6, with red circles highlighting challenging regions for reconstruction. Most state-of-the-art methods fail to recover severely occluded or blocked areas, while our method successfully infers structures using weak wall cues (e.g., scene 3292). HEAT [13] and PolyGraph [36] reconstruct wall centerlines, whereas other methods produce double-wall representations, with each polygon denoting room interior boundaries. HEAT follows a two-stage process—detecting corners and inferring connectivity—which often leads to missing corners and incorrect edge associations (e.g., scene 3493), resulting in open polygons. PolyRoom [37] generates simple, closed polygons but struggles in occluded regions (e.g., 3292, 3356). RoomFormer [12] predicts polygonal layouts via sequences of corners, ensuring closure but often producing irregular shapes due to weak constraints on edge relationships (e.g., 3292, 3493, 3484). While FRI-Net [15] and PolyRoom demonstrate better regularity and partial robustness to occlusion, they still fail under heavy blockage (e.g., 3292, 3356). In contrast, our method explicitly models global structure through edge-aware representation and denoising-based training, leading to more regular room geometries and robust reconstruction in occluded scenarios. Figure 7 further confirms this trend, with our model consistently producing more plausible results compared to corner-based optimization methods.

Cross-Dataset Generalization. In cross-dataset evaluation (training on Structured3D [16] and testing on SceneCAD [17]), *PolyRoom* and **CAGE** achieve the top results across Room IoU and fine-level corner/angle metrics (Table 3). PolyRoom’s segmentation-based pipeline aligns well with SceneCAD’s single-room layouts, yielding strong room-level performance, but its multi-stage design struggles with the complex multi-room structures in Structured3D (Table 1). In contrast, **CAGE**’s end-to-end trainable, edge-based modeling consistently maintains high accuracy (85.6% Room IoU, 87.5% Corner Recall, 70.6% Angle Recall, 61.6% Angle F1) without post-processing. These results validate that combining edge-based representations with global shape modeling and denoising strategies enables **CAGE** to achieve robust and balanced floorplan reconstruction under domain shifts and varying scene complexities.

Table 2: Quantitative comparison on the SceneCAD validation set [17]. Results for prior methods are reported from [12; 15; 38].

Method	Room			Corner			Angle		
	IOU	Prec.	Rec.	F1	Prec.	Rec.	F1		
Floor-SP [11]	91.6	89.4	85.8	87.6	74.3	71.9	73.1		
HEAT [13]	84.9	87.8	79.1	83.2	73.2	67.8	70.4		
RoomFormer [12]	91.7	92.5	85.3	88.8	78.0	73.7	75.8		
FRI-Net [15]	92.3	92.8	85.9	89.2	78.3	73.6	75.9		
PolyRoom [37]	92.8	96.8	86.1	91.2	81.7	74.5	78.0		
CAGE(Ours)	93.7	93.7	87.7	90.6	81.2	77.3	79.2		

Table 3: Cross-data generalization. Models are trained on Structured3D train set but evaluated on SceneCAD val set.

Method	Room			Corner			Angle		
	IOU	Prec.	Rec.	F1	Prec.	Rec.	F1		
HEAT [13]	52.5	50.9	51.1	51.0	42.2	42.0	41.6		
RoomFormer [12]	74.0	56.2	65.0	60.3	44.2	48.4	46.2		
FRI-Net [15]	80.6	66.4	79.5	72.4	56.3	67.2	61.3		
PolyRoom [37]	85.2	77.8	79.9	78.9	59.4	61.7	60.6		
CAGE (Ours)	85.6	68.2	87.5	76.7	54.6	70.6	61.6		

5.3 Ablation Study

We evaluate the impact of three core components: the edge-based polygon representation, the denoising (DN) training strategy, and the image backbone. As shown in Table 4, both the Edge and DN modules yield consistent improvements individually, and their combination results in further performance gains. By contrast, incorporating SwinTransformer-V2 as the backbone provides only marginal benefits beyond the combined Edge and DN configuration. Further analysis and visual examples are provided in Supplementary Material B.5.

Polygon Representation via Edge. We first examine the effect of the proposed Edge module, as shown in the third row of Table 4. In this variant, the original corner-based representation in RoomFormer is replaced with an edge-based formulation, while keeping all other settings unchanged.

Table 4: Ablation study on Structured3D. Edge and DN modules significantly improve performance. SwinT backbones provide limited or reduced gains compared to ResNet50. RoomFormer [12] results are reproduced using our training setup for fair comparison.

Method	Edge	DN	Backbone	Room			Corner			Angle		
				Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
RoomFormer	–	–	ResNet50[46]	96.9	95.9	96.4	87.3	83.8	85.5	81.2	77.9	79.5
Ours	✓	–	ResNet50[46]	98.7	97.6	98.2	91.2	87.0	89.0	85.1	81.3	83.2
Ours	–	✓	ResNet50[46]	98.0	96.2	97.1	90.4	84.4	87.3	85.0	79.5	82.2
Ours	–	–	SwinT_V2[45]	97.0	95.7	96.3	87.3	83.4	85.3	80.5	76.9	78.6
Ours	✓	✓	ResNet50[46]	99.6	98.7	99.2	93.9	88.8	91.3	89.7	84.9	87.2
Ours	✓	✓	SwinT_V1[47]	99.4	98.5	98.9	94.5	89.1	91.7	91.1	85.9	88.4
Ours	✓	✓	SwinT_V2[45]	99.6	98.7	99.1	95.0	88.6	91.7	92.5	86.4	89.3

The Edge module consistently improves performance across all evaluation metrics, increasing Room F1 from 96.4% to 98.2%, Corner F1 from 85.5% to 89.0%, and Angle F1 from 79.5% to 83.2%. These results demonstrate that representing polygons by edges enables the model to more effectively capture structural geometry and enhances its ability to localize wall boundaries.

Training Strategy via Denoising. To further improve learning stability and geometric reasoning, we incorporate a denoising strategy in which ground-truth data with noise are introduced alongside random queries during training. This approach supports convergence in the early stages of prediction and stabilize the Hungarian Matching process. As illustrated in the third row of Table 4, the addition of the DN module alone leads to moderate gains. While the DN module is less impactful than the Edge module in isolation, its value becomes more evident in combination.

When both Edge and DN modules are integrated (fifth row of Table 4), we observe substantial performance gains across all metrics, with Room F1 rising to 99.2%, Corner F1 to 91.3%, and Angle F1 to 87.2%. Notably, the most significant gains are observed in the Corner and Angle metrics, indicating improved precision in local geometric structures. This synergy suggests that the Edge and DN modules complement each other by jointly enhancing both global room layout understanding and fine-grained structural details.

Image Backbone. We evaluate the impact of different image backbones—ResNet-50 [46], SwinTransformer-V1 [47], and SwinTransformer-V2 [45]—while keeping the proposed Edge and DN modules fixed. All configurations utilize four image scales and a token dimension of 256. As shown in the last three rows of Table 4, integrating Swin Transformer backbones yields mixed results. While SwinTransformer-V2 achieves the highest Angle F1 score (89.3%) and slightly improves Corner F1 (91.7%), these gains are relatively modest compared to the improvements brought by the Edge and DN modules. Notably, ResNet-50 combined with both modules attains the highest Room F1 score (99.2%), underscoring its effectiveness in capturing global structural context.

These results suggest that the primary performance improvements stem from the proposed architectural enhancements rather than the choice of backbone. SwinTransformer variants offer benefits in modeling local geometric relationships through shifted windowing and cross-boundary attention, particularly enhancing the Corner and Angle metrics. However, the convolutional features of ResNet-50 remain highly effective for global layout prediction. Overall, SwinTransformer-V2 provides a balanced trade-off between local and global representation capabilities. Therefore, unless otherwise stated, we adopt the configuration with Edge, DN, and SwinTransformer-V2 as the default model for subsequent experiments.

6 Conclusion

We present **CAGE**, an end-to-end floorplan reconstruction framework that models rooms as sequences of edges using a dual-query transformer architecture. By predicting edges instead of corners, our method captures global geometric structure and directional priors, enabling robust inference of room layouts even from partially observed wall segments through edge-based attention. This significantly improves resilience to occlusion and layout noise. A denoising-based training strategy with perturbed and latent queries facilitates stable and efficient iterative refinement. Extensive experiments on Structured3D and SceneCAD demonstrate state-of-the-art performance. Our method still puzzles over reconstructing scenes with dense occlusions or fine structural elements. Future work includes integrating semantic priors, extending to full architectural elements, and scaling to outdoor scenarios.

References

- [1] Fayolle, P.-A., M. Friedrich. A survey of methods for converting unstructured data to csg models. *Computer-Aided Design*, 168:103655, 2024.
- [2] Zhu, Q., L. Fan, N. Weng. Advancements in point cloud data augmentation for deep learning: A survey. *Pattern Recognition*, page 110532, 2024.
- [3] Li, J., C. L. Chan, J. Le Chan, et al. Cognitive navigation for indoor environment using floorplan. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9030–9037. IEEE, 2021.
- [4] Ma, Z., Y. Liu, J. Li. Review on automated quality inspection of precast concrete components. *Automation in Construction*, 150:104828, 2023.
- [5] Liu, H., C. Cao, H. Ye, et al. Lightweight structured line map based visual localization. *IEEE Robotics and Automation Letters*, 9(6):5182–5189, 2024.
- [6] Chen, C., R. Wang, C. Vogel, et al. F3Loc: Fusion and filtering for floorplan localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18029–18038. 2024.
- [7] Ibrahem, H., A. Salem, H.-S. Kang. ST-RoomNet: Learning room layout estimation from single image through unsupervised spatial transformations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3376–3384. 2023.
- [8] Jiang, Z., Z. Xiang, J. Xu, et al. LGT-Net: Indoor panoramic room layout estimation with geometry-aware Transformer network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1644–1653. 2022.
- [9] Zheng, Z., J. Li, L. Zhu, et al. GAT-CADNet: Graph attention network for panoptic symbol spotting in CAD drawings. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11737–11746. 2022.
- [10] Huang, J., S. Zhang, B. Duan, et al. Arrangementnet: learning scene arrangements for vectorized indoor scene modeling. *ACM Transactions on Graphics (TOG)*, 42(4):1–15, 2023.
- [11] Chen, J., C. Liu, J. Wu, et al. Floor-sp: Inverse cad for floorplans by sequential room-wise shortest path. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2661–2670. 2019.
- [12] Yue, Y., T. Kontogianni, K. Schindler, et al. Connecting the dots: Floorplan reconstruction using two-level queries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 845–854. 2023.
- [13] Chen, J., Y. Qian, Y. Furukawa. Heat: Holistic edge attention transformer for structured reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16448–16457. 2022.
- [14] Su, J.-W., K.-Y. Tung, C.-H. Peng, et al. Slibo-net: Floorplan reconstruction via slicing box representation with local geometry regularization. *Advances in Neural Information Processing Systems*, 36:48781–48792, 2023.
- [15] Xu, H., J. Xu, Z. Huang, et al. Fri-net: Floorplan reconstruction via room-wise implicit representation. In *European Conference on Computer Vision*, pages 1–17. Springer, 2024.
- [16] Zheng, J., J. Zhang, J. Li, et al. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 283–299. 2020.
- [17] Avetisyan, A., T. Khanova, C. Choy, et al. Scenecad: Predicting object alignments and layouts in rgb-d scans. In *European Conference on Computer Vision (ECCV)*, pages 596–612. 2020.
- [18] Ochmann, S., R. Vock, R. Wessel, et al. Automatic reconstruction of parametric building models from indoor point clouds. *Computers & Graphics*, 54:94–103, 2016.

[19] Han, J., Y. Liu, M. Rong, et al. FloorUSG: Indoor floorplan reconstruction by unifying 2D semantics and 3D geometry. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196:490–501, 2023.

[20] Sun, C., C.-W. Hsiao, M. Sun, et al. HorizonNet: Learning room layout with 1D representation and pano stretch data augmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1047–1056. 2019.

[21] Liu, W., T. Yang, Y. Wang, et al. Symbol as points: Panoptic symbol spotting via point-based representation. *International Conference on Learning Representations*, 2024.

[22] Ganon, K., M. Alper, R. Mikulinsky, et al. Waffle: Multimodal floorplan understanding in the wild. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1488–1497. IEEE, 2025.

[23] Fang, H., F. Lafarge, C. Pan, et al. Floorplan generation from 3d point clouds: A space partitioning approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, 175:44–55, 2021.

[24] Ikehata, S., H. Yang, Y. Furukawa. Structured indoor modeling. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1323–1331. 2015.

[25] Cai, R., H. Li, J. Xie, et al. Accurate floorplan reconstruction using geometric priors. *Computers & Graphics*, 102:360–369, 2022.

[26] Jiang, J., M. Zhao, S. Xin, et al. Structure-aware surface reconstruction via primitive assembly. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14171–14180. 2023.

[27] Xiong, B., Y. Jin, F. Li, et al. Knowledge-driven inference for automatic reconstruction of indoor detailed as-built bims from laser scanning data. *Automation in Construction*, 156:105097, 2023.

[28] Xu, J., B. Stenger, T. Kerola, et al. Pano2cad: Room layout from a single panorama image. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 354–362. IEEE, 2017.

[29] Shen, Z., C. Lin, K. Liao, et al. Panoformer: panorama transformer for indoor 360° depth estimation. In *European Conference on Computer Vision*, pages 195–211. Springer, 2022.

[30] Pantoja-Rosero, B. G., A. Rusnak, F. Kaplan, et al. Generation of lod4 models for buildings towards the automated 3d modeling of bims and digital twins. *Automation in Construction*, 168:105822, 2024.

[31] Liu, C., J. Wu, Y. Furukawa. Floornet: A unified framework for floorplan reconstruction from 3d scans. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 203–219. 2018.

[32] Stekovic, S., M. Rad, F. Fraundorfer, et al. Montefloor: Extending mcts for reconstructing accurate large-scale floor plans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16014–16023. 2021.

[33] He, K., G. Gkioxari, P. Dollár, et al. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969. 2017.

[34] Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[35] Browne, C. B., E. Powley, D. Whitehouse, et al. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[36] Sun, Q., C. Fang, S. Liu, et al. Polygraph: a graph-based method for floorplan reconstruction from 3d scans. *IEEE Transactions on Visualization and Computer Graphics*, 2025.

[37] Liu, Y., L. Zhu, X. Ma, et al. Polyroom: Room-aware transformer for floorplan reconstruction. In *European Conference on Computer Vision*, pages 322–339. Springer, 2024.

- [38] Chen, J., R. Deng, Y. Furukawa. Polydiffuse: Polygonal shape reconstruction via guided set diffusion models. *Advances in Neural Information Processing Systems*, 36:1863–1888, 2023.
- [39] Chen, Z., A. Tagliasacchi, H. Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 45–54. 2020.
- [40] Zhu, X., W. Su, L. Lu, et al. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations (ICLR)*. 2021.
- [41] Li, F., H. Zhang, S. Liu, et al. Dn-detr: Accelerate detr training by introducing query denoising. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13619–13627. 2022.
- [42] Milletari, F., N. Navab, S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016.
- [43] Lazarow, J., W. Xu, Z. Tu. Instance segmentation with mask-supervised polygonal boundary transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11823–11832. 2022.
- [44] Dai, A., A. X. Chang, M. Savva, et al. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2443. 2017.
- [45] Liu, Z., H. Hu, Y. Lin, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12009–12019. 2022.
- [46] He, K., X. Zhang, S. Ren, et al. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 2016.
- [47] Liu, Z., Y. Lin, Y. Cao, et al. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022. 2021.
- [48] Kingma, D. P., J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Our main claims are summarized in Figure 1, Section 3 and Section 4 offer detailed explanations. All claims are supported by extensive experimental results.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: The limitations of our work are stated in Supplementary Material B.3.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: This paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The model settings and training details of our experiments are discussed in Supplementary Material A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: **[No]**

Justification: Our code and pre-trained models will be released upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: **[Yes]**

Justification: We have explained the experimental settings in Section 5.1 and Supplementary Material A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: **[Yes]**

Justification: We have reported our quantitative results in Section 5.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer “Yes” if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The configurations our computer resources for each experiment have been reported in Section 5.1. The runtimes are given in Section 5.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: In every respect, we have tried our best to conform with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Our paper have proposed a novel neural network for robust floorplan reconstruction. And there is no societal impact of our work performed.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate

to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our work only focuses on floorplan reconstruction with benchmark datasets, which poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: When using code from a third party, we report the source of the code directly with the block of code used.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We will release our code and pre-trained models with included README files.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This work does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorosity, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of our research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

Supplementary Materials

A Implementation Details

Model Settings. Our primary image backbone is Swin Transformer V2 [45], while ResNet-50 [46] and Swin Transformer V1 [47] are used for comparison in the ablation study. Following RoomFormer [12], for ResNet-50, we extract multi-scale feature maps from the last three stages without using a feature pyramid network (FPN). For Swin Transformers, we adopt the official large-variant configurations: SwinV1 uses Swin-L pretrained weights, and SwinV2 uses SwinV2-L pretrained weights, both with a base dimension of $C = 192$ and stage depths of $[2, 2, 18, 2]$. The window sizes differ, with SwinV1 using a window size of 12 and SwinV2 using 16. For both ResNet-50 and Swin Transformer backbones, the fourth-scale feature map is obtained via a 3×3 convolution with stride 2 applied to the final-stage output. All feature maps are projected to 256 channels before being fed into the transformer. Pretrained weights from [46; 47; 45] are used for all backbone layers and are fine-tuned during training. Our transformer architecture comprises 6 encoder layers and 6 decoder layers, each with a hidden dimension of 256. We set the number of room feature codes m to 20 and the number of edge queries n to 40.

Training. We train our model using the Adam optimizer [48] with a weight decay of $1e^{-4}$. Depending on the dataset size, we train the model on Structured3D for 650 epochs with an initial learning rate of $2e^{-4}$, and on SceneCAD for 400 epochs with an initial learning rate of $5e^{-5}$. In both cases, the learning rate is decayed by a factor of 0.1 during the final 20% of epochs. The loss weights are set as $\lambda_{cls} = 0.6$, $\lambda_{edge} = 6$, $\lambda_{ras} = 1$, $\lambda_{DN_cls} = 0.6$, and $\lambda_{DN_edge} = 6$. For perturbed queries, the noise scale is set to 0.2 for class and 0.4 for edge. We implement our model in PyTorch and conduct all experiments on an NVIDIA GeForce RTX 4090 GPU with 24 GB of memory.

B Additional Results and Visualizations

B.1 Evolution of Edge Queries

To further examine the decoding process, we visualize a layer-wise edge query refinement as illustrated in Figure 9. As decoding progresses, the predicted edge structures become increasingly coherent and better aligned with the underlying point density map, even in regions with limited or missing wall evidence. Correspondingly, the reconstructed polygons evolve from coarse outlines to precise, watertight layouts. This illustrates the effectiveness of our progressive decoding strategy in improving both geometric accuracy and topological consistency.

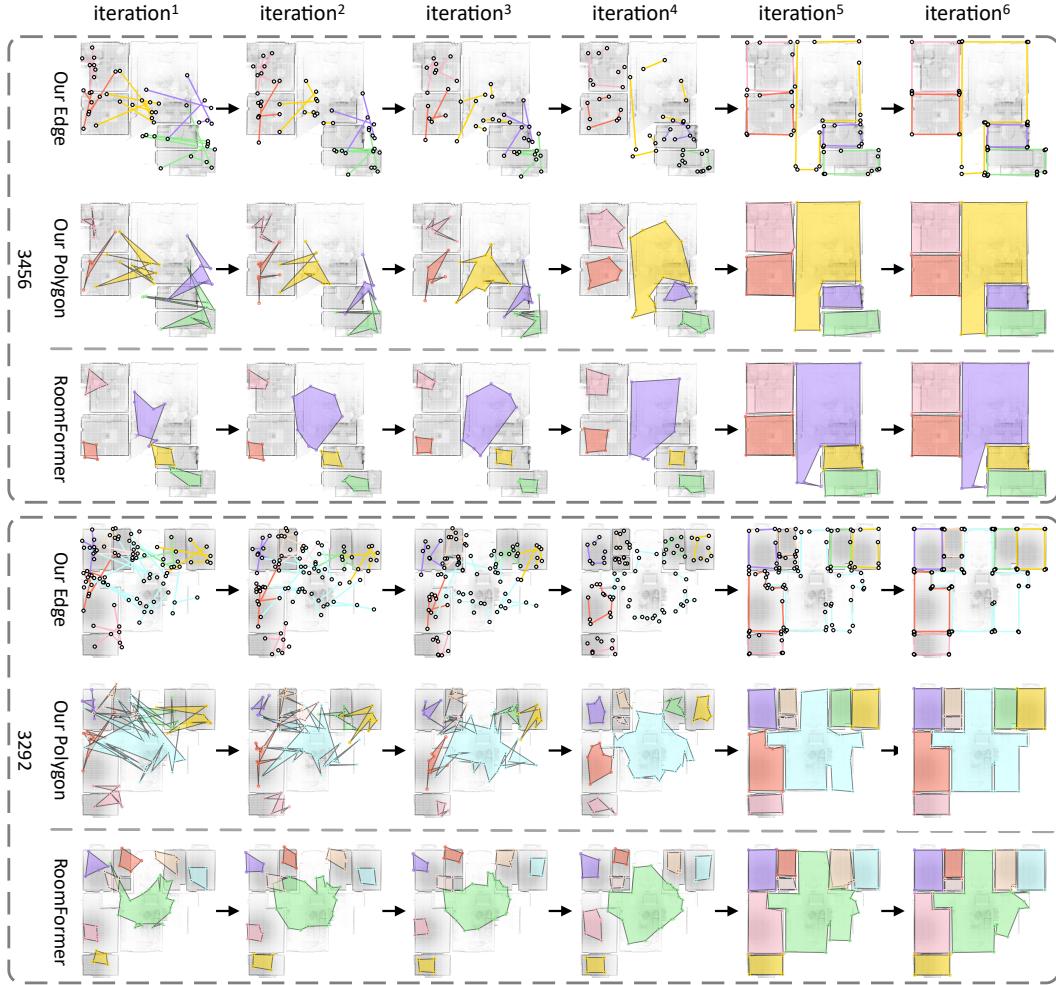


Figure 9: Additional results for edge query evolution and polygon formation. Each example shows a density map from Structure3D [16] (scene ID on the left), with edge query predictions over six decoding layers (top row), the corresponding polygon reconstructions (second row), and the polygon evolution of Roomformer [12] (third row).

B.2 More Reconstruction result on Structured3D

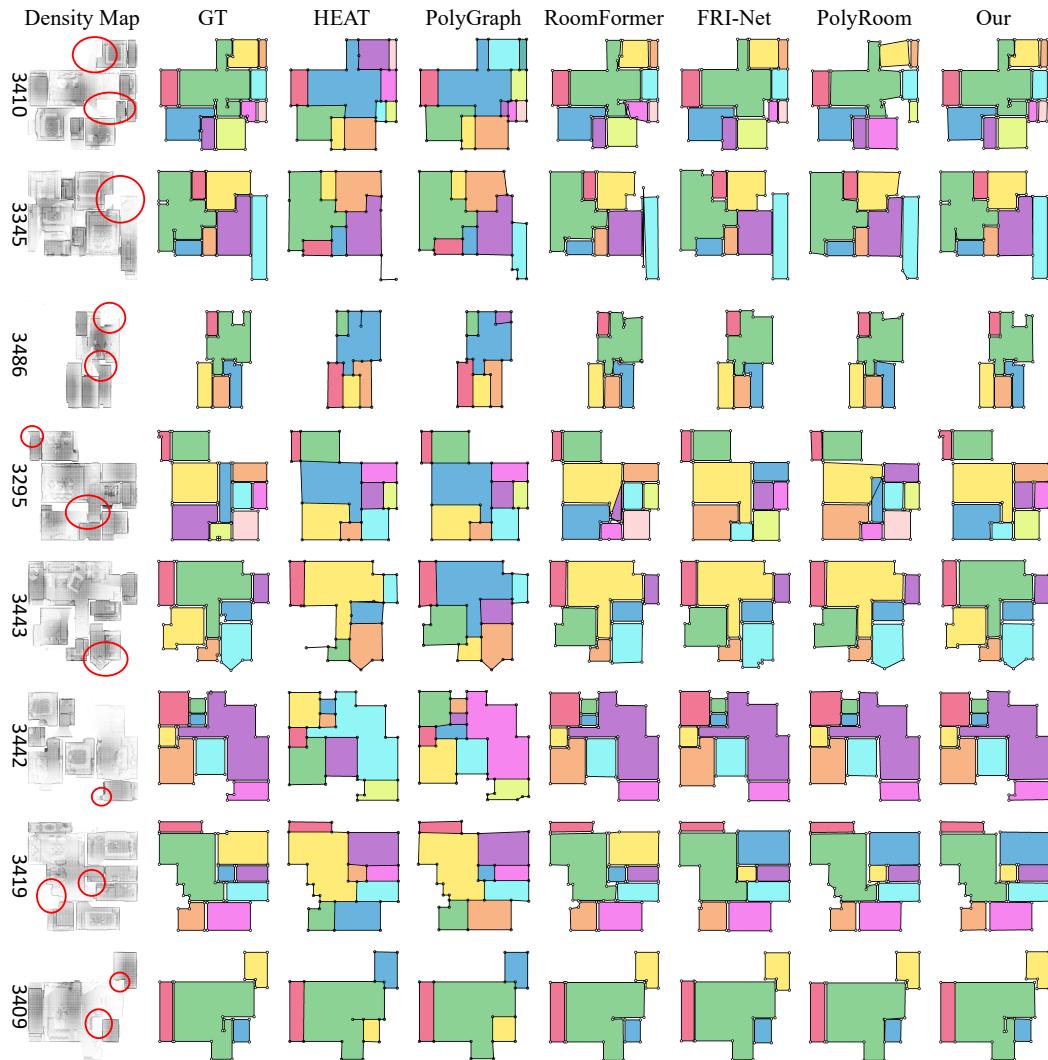


Figure 10: Additional results for Qualitative evaluations on Structured3D [17].

B.3 Failure cases on Structure3D

Our proposed **CAGE** model demonstrates strong performance on both the Structure3D and SceneCAD datasets. However, several typical failure cases remain, as illustrated in Fig. 11. In some cases, ambiguity in corner completion arises: certain ground truth layouts require polygonal closure at missing corners, while others do not, leading to inconsistencies in prediction (Fig. 11(a)). Additionally, small architectural elements such as interior rooms or columns may be absent from the input density map, resulting in missed detections (Fig. 11(b)). Severe occlusions caused by missing scan stations can further result in incomplete reconstructions (Fig. 11(c)). Lastly, we observe variability in prediction quality across different checkpoints within the same training run—some checkpoints yield strong results, while others perform poorly—likely due to sensitivity to parameter initialization or optimization instability (Fig. 11(d)). Addressing these challenges may involve incorporating more robust data augmentation techniques, exploring reinforcement learning-based refinement, or leveraging checkpoint ensembles to enhance model stability and consistency.

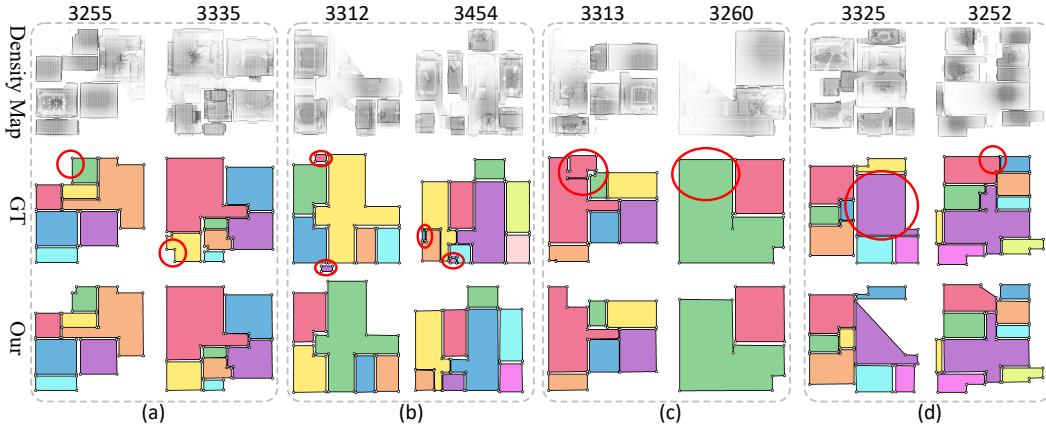


Figure 11: Failure cases on Structure3D [16], including polygon ambiguity(a), missing structures(b), incomplete scans(c), and checkpoint instability(d).

B.4 More Reconstruction result on SceneCAD

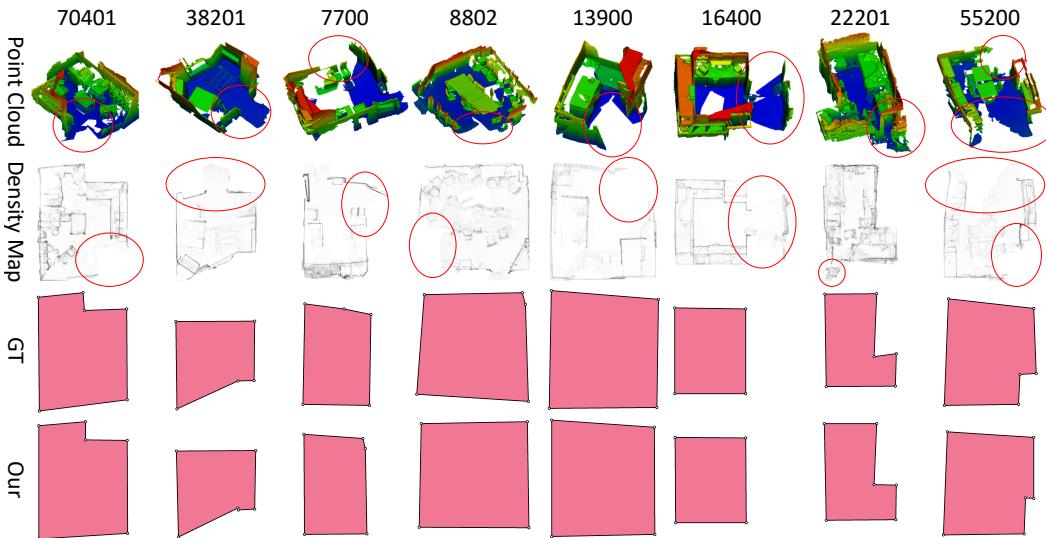


Figure 12: Additional results for Qualitative evaluations on SceneCAD [17].

Table 5: Sensitivity of reconstruction accuracy to threshold values. \dagger : the threshold we used in our experiments.

Eps	Room			Corner			Angle		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
0.2	99.59	98.69	99.14	95.04	88.57	91.70	92.57	86.40	89.39
0.1 \dagger	99.59	98.69	99.14	94.99	88.55	91.66	92.51	86.37	89.34
0.05	99.59	98.69	99.14	94.96	88.55	91.65	92.46	86.34	89.30
0.01	99.59	98.69	99.14	94.95	88.58	91.66	92.40	86.34	89.27
0.0001	99.59	98.69	99.14	94.94	88.60	91.67	92.38	86.34	89.26

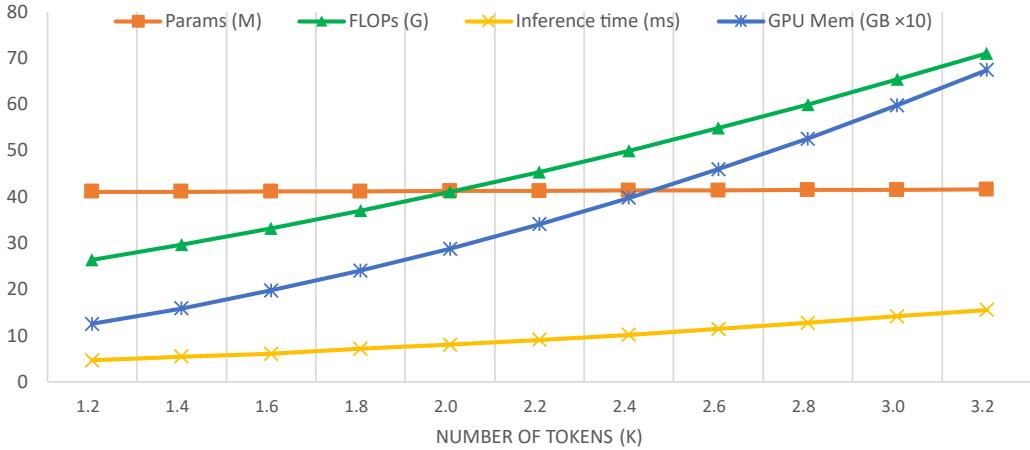


Figure 13: Impact of the number of tokens on model complexity and efficiency. As the token count increases, the number of parameters, FLOPs, inference time, and GPU memory usage grow nearly quadratically, reflecting the scalability characteristics of transformer-based architectures.

B.5 Extended Evaluation

Sensitivity to threshold in edge-to-polygon conversion. The edge-to-polygon conversion’s sensitivity to the threshold (epsilon) is minimal as shown in Table 5, varying epsilon from 0.0001 to 0.2 \times edge length changes Room metrics by <0.01% and other metrics by less than 0.05%, confirming stability and negligible impact on accuracy. This threshold-based approach offers a practical trade-off between theoretical rigor and engineering feasibility.

Computational cost analysis. We analyze the computational cost of our model with respect to the number of tokens used in the decoder. The metrics include the number of parameters, floating-point operations (FLOPs), inference time per image, and GPU memory usage for a batch of 10. Figure 13 summarizes these metrics for different token counts. Increasing the number of tokens leads to a higher number of parameters and FLOPs, longer inference time, and increased GPU memory consumption. Our method is inherently scalable and built on a general framework. The primary constraint is token length, which is bounded by GPU memory. Scaling is feasible by increasing the token count, though it come with quadratic complexity ($O(n^2)$ in both runtime and memory), as is typical for transformer-based models.

Visual Analysis. Figure 14 (main paper) and Figure 15 (appendix) illustrate qualitative differences across ablation settings. Models with edge-based representation and denoising exhibit better closure, alignment, and regularity, even under occlusion or missing data.

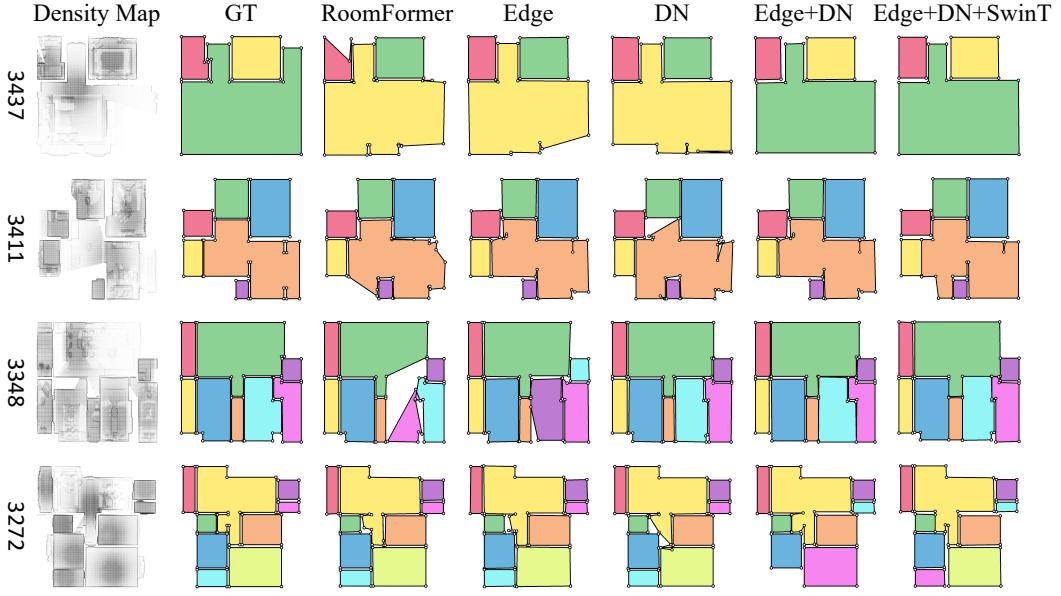


Figure 14: The ablation study on key components. RoomFormer serves as the baseline model. We progressively introduce the Edge module, the Denoising (DN) strategy, and the SwinTransformer-V2 (SwinT) backbone to assess their individual and combined effects.

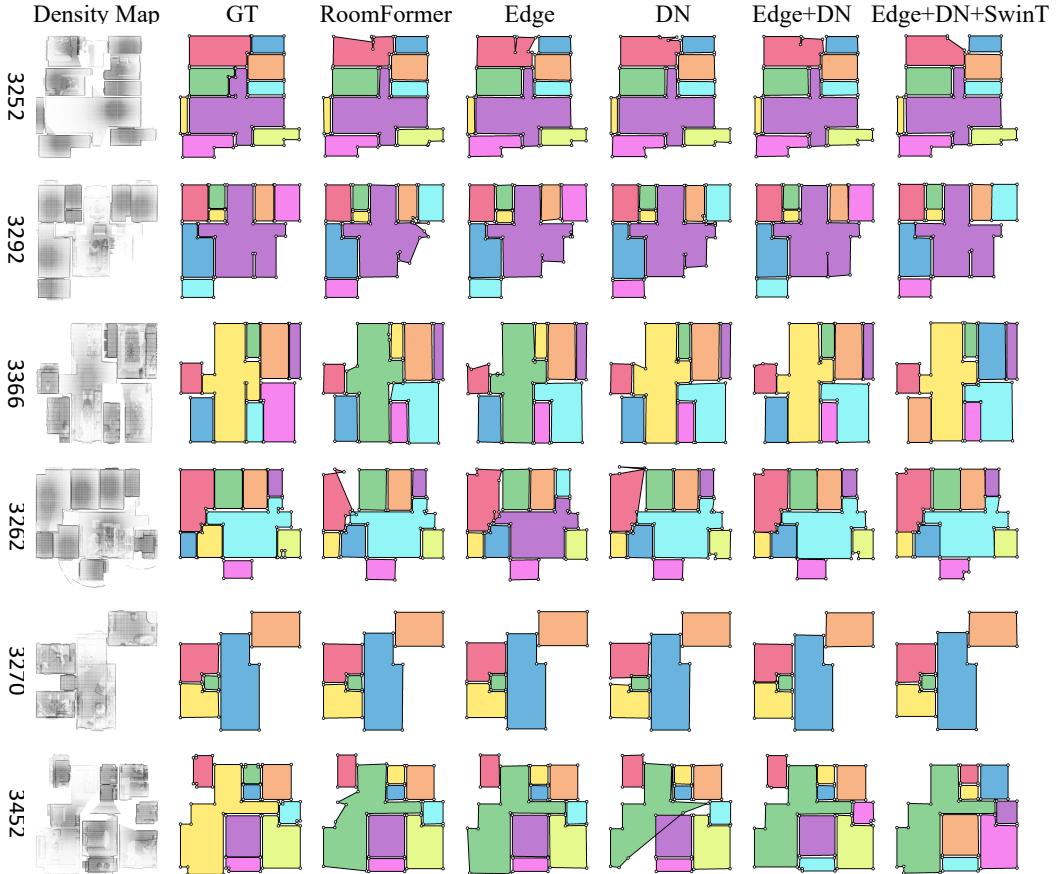


Figure 15: Additional results for The ablation study on key components. RoomFormer serves as the baseline model. We progressively introduce the Edge module, the Denoising (DN) strategy, and the SwinTransformer-V2 (SwinT) backbone to assess their individual and combined effects.