




TLC-Plan: A Two-Level Codebook Based Network for End-to-End Vector Floorplan Generation

Biao Xiong¹ , Zhen Peng¹, Ping Wang¹, Qiegen Liu² , and Xian Zhong^{1,†} 

¹ Hubei Key Laboratory of Transportation Internet of Things, School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China

² School of Information Engineering, Nanchang University, Nanchang 330031, China

Abstract

Automated floorplan generation aims to improve design quality, architectural efficiency, and sustainability by jointly modeling global spatial organization and precise geometric detail. However, existing approaches operate in raster space and rely on post hoc vectorization, which introduces structural inconsistencies and hinders end-to-end learning. Motivated by compositional spatial reasoning, we propose TLC-Plan, a hierarchical generative model that directly synthesizes vector floorplans from input boundaries, aligning with human architectural workflows based on modular and reusable patterns. TLC-Plan employs a two-level VQ-VAE to encode global layouts as semantically labeled room bounding boxes and to refine local geometries using polygon-level codes. This hierarchy is unified in a CodeTree representation, while an autoregressive transformer samples codes conditioned on the boundary to generate diverse and topologically valid designs, without requiring explicit room topology or dimensional priors. Extensive experiments show state-of-the-art performance on RPLAN dataset (FID = 1.84, MSE = 2.06) and leading results on LIFULL dataset. The proposed framework advances constraint-aware and scalable vector floorplan generation for real-world architectural applications. Source code and trained models are released at <https://github.com/rosolose/TLC-PLAN>.

CCS Concepts

• **Computing methodologies** → **Machine learning**; **Graphics systems and interfaces**; • **Theory of computation** → **Computational geometry**;

1. Introduction

Automatic floorplan generation is a central problem in artificial intelligence and architectural design, with broad impact on computer-aided design, game-level creation, interior planning, construction automation, and sustainable urban development [MA25,WLH*23]. A floorplan specifies the arrangement of rooms and functional spaces within a building, typically represented by walls on a two-dimensional plane [CKP*24]. Effective synthesis must simultaneously satisfy architectural constraints such as adjacency and alignment, while producing diverse and plausible designs that support downstream applications, including site-scale layout generation [WZLC24], BIM model construction [ZGL*25], and daylight performance optimization [HZL25].

Early approaches relied on procedural rules or expert-guided

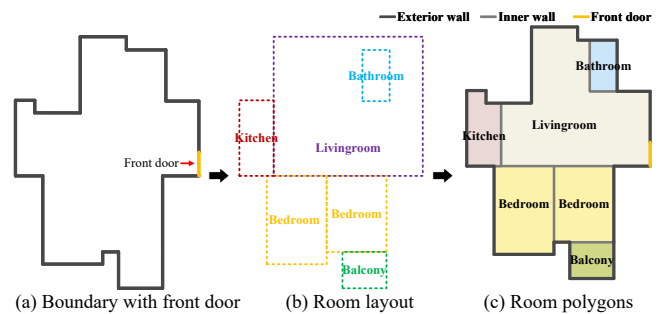


Figure 1: TLC-Plan directly generates vector floorplans end-to-end in two stages: (a) an input boundary with front-door location, (b) a layout-level codebook predicts semantically labeled room bounding boxes, and (c) a polygon-level codebook refines each box into a detailed room polygon. The resulting floorplan is geometrically aligned and CAD-ready, requiring no post-processing.

† Corresponding author: zhongx@whut.edu.cn. This work was supported by the National Natural Science Foundation of China (Grant No. 62271361) and the Hubei Provincial Key Research and Development Program (Grant No. 2024BAB039).

optimization [CKP*24], using stochastic search [MSK10] or portal graphs [BYMW13] to construct residential layouts. More re-

cent convolutional network-based methods learn layout distributions from real floorplans, but typically generate rasterized representations followed by vectorization [HHW22, ZSD24]. Although convolutional and adversarial networks can capture global structure, rasterization introduces discretization artifacts, leading to misaligned walls and corners during non-differentiable vectorization and hindering end-to-end training [LXN*25]. Vector-based methods, including FloorplanGAN’s hybrid generator-discriminator [LH22], Graph2Plan’s graph networks with predefined adjacency [HHT*20], and HouseDiffusion’s denoising framework [SHF23], alleviate some of these issues but depend on hand-crafted graphs or other manual inputs, which limits compositional generalization.

We argue that effective floorplan generation requires disentangled representations of global layout and local geometry, analogous to macro-level planning in urban design and micro-level detailing in building shapes [LLW*25, WWH*25]. Based on this insight, we introduce TLC-Plan, an end-to-end framework for structured vector floorplan synthesis (see Figure 1). TLC-Plan employs a two-level vector-quantized variational autoencoder (VQ-VAE) to encode floorplans into discrete latent codes: top-level codes model room layouts, while bottom-level codes refine precise room polygons. Given only an input boundary, an autoregressive transformer samples a CodeTree that combines these codes and decodes it into a topologically and geometrically aligned, computer-aided design (CAD)-ready floorplan. Experiments on RPLAN [WFT*19] and LIFULL [LC16] datasets show that TLC-Plan achieves state-of-the-art performance, including an FID of 1.84 and an MSE of 2.06 on RPLAN, while exhibiting superior geometric fidelity and strong generalization across diverse layout distributions. Our main contributions are threefold:

- We introduce TLC-Plan, a novel hierarchical VQ-VAE framework for direct end-to-end vector floorplan synthesis from boundaries, eliminating rasterization artifacts and advancing structured generative modeling in spatial AI.
- We propose a two-level discrete latent space that separates global layouts from local geometries, capturing architectural hierarchies without requiring room topology or dimensional priors, and enabling compositional generalization aligned with design principles.
- Extensive evaluations on RPLAN and LIFULL datasets demonstrate that TLC-Plan achieves state-of-the-art performance in coherence, precision, and diversity, outperforming existing baselines and supporting scalable applications in automated architecture.

2. Related Work

2.1. Raster-based Floorplan Generation

Early learning-based approaches model floorplans as multi-channel raster images with pixel-wise room labels. RPLAN [WFT*19] rasterizes layouts and applies heuristic vector tracing, which often introduces blur and structural inconsistencies. Graph2Plan [HHT*20] predicts room bounding boxes from a predefined adjacency graph, but its separate alignment module can fail on irregular boundaries. WallPlan [SWL*22] improves

semantic understanding through a wall-graph decoder, yet enforces geometric consistency only as a post-processing step. PlanNet [FHLF24] retrieves similar rasterized layouts instead of learning high-level design priors, limiting its generative flexibility.

Subsequent methods incorporate user interactivity and semantic conditioning. ActFloorGAN [WZC*23] synthesizes room layouts guided by human activity maps. iPLAN [HHW22] supports interactive editing via cascaded masks, but repeated raster operations accumulate geometric drift. MaskPLAN [ZSD24] employs dynamic masked autoencoders for flexible editing, yet remains constrained by pixel grids. Overall, raster-based pipelines depend on post hoc vectorization and struggle to achieve CAD-level geometric precision.

2.2. Vector-based Floorplan Generation

Vector-based methods represent layouts with explicit geometric or topological constraints. MIQP [WFLW18] formulates floorplan synthesis as a mixed-integer quadratic program that decomposes layouts into rectangles under size, position, and adjacency constraints. HouseGAN and HouseGAN++ [NCC*20, NHC*21] generate axis-aligned room boxes using relational GANs, but the rectangular assumption precludes curved walls and diagonal structures. Constraint-graph-based approaches [PGK*21, LXD*22] predict nodes and edges before solving an external optimization, which breaks end-to-end learning. G2Plan [BSU*22] and BubbleFormer [SZZW23] generate bubble diagrams from boundaries but stop short of producing full vector floorplans.

Structured and disentangled representation learning has also been explored in other domains [YYZ*25]. More recent work adopts diffusion models to improve geometric fidelity. HouseDiffusion [SHF23], Cons2Plan [HZD*24], and GSDiff [HWW*25] denoise corner or graph representations, but still rely on bubble diagrams and multi-stage pipelines. DiffPlanner [WP25] iteratively refines layouts, yet separates topology from geometry. In contrast, TLC-Plan learns a unified two-level CodeTree that jointly captures global layout and room-level geometry, enabling direct end-to-end vector floorplan generation without external graph supervision.

2.3. LLM-Guided Layout and Scene Design

Recent advances in large language models (LLMs) enable semantic control over spatial design through natural language, with applications spanning mechanical design [WCL*25], BIM modeling [DENB24], visual storytelling [YXH25], and mobility analysis [WJY*24]. HouseLLM [ZZT24] translates user prompts into bubble diagrams via chain-of-thought reasoning and diffusion, but degrades under long or ambiguous inputs. I-Design and Holodeck [CHS*24, YSW*24] support 3D scene arrangement, yet depend on fixed geometry templates. LayoutVLM [SLG*25] optimizes image-space layouts but cannot produce vector graphics.

Despite their semantic flexibility, most LLM-driven systems lack the geometric precision required for CAD-faithful floorplans. Mamba-CAD [LLSZ25] models long CAD sequences through self-supervised learning, while CAD2Program and RECAD [WZH*25,

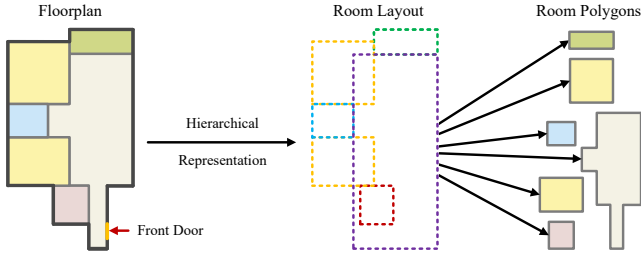


Figure 2: Hierarchical two-level representation: the layout level captures global room bounding boxes, while the polygon level refines them into detailed shapes, enabling structured codebook learning.

[LZG*25] reconstruct 3D parametric models from 2D sketches using vision-language and diffusion frameworks. By contrast, TLC-Plan introduces a discrete, geometry-aware code space that preserves vector accuracy, bridging semantic reasoning and precise floorplan synthesis, and providing a foundation for future LLM-driven architectural design.

3. Proposed Method

We present **TLC-Plan**, a two-stage generative framework that directly synthesizes high-quality vector floorplans from building boundaries. Inspired by architectural design principles and hierarchical modeling in HNC-CAD [XJL*23], TLC-Plan decomposes floorplan generation into two stages: global layout planning and room-level geometric refinement. Unlike HNC-CAD, which considers only geometric information, TLC-Plan explicitly incorporates semantic room types at the layout level and structural cues at the polygon level, enabling functional spatial reasoning. This hierarchy is realized through parallel VQ-VAE networks that learn a two-level codebook, which is then synthesized by an autoregressive transformer conditioned solely on the input boundary.

3.1. Hierarchical Floorplan Representation

Vectorized floorplans provide precise geometric and semantic descriptions required by CAD and architectural applications. TLC-Plan adopts a two-level hierarchical representation [YXHZ25] that decomposes a floorplan into a high-level room layout and low-level polygon geometries, as shown in Figure 2.

3.1.1. Room Layout

The global layout L is represented as an ordered sequence of room descriptors $(x_i, y_i, w_i, h_i, c_i)$ for $i = 1, \dots, M$, where (x_i, y_i) denote the bottom-left coordinates, (w_i, h_i) the room dimensions, and c_i the room type:

$$L = \{(x_i, y_i, w_i, h_i, c_i)\}_{i=1}^M. \quad (1)$$

Rooms are ordered by placing the living room first, followed by the remaining rooms sorted by increasing x and then y coordinates.

3.1.2. Room Polygons

Each room polygon P_i is defined as a clockwise-ordered list of vertices:

$$P_i = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}. \quad (2)$$

This vertex list delineates the geometric contour of the room polygon. The front door, if present, is encoded within the polygon by re-ordering the vertex sequence such that the two vertices corresponding to the door appear first, while preserving the overall clockwise orientation of the polygon.

3.1.3. Floorplan CodeTree

A complete floorplan is represented by its global layout L and the set of room polygons $\{P_i\}_{i=1}^M$:

$$F = [L, P_1, P_2, \dots, P_M]. \quad (3)$$

To enable compact representation and efficient generation, we discretize L and each P_i into codebook indices $c_L \in \mathcal{B}_L$ and $c_{P_i} \in \mathcal{B}_P$, yielding a hierarchical CodeTree:

$$C = [c_L, c_{P_1}, c_{P_2}, \dots, c_{P_M}]. \quad (4)$$

This discrete sequence jointly captures global spatial configuration and detailed room geometry, facilitating modular learning and autoregressive sampling.

3.2. Two-Level Codebook Learning

To learn modular and reusable design abstractions, TLC-Plan adopts a dual VQ-VAE framework [vdOVK18, RvV19]: one encoder-decoder pair for room layouts (see Figure 3(a)) and another for room polygons (see Figure 3(b)). The two models share the same architecture but use different tokenization and embedding strategies.

3.2.1. Room Embedding and Encoding

Each layout tuple (x, y, w, h, c) is discretized into 6-bit values and embedded into 32-D vectors. The resulting 320-D room vector is projected to a 256-D token using a two-layer MLP with positional encoding:

$$T_t^E = \text{MLP}(W_g x_t \| W_g y_t \| W_g w_t \| W_g h_t \| W_c c_t) + \gamma_t, \quad (5)$$

where $\|$ denotes concatenation, W_g embeds spatial components (x, y, w, h) , W_c embeds the room type, and γ_t is the positional encoding. A Transformer encoder processes the sequence $\{T_t^E\}$, and average pooling yields the layout feature $\bar{E}(T^E)$.

3.2.2. Vector Quantization

The encoded feature is quantized via nearest-neighbor search:

$$d = b_k, \quad k = \arg \min_i \|\bar{E}(T^E) - b_i\|^2, \quad (6)$$

where $\{b_i\}$ are codebook entries.

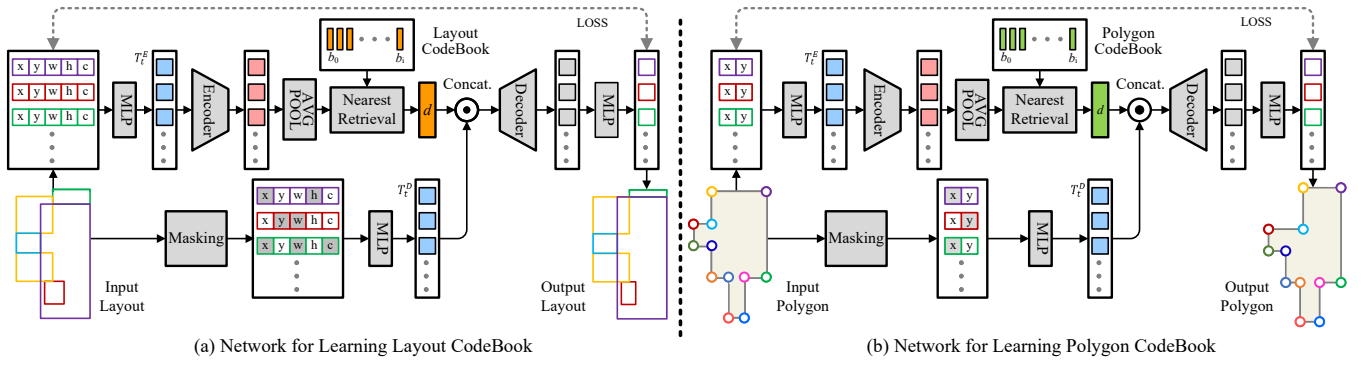


Figure 3: Network architecture for codebook learning: layout-level and polygon-level models share the same architecture but differ in input tokenization. The encoder extracts sequence features, which are quantized via a learned codebook, and the decoder reconstructs masked inputs from quantized codes to capture reusable design patterns.

3.2.3. Masked Reconstruction

To encourage abstraction learning, we randomly mask 30-70% of the input embeddings and require the decoder to reconstruct the missing values from the quantized codes. This masking strategy prevents direct memorization and forces the codebooks to capture essential structural patterns.

3.2.4. Codebook Update

TLC-Plan updates both the layout and polygon codebooks using an Exponential Moving Average (EMA) scheme, avoiding backpropagation through discrete code indices [HYS16]. During training, the encoder maps input tokens to continuous embeddings, which are quantized by assigning each embedding to its nearest codeword. For each codeword, EMA updates are applied by accumulating the assigned embeddings and their counts, and then updating the codeword as the normalized moving average of these embeddings. This process gradually adapts the codebook to the data distribution while maintaining training stability. The masked reconstruction objective encourages codewords to represent reusable design patterns, and the commitment loss ensures encoder outputs remain close to their assigned codewords.

3.2.5. Loss Function

The overall objective combines reconstruction, codebook, and commitment losses:

$$\mathcal{L} = \sum_t \text{EMD} \left(D(d, \{T_t^D\}), \mathcal{K}_{T_t} \right) + \left\| \text{sg} \left[\bar{E} \left(T^E \right) \right] - d \right\|^2 + \beta \left\| \bar{E} \left(T^E \right) - \text{sg} [d] \right\|^2, \quad (7)$$

with $\beta = 0.25$ and $\text{sg}[\cdot]$ the stop-gradient operator. We use the squared Earth Mover's Distance (EMD) [HYS17] for reconstruction:

$$\text{EMD} \left(D(d, \{T_t^D\}), \mathcal{K}_{T_t} \right) = \frac{1}{C} \sum_{c=1}^C \left(\sum_{i=1}^c p_{t,i} - \sum_{i=1}^c q_{t,i} \right)^2, \quad (8)$$

where p_i and q_i denote the predicted distribution $D(d, \{T_t^D\})$ and the target distribution \mathcal{K}_{T_t} , respectively.

3.2.6. Polygon-Level Model

The polygon-level VQ-VAE mirrors the layout-level model but operates on vertex sequences instead of bounding boxes, enabling precise room-level geometric modeling.

3.3. Vector Floorplan Generation

TLC-Plan synthesizes complete vector floorplans from input boundaries by autoregressively predicting a discrete *CodeTree* that jointly encodes global layout structure and detailed room geometry. During training, ground-truth *CodeTrees* provide supervision. At inference time, the model samples a *CodeTree* conditioned on boundary features and decodes it into a coherent vector floorplan.

3.3.1. Network Architecture

The end-to-end pipeline is illustrated in Figure 4(a). During training, a ground-truth floorplan is converted into a supervision *CodeTree* by tokenizing, encoding, and quantizing the layout, boundary, and room polygons using the pretrained layout and polygon codebooks. In parallel, the *Polygon Encoder* extracts boundary features T_l^D , following the same encoding strategy as in polygon codebook learning.

Conditioned on T_l^D and guided by the supervision *CodeTree*, the *CodeTree Decoder* autoregressively predicts the target *CodeTree* T_l^C . At inference, nucleus (top- p) sampling over boundary features T_l^E enables diverse *CodeTree* generation. The predicted *CodeTree* T_l^C is concatenated with T_l^D and passed to the *Polygon Decoder*, which generates room polygons T_l^P corner by corner, using delimiter tokens to separate rooms.

To ensure structural consistency across variable-length and irregular floorplans, we fix the room order during training: the living room is placed first, followed by the remaining rooms sorted by their bottom-left coordinates. All components, the *Polygon Encoder*, *CodeTree Decoder*, and *Polygon Decoder*, are implemented using standard Transformer encoder and decoder blocks.

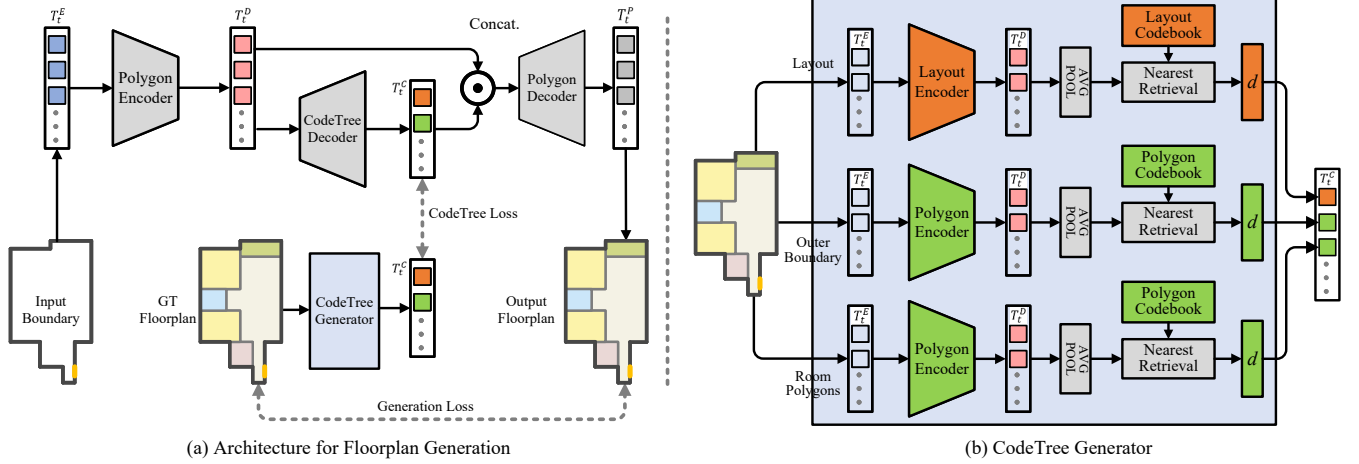


Figure 4: CodeTree-based vector floorplan generation: (a) given an input boundary, the model encodes boundary features, autoregressively predicts a CodeTree, and decodes it into room polygons; (b) the CodeTree Generator encodes the layout, boundary, and room polygons, quantizes them with learned codebooks, and concatenates them into a unified CodeTree for supervision and generation.

3.3.2. CodeTree Generator

The *CodeTree Generator* autoregressively constructs a discrete CodeTree T_t^C that represents the complete floorplan hierarchy, including the global layout, outer boundary, and individual room geometries. Generation is conditioned on boundary features T_t^E and the pretrained layout and polygon codebooks (see §3.2).

As shown in Figure 4(b), CodeTree construction proceeds sequentially: the layout code is generated first, followed by the outer boundary (treated as a special polygon), and then the polygon codes for each room. During training, the supervision CodeTree is obtained by tokenizing and quantizing the ground-truth layout and polygons via the learned encoders. This hierarchical representation allows the model to jointly learn global structure and fine-grained geometry directly from the input boundary.

3.3.3. Training Objective

CodeTree prediction is formulated as an autoregressive sequence modeling problem. At each decoding step t , the model predicts a discrete token z_t from one of three vocabularies: layout or polygon code indices $\mathcal{V}_{\text{code}}$, discretized coordinate bins $\mathcal{V}_{\text{pos}} = \{0, \dots, 63\}$, or room-type labels $\mathcal{V}_{\text{type}}$. Let $\tau_t \in \{\text{code}, \text{pos}, \text{type}\}$ denote the token type at step t , and $p_\theta(\cdot | \mathbf{z}_{<t})$ the predicted distribution conditioned on the preceding context. The per-step loss is defined by the cross-entropy:

$$\ell_t = -\log p_\theta(z_t^* | \mathbf{z}_{<t}, \tau_t), \quad (9)$$

where z_t^* is the ground-truth token.

We group tokens by type and define the average loss for each group as:

$$\mathcal{L}_\tau = \frac{1}{|\mathcal{T}_\tau|} \sum_{t \in \mathcal{T}_\tau} \ell_t, \quad \tau \in \{\text{code}, \text{pos}, \text{type}\}. \quad (10)$$

The final training objective balances the three components:

$$\mathcal{L} = w_1 \mathcal{L}_{\text{code}} + w_2 \mathcal{L}_{\text{pos}} + w_3 \mathcal{L}_{\text{type}}, \quad (11)$$

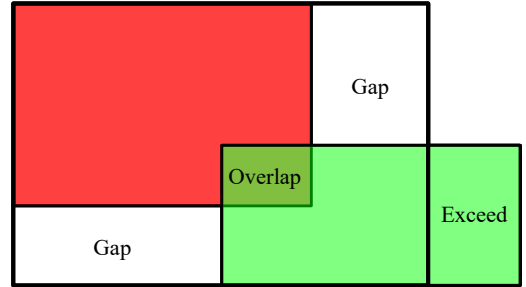


Figure 5: Geometric metrics for vector-based boundary constraints. Given an input boundary polygon (white) and generated room polygons (green and red), we measure Gap (uncovered interior area), Overlap (intersecting area between rooms), and Exceed (area extending beyond the boundary).

where $w_1 = w_2 = w_3 = 1$ in all experiments.

4. Experimental Results

4.1. Experimental Settings

4.1.1. Dataset and Evaluation Metrics

We evaluate TLC-Plan on RPLAN dataset [WFT*19], which contains 81,235 annotated residential layouts. Each floorplan is stored as a 256×256 four-channel image (interior structure, outer boundary, semantic masks, and instance IDs). Following [ZSD24], we split the dataset into train/val/test sets with an 8:1:1 ratio. For vector-based evaluation, we convert raster floorplans into polygonal representations using Graph2Plan [HHT*20], retaining six room types (living room, bedroom, bathroom, kitchen, balcony, and storage).

We report two complementary metric suites. First, we evaluate visual and statistical fidelity using Fréchet inception dis-

tance (FID_{img}) [HRU*17] on rendered images, together with mean squared error (MSE) on room counts (MSE_T), adjacency (MSE_A), and sizes (MSE_S) [ZSD24]. Second, to measure strict vector consistency under boundary constraints, we define three geometric metrics: Mean Ratio of Gap (MRG), Mean Ratio of Overlap (MRO), and mean ratio of exceed (MRE) (see Figure 5). For N samples,

$$MRG = \frac{1}{N} \sum_{i=1}^N \frac{A_{gap}^{(i)}}{A_{boundary}^{(i)}}, \quad (12)$$

$$MRO = \frac{1}{N} \sum_{i=1}^N \frac{A_{overlap}^{(i)}}{A_{boundary}^{(i)}}, \quad (13)$$

$$MRE = \frac{1}{N} \sum_{i=1}^N \frac{A_{exceed}^{(i)}}{A_{exceed}^{(i)} + A_{boundary}^{(i)}}, \quad (14)$$

where $A_{gap}^{(i)}$, $A_{overlap}^{(i)}$, and $A_{exceed}^{(i)}$ denote the gap, overlap, and exceed areas of the i -th sample, and $A_{boundary}^{(i)}$ is the corresponding boundary area.

4.1.2. Implementation Details

All models are implemented in PyTorch with Transformer backbones. The embedding and hidden dimensions are 256, the feed-forward dimension is 512, and dropout is 0.1. The codebook learning networks use 4 layers with 8 heads, while the generation network uses 6 layers with 8 heads. The layout and polygon codebooks contain 6,000 and 5,000 entries, respectively. We cap the number of rooms at 20 and the number of polygon vertices at 40. During codebook training, we randomly mask 30-70% of tokens. During generation, we cap the CodeTree length at 35 and the polygon sequence length at 800, and apply top- p sampling with $p = 0.95$.

4.1.3. Training Protocol

All experiments are conducted on a single NVIDIA RTX 4090 GPU. For RPLAN dataset, we apply data augmentation using 90° , 180° , and 270° rotations together with horizontal and vertical flips, yielding approximately 172,000 training samples. The layout and polygon codebook networks are trained independently with a batch size of 512 for 500 epochs, requiring about 3.8 hours and 24.8 hours, respectively. The generation network is trained with a batch size of 256 for 800 epochs, taking roughly 41.6 hours. Coordinates are normalized to $[0, 63]$. We optimize all models using AdamW [LH19], with a 200-step linear warm-up and a peak learning rate of 0.001.

4.1.4. Baselines

We compare TLC-Plan with four raster-based methods, RPLAN [WFT*19], Graph2Plan [HHT*20], iPLAN [HHW22], and MaskPLAN [ZSD24], and one vector-based method, GSDiff [HWW*25]. RPLAN rasterizes wall lines and heuristically vectorizes them. Graph2Plan retrieves a topology graph from the boundary and uses a CNN-GNN hybrid to generate aligned bounding boxes and rasterized layouts. iPLAN predicts per-pixel room types through cascaded mask refinement, while MaskPLAN

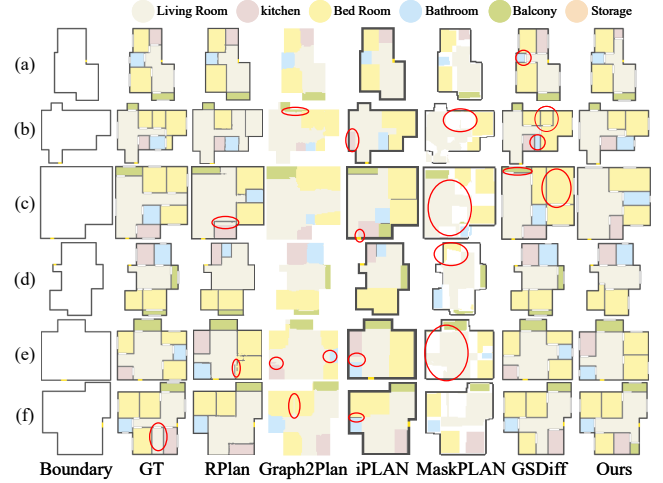


Figure 6: Qualitative comparison of floorplan generation: results from RPLAN, Graph2Plan, iPLAN, MaskPLAN, GSDiff, and our method, with the input boundary and GT shown for reference.

employs dynamic masked autoencoders for flexible editing. GSDiff first denoises corner coordinates with diffusion and then infers edges between corner pairs. All baselines rely on post hoc vectorization or separate alignment steps, which can lead to mis-decomposition and misalignment, whereas TLC-Plan operates in a single end-to-end vector pipeline.

4.2. Comparisons to State-of-the-Art Methods

4.2.1. Quantitative Results

Table 1 compares TLC-Plan with state-of-the-art methods on RPLAN dataset. Inference time is evaluated by generating one floorplan per image over 3,000 samples and averaging the results. TLC-Plan achieves the second-fastest inference speed, surpassed only by Graph2Plan. TLC-Plan attains the best FID_{img} of 1.84, indicating superior visual fidelity. For room sizes, TLC-Plan achieves the lowest MSE_S (2.406), benefiting from accurate polygon-level refinement via two-level code decoding. On room-type counts, TLC-Plan achieves $MSE_T = 0.200$, close to GSDiff (0.192) but higher than Graph2Plan (0.016), reflecting a trade-off between expressive vector synthesis and strict type alignment. For room adjacency, TLC-Plan obtains $MSE_A = 2.088$, comparable to RPLAN (1.858) and iPLAN (2.395) despite not explicitly modeling adjacency graphs; MaskPLAN remains the strongest (0.058) due to dense adjacency supervision.

TLC-Plan also excels in vector-specific geometric consistency, with all three vector metrics below 1%. It achieves the lowest mean ratio of gap (MRG) at 0.71% and the lowest mean ratio of exceed (MRE) at 0.10%, while maintaining a controlled mean ratio of overlap (MRO) at 0.56%. By comparison, GSDiff exhibits higher MRG (2.84%) and MRE (0.45%) but zero MRO, since its corner-based construction inherently avoids overlaps. Overall, these results validate TLC-Plan's effectiveness in end-to-end vector floorplan generation under geometric and topological constraints.

Type	Method	Venue	Size (M)	Time	FID _{img}	MSE _T	MSE _A	MSE _S	MRG	MRE	MRO
Raster	RPLAN	TOG'19	111	1.2	9.88	0.165	1.858	24.791	-	-	-
	Graph2Plan	TOG'20	8	0.1	2.36	0.016	1.691	6.053	-	-	-
	iPLAN	CVPR'22	31	2.8	3.84	0.256	2.395	18.393	-	-	-
	MaskPLAN	CVPR'24	947	2.5	10.58	<u>0.039</u>	0.058	44.420	-	-	-
Vector	GSDiff	AAAI'25	105	0.7	1.98	0.192	0.069	2.989	2.84%	0.45%	0.00%
	TLC-Plan	-	<u>22</u>	<u>0.6</u>	1.84	0.200	2.088	2.406	0.71%	0.10%	<u>0.56%</u>

Table 1: Quantitative comparison of TLC-Plan and state-of-the-art methods on RPLAN dataset. Bold and underlined indicate the best and second-best results, respectively.



Figure 7: Diverse floorplans from a single boundary: our method generates multiple layouts for the same input boundary, demonstrating sampling variability and design flexibility.

4.2.2. Qualitative Evaluation

Figure 6 compares floorplans generated by the baselines and TLC-Plan. Raster-based methods (RPLAN, Graph2Plan, iPLAN, MaskPLAN) produce pixel-level outputs that require post-processing, often leading to misalignment, jagged walls, or incomplete rooms. RPLAN frequently yields disjoint interiors and duplicated walls (c, e). Although generally box-aligned, Graph2Plan can generate unrealistic proportions (b, e, f). Without interactive editing, iPLAN may produce door conflicts and area inconsistencies (b, c, e). MaskPLAN, constrained by learned priors, can miss key rooms (b-d) or leave large empty regions (a, e, f). Among vector methods, GSDiff produces plausible layouts (d-f) but provides weaker control over room counts and sizes (a-c). In contrast, TLC-Plan consistently generates well-aligned, coherent, and diverse floorplans that better match the ground truth while satisfying boundary constraints.

4.2.3. Layout Diversity

As shown in Figure 7, TLC-Plan generates multiple plausible interior layouts from the same building boundary. The samples vary in room arrangement, orientation, and connectivity, while preserving geometric validity and architectural coherence. This diversity

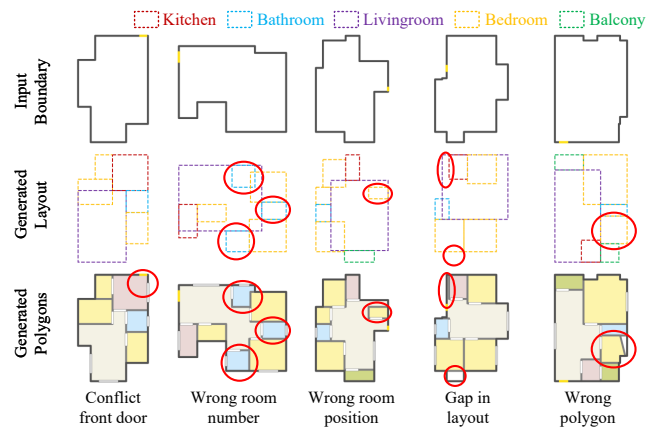


Figure 8: Failure cases on RPLAN.

arises from autoregressive sampling over the CodeTree, which encodes both high-level layout patterns and geometric refinements. Unlike deterministic pipelines, TLC-Plan supports design exploration through automatic alternative generation and customization. All sampled outputs remain topologically consistent and semantically complete, demonstrating robust CodeTree-guided generation.

4.2.4. Failure Cases

While TLC-Plan typically produces high-quality vector floorplans, challenging inputs can expose failure modes (see Figure 8). Common errors on RPLAN include misplaced front doors (e.g., adjacent to kitchens instead of living rooms), unrealistic bathroom counts for small units, bedrooms disconnected from exterior walls, residual boundary gaps, and locally invalid room polygons despite coherent global layouts.

In 3,000 generated samples, front-door misplacement (not adjacent to a living room) occurs in 3.77% (113) of cases, while fully coherent layouts—without gaps or invalid rooms—are achieved in 86.83% (2,605) of outputs. These errors primarily stem from three factors. (1) **Dataset bias:** the RPLAN dataset mainly contains layouts with 2 to 8 rooms, limiting generalization to larger or more complex plans and sometimes resulting in incomplete outputs. (2) **Codebook quantization:** limited discrete capacity can introduce refinement artifacts during polygon decoding. (3) **Autoregressive drift:** local sampling errors may accumulate across decoding steps, leading to global inconsistencies such as invalid adjacency or ge-



Figure 9: Generalization to extreme boundaries. The numbers denote boundary vertex counts, demonstrating robustness to complex shapes beyond the training distribution.

ometry. The model also struggles with highly complex boundaries, such as those with more than 30 corners, which are rare in training data. Some failure cases are further caused by annotation inconsistencies in RPLAN (e.g., case (f) in Figure 6).

Future improvements include higher-quality training data, expanded codebook capacity, and constrained or guided decoding (e.g., via reinforcement learning) to better enforce architectural validity and functional coherence.

4.2.5. Extreme Cases

Figure 9 presents floorplans generated from input boundaries that are substantially more complex than those seen during training. In the RPLAN dataset, outer boundaries contain between 4 and 35 vertices, with an average of 7.0, while the majority of training samples fall within 6 to 10 vertices. In contrast, the examples shown here include boundaries with up to 33 vertices, representing a clear distribution shift.

Cases A–F follow the Manhattan assumption and vary widely in boundary complexity, from 5 to 33 vertices. TLC-Plan generates coherent and geometrically valid layouts across all these cases, demonstrating robustness to increased boundary complexity. Cases G and H further evaluate generalization beyond the training distribution by introducing slanted boundary edges, which are not present in RPLAN dataset. Although small gaps appear near the slanted edges, the overall layouts remain reasonable and structurally consistent. These results indicate that TLC-Plan generalizes well to extreme and partially out-of-distribution boundary conditions, while also highlighting limitations when assumptions in the training data are violated.



Figure 10: Qualitative evaluation on LIFULL.

Method	FID _{img}	MSE _T	MSE _A	MSE _S
RPLAN	50.19	5.15	-	159.81
iPLAN	37.35	2.52	-	36.55
GSDiff	12.44	0.98	0.14	20.12
Our	9.08	0.34	4.86	8.96

Table 2: Quantitative evaluation on LIFULL dataset.

4.2.6. Generalization to LIFULL

We evaluate generalization on LIFULL [LC16], which contains 10,804 Japanese residential floorplans vectorized via Raster-to-Graph [HWS*24]. Using the same training configuration as for RPLAN, we only adjust the room taxonomy to match LIFULL’s 12-class scheme. Baseline results for RPLAN and iPLAN [HHW22] (without MSE_A), and for GSDiff [HWW*25]. As shown in Table 2 and Fig. 10, TLC-Plan achieves the best FID_{img} (9.08), MSE_T (0.34), and MSE_S (8.96), demonstrating strong visual realism, type accuracy, and size consistency. While GSDiff performs better on room adjacency (MSE_A), it lags in overall fidelity. These results confirm TLC-Plan’s ability to generalize across datasets with distinct distributions. However, as noted in [HWS*24], the lower annotation quality of LIFULL can limit vectorization fidelity and evaluation reliability.

4.3. Ablation Study

We conduct ablation studies to assess the impact of type encoding, architectural components, masking strategies, and codebook sizes. These experiments isolate the contribution of each factor and validate our design choices. Overall, the masked skip connection markedly improves generalization, while properly sized codebooks balance representational capacity and training stability.

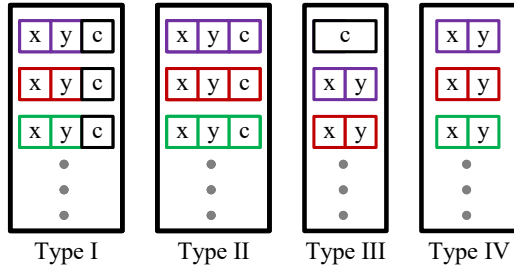


Figure 11: Polygon type encoding: (I) uniform room-type label on each vertex; (II) mixed labels with a dedicated front-door tag; (III) single prefix label for the entire polygon; (IV) no type labels (coordinates only).

Type	FID _{img}	MSE _T	MSE _A	MSE _S	MRG	MRE	MRO
I	1.91	0.227	2.201	2.620	1.40%	0.11%	1.12%
II	1.96	0.228	2.193	2.555	0.90%	0.15%	1.14%
III	1.93	0.221	2.170	2.435	1.07%	0.18%	0.94%
IV	1.84	0.200	2.088	2.406	0.71%	0.10%	0.56%

Table 3: Effect of different polygon type encoding strategies.

Structure	FID _{img}	MSE _T	MSE _A	MSE _S
w/o Masked Skip	16.49	2.107	5.117	8.695
w/o CodeTree	26.97	1.950	8.132	14.760
w/o Layout Code	24.29	1.282	6.977	12.553
w/o Polygon Code	14.01	1.020	5.244	6.248
Full Method	1.84	0.200	2.088	2.406

Table 4: Ablation study of key network components.

Based on these findings, we adopt the following configuration in all experiments (see §3): (i) omit type encoding in the polygon layer to improve geometric generalization; (ii) retain all architectural components, including the masked skip connection and hierarchical codetree; (iii) apply a random masking ratio of 30%-70% during VQ-VAE training; (iv) set codebook sizes to 6,000 for layouts and 5,000 for polygons; and (v) discretize layout tuples at 6-bit resolution.

4.3.1. Type Encoding in the Polygon Layer

In the layout layer, room types are encoded in (x, y, w, h, c) , where c denotes the semantic class. At the polygon level, we examine whether type labels benefit geometric modeling. Counterintuitively, removing type information yields slightly better performance, likely because semantics are already conveyed by the codetree and layout codebook, allowing the polygon codebook to focus on geometry.

We compare four strategies (see Figure 11): Type I assigns a uniform type label to all vertices; Type II additionally marks front-door corners; Type III prepends a single type label to the vertex sequence; and Type IV uses coordinates only. As shown in Table 3, Type IV consistently performs best across all metrics, supporting the benefit of decoupling semantics from geometry.

Masking Range	FID _{img}	MSE _T	MSE _A	MSE _S
10%-90%	1.875	0.202	2.082	2.407
20%-80%	1.866	0.201	2.085	2.426
30%-70%	1.836	0.200	2.088	2.406
40%-60%	1.857	0.201	2.086	2.418

Table 5: Effect of masking ratio on generation quality.

Layout Codebook Size	Polygon Codebook Size	FID _{img}	MSE _T	MSE _A	MSE _S
4,000		3.26	0.373	3.151	3.586
5,000		2.68	0.262	2.467	3.054
6,000	5,000	1.84	0.200	2.088	2.406
7,000		1.86	0.202	2.106	2.443
8,000		1.90	0.218	2.173	2.598
	3,000	2.89	0.223	3.444	4.255
	4,000	2.07	0.212	2.655	3.098
6,000	5,000	1.84	0.200	2.088	2.406
	6,000	1.92	0.204	2.135	2.779
	7,000	2.02	0.212	2.303	2.953

Table 6: Performance under different layout and polygon codebook sizes.

4.3.2. Component Effect

Table 7 compares five variants: removing the masked skip connection (w/o Masked Skip), removing the codetree (w/o CodeTree), using only layout codes (w/o Polygon Code), using only polygon codes (w/o Layout Code), and the full model. Removing the masked skip connection significantly degrades performance, confirming its role in preventing token memorization and encouraging abstraction. Eliminating the codetree yields the worst results, as the model loses intermediate layout guidance. Using only one codebook sacrifices either global structure (polygon-only) or fine geometry (layout-only). The full model consistently performs best, indicating that both codebooks and the codetree are essential for structured vector floorplan generation.

4.3.3. Mask Ratio

To improve codebook generalization and promote pattern reuse, we train VQ-VAEs with a masked skip connection: a random portion of input tokens is masked and reconstructed from the latent codes. The masking ratio is sampled uniformly from intervals centered at 50%. As shown in Table 5, the 30%-70% range yields the best overall results, achieving the lowest FID_{img}, MSE_T, and competitive MSE_S. Wider ranges introduce excessive variance, while narrower ranges reduce reconstruction difficulty. We therefore use 30%-70% masking in all experiments.

Bits	FID _{img}	MSE _T	MSE _A	MSE _S
5-bit	2.73	0.190	2.081	2.529
6-bit	1.84	0.200	2.088	2.406
7-bit	2.81	0.186	2.048	3.026

Table 7: Effect of layout tuple quantization bit width.

4.3.4. Codebook Size

Codebook size affects both generation quality and stability. Larger codebooks can represent finer-grained patterns but may reduce utilization and impair generalization, while smaller codebooks are efficient but risk insufficient expressiveness. For the layout codebook, increasing the size from 4,000 to 6,000 consistently improves all metrics (see Table 6), suggesting that smaller codebooks cannot adequately separate distinct layout patterns. Further increasing beyond 6,000 yields diminishing returns and slight degradation, likely due to over-parameterization and lower effective usage. We therefore set the layout codebook size to 6,000.

For the polygon codebook, performance improves substantially from 3,000 to 5,000 entries, indicating that small codebooks cannot capture geometric variability. Beyond 5,000, gains saturate and slightly drop, again suggesting reduced generalization with overly large codebooks. We thus fix the polygon codebook size at 5,000, which provides the best trade-off between expressiveness and stability.

4.3.5. Limits on Layout Size and Complexity

The ablation on quantization bit width reveals a clear trade-off between representational precision and layout complexity. As shown in Table 7, 6-bit quantization achieves the best balance, yielding strong visual fidelity (FID_{img} = 1.84) and room-size accuracy (MSE_S = 2.406). In contrast, both 5-bit and 7-bit settings lead to degraded performance. With 5-bit quantization, insufficient resolution limits geometric detail, resulting in oversimplified contours. With 7-bit quantization, the increased discrete space introduces redundant variability under a fixed model capacity, which destabilizes long-range geometric dependency modeling during autoregressive decoding and degrades scale consistency, as reflected by the higher MSE_S (3.026).

Varying the quantization bit width has only a minor impact on topological adjacency error (MSE_A) and functional semantics (MSE_T), indicating that symbolic layout structure and continuous geometry are handled by different components of the model. Overall, quantization acts as a bottleneck that limits the amount of geometric information that can be compressed and faithfully reconstructed. As layout irregularity and functional complexity increase, the fixed-size codebook and autoregressive decoding struggle to jointly preserve fine-scale geometry and global structure. Within the complexity range of the target data, 6-bit quantization therefore represents a practical local optimum. Scaling to larger and more complex layouts will likely require architectural extensions, such as adaptive codebooks or hybrid representations that decouple global topology from local geometry to better allocate representational capacity.

5. Conclusions and Limitations

TLC-Plan introduces a hierarchical VQ-VAE framework for end-to-end vector floorplan generation that synthesizes layouts directly from input boundaries and avoids rasterization. By encoding global layout and local geometry into a unified codetree, it reflects human design workflows and supports compositional generalization. Experiments on RPLAN and LIFULL demonstrate state-of-the-art performance in structural coherence, geometric accuracy, and layout diversity, providing a scalable foundation for AI-driven architectural design. More broadly, TLC-Plan moves toward constraint-aware spatial design agents that unify hierarchical planning and geometric reasoning for architectural and urban applications.

Despite these strengths, TLC-Plan currently conditions only on the boundary and focuses on room partitioning. Future work will incorporate richer constraints, such as textual prompts, fixed structural elements, or predefined adjacencies. The codetree formulation is flexible and can integrate such conditions by conditioning autoregressive decoding on additional constraint tokens or masking subsequences to enforce fixed placements. Further directions include modeling interior elements (e.g., doors, windows, furniture) and scaling to multi-floor buildings and city-scale layouts, enabling more complex automated design scenarios.

References

- [BSU*22] BISHT S., SHEKHAWAT K., UPASANI N., JAIN R. N., TIWASKAR R. J., HEBBAR C.: Transforming an adjacency graph into dimensioned floorplan layouts. *cgforum* 41, 6 (2022), 5–22. 2
- [BYMW13] BAO F., YAN D., MITRA N. J., WONKA P.: Generating and exploring good building layouts. *tog* 32, 4 (2013), 122:1–122:10. 1
- [CHS*24] ÇELEN A., HAN G., SCHINDLER K., GOOL L. V., ARMENI I., OBUKHOV A., WANG X.: I-design: Personalized LLM interior designer. In *Proc. Eur. Conf. Comput. Vis. Workshop* (2024), pp. 217–234. 2
- [CKP*24] COGO E., KRUPALIJA E., PRAZINA I., BECIROVIC S., OKANOVIC V., RIZVIC S., MULAHASANOVIC R. T.: A survey of procedural modelling methods for layout generation of virtual scenes. *cgforum* 43, 1 (2024). 1
- [DENB24] DU C., ESSER S., NOUSIAS S., BORRMANN A.: Text2bim: Generating building models using a large language model-based multi-agent framework. *arXiv preprint arXiv:2408.08054* (2024). 2
- [FHLF24] FU Q., HE S., LI X., FU H.: Plannet: A generative model for component-based plan synthesis. *IEEE Trans. Vis. Comput. Graph.* 30, 8 (2024), 4739–4751. 2
- [HHT*20] HU R., HUANG Z., TANG Y., VAN KAICK O., ZHANG H., HUANG H.: Graph2plan: Learning floorplan generation from layout graphs. *ACM Trans. Graph.* 39, 4 (2020), 118. 2, 5, 6
- [HHW22] HE F., HUANG Y., WANG H.: iplan: Interactive and procedural layout planning. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (2022), pp. 7783–7792. 2, 6, 8
- [HRU*17] HEUSEL M., RAMSAUER H., UNTERTHINER T., NESSLER B., HOCHREITER S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Adv. Neural Inf. Process. Syst.* (2017), pp. 6626–6637. 6
- [HWS*24] HU S., WU W., SU R., HOU W., ZHENG L., XU B.: Raster-to-graph: Floorplan recognition via autoregressive graph prediction with an attention transformer. *cgforum* 43, 2 (2024), i–iii. 8
- [HWW*25] HU S., WU W., WANG Y., XU B., ZHENG L.: Gsdiff: Synthesizing vector floorplans via geometry-enhanced structural graph generation. In *Proc. AAAI Conf. Artif. Intell.* (2025), pp. 17323–17332. 2, 6, 8

- [HYS16] HOU L., YU C.-P., SAMARAS D.: Squared earth mover's distance-based loss for training deep neural networks. *arXiv preprint arXiv:1611.05916* (2016). 4
- [HYS17] HOU L., YU C.-P., SAMARAS D.: Squared earth mover's distance-based loss for training deep neural networks, 2017. URL: <https://arxiv.org/abs/1611.05916>, arXiv:1611.05916. 4
- [HZD*24] HONG S., ZHANG X., DU T., CHENG S., WANG X., YIN J.: Cons2plan: Vector floorplan generation from various conditions via a learning framework based on conditional diffusion models. In *Proc. ACM Int. Conf. Multimedia* (2024), pp. 3248–3256. 2
- [HZL25] HU X., ZHENG H., LAI D.: Prediction and optimization of daylight performance of AI-generated residential floor plans. *Build. Environ.* (2025), 113054. 1
- [LC16] LIFULL CO. L.: Lifull home's high resolution floor plan image data. Informatics Research Data Repository, National Institute of Informatics (dataset), 2016. 2, 8
- [LH19] LOSHCILOV I., HUTTER F.: Decoupled weight decay regularization. In *Proc. Int. Conf. Learn. Represent.* (2019). 6
- [LH22] LUO Z., HUANG W.: FloorplanGAN: Vector residential floorplan adversarial generation. *Autom. Constr.* 142 (2022), 104470. 2
- [LLSZ25] LI X., LOU Y., SONG Y., ZHOU X.: Mamba-cad: State space model for 3D computer-aided design generative modeling. In *Proc. AAAI Conf. Artif. Intell.* (2025), pp. 5013–5021. 2
- [LLW*25] LIU Y., LIU C., WANG B., JIAO W., WU B., FAN L., CHEN Y., LI F., XIONG B.: Cage: Continuity-aware edge network unlocks robust floorplan reconstruction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems (NeurIPS)* (2025), pp. pp. 1–12. 2
- [LXD*22] LIU J., XUE Y., DUARTE J. P., SHEKHAWAT K., ZHOU Z., HUANG X.: End-to-end graph-constrained vectorized floorplan generation with panoptic refinement. In *Proc. Eur. Conf. Comput. Vis.* (2022), pp. 547–562. 2
- [LXN*25] LIU J., XUE Y., NI H., YU R., ZHOU Z., HUANG S. X.: Computer-aided layout generation for building design: A review. *arXiv preprint arXiv:2504.09694* (2025). 2
- [LZG*25] LI P., ZHANG W., GUO J., CHEN J., YAN D.: Revisiting CAD model generation by learning raster sketch. In *Proc. AAAI Conf. Artif. Intell.* (2025), pp. 4869–4877. 2
- [MA25] MESELHY A., ALMALKAWI A.: A review of artificial intelligence methodologies in computational automated generation of high performance floorplans. *npj Clean Energy* 1, 1 (2025), 2. 1
- [MSK10] MERRELL P., SCHKUFZA E., KOLTUN V.: Computer-generated residential building layouts. *ACM Trans. Graph.* 29, 6 (2010), 181. 1
- [NCC*20] NAUATA N., CHANG K., CHENG C., MORI G., FURUKAWA Y.: House-gan: Relational generative adversarial networks for graph-constrained house layout generation. In *Proc. Eur. Conf. Comput. Vis.* (2020), pp. 162–177. 2
- [NHC*21] NAUATA N., HOSSEINI S., CHANG K., CHU H., CHENG C., FURUKAWA Y.: House-gan++: Generative adversarial layout refinement network towards intelligent computational agents. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (2021), pp. 13632–13641. 2
- [PGK*21] PARA W., GUERRERO P., KELLY T., GUIBAS L. J., WONKA P.: Generative layout modeling using constraint graphs. In *Proc. IEEE/CVF Int. Conf. Comput. Vis.* (2021), pp. 6670–6680. 2
- [RvV19] RAZAVI A., VAN DEN OORD A., VINYALS O.: Generating diverse high-fidelity images with VQ-VAE-2. In *Adv. Neural Inf. Process. Syst.* (2019), pp. 14837–14847. 3
- [SHF23] SHABANI M. A., HOSSEINI S., FURUKAWA Y.: Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (2023), pp. 5466–5475. 2
- [SLG*25] SUN F., LIU W., GU S., LIM D., BHAT G., TOMBARI F., LI M., HABER N., WU J.: LayoutVlm: Differentiable optimization of 3D layout via vision-language models. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (2025), pp. 29469–29478. 2
- [SWL*22] SUN J., WU W., LIU L., MIN W., ZHANG G., ZHENG L.: Wallplan: synthesizing floorplans by learning to generate wall graphs. *tog* 41, 4 (2022), 92:1–92:14. 2
- [SZZW23] SUN J., ZHENG L., ZHANG G., WU W.: Bubbleformer: Bubble diagram generation via dual transformer models. *cgforum* 42, 7 (2023). 2
- [vdOVK18] VAN DEN OORD A., VINYALS O., KAVUKCUOGLU K.: Neural discrete representation learning, 2018. URL: <https://arxiv.org/abs/1711.00937>, arXiv:1711.00937. 3
- [WCL*25] WANG S., CHEN C., LE X., XU Q., XU L., ZHANG Y., YANG J.: CAD-GPT: synthesizing CAD construction sequence with spatial reasoning-enhanced multimodal LLMs. In *Proc. AAAI Conf. Artif. Intell.* (2025), pp. 7880–7888. 2
- [WFLW18] WU W., FAN L., LIU L., WONKA P.: Miqp-based layout design for building interiors. *cgforum* 37, 2 (2018), 511–521. 2
- [WFT*19] WU W., FU X., TANG R., WANG Y., QI Y., LIU L.: Data-driven interior plan generation for residential buildings. *ACM Trans. Graph.* 38, 6 (2019), 234:1–234:12. 2, 5, 6
- [WJY*24] WANG J., JIANG R., YANG C., WU Z., ONIZUKA M., SHIBASAKI R., KOSHIZUKA N., XIAO C.: Large language models as urban residents: An LLM agent framework for personal mobility generation. In *Adv. Neural Inf. Process. Syst.* (2024). 2
- [WLH*23] WANG Y. T., LIANG C., HUAI N., CHEN J., ZHANG C. J.: A survey of personalized interior design. *cgforum* 42, 6 (2023). 1
- [WP25] WANG S., PAJAROLA R.: Eliminating rasterization: Direct vector floor plan generation with diffplanner. *IEEE Trans. Vis. Comput. Graph.* (2025). 2
- [WWH*25] WANG X., WANG L., HUANG W., LIU Y., ZHANG S., ZHONG X.: Local-global sparse transformer for road extraction from remote sensing imagery. *IEEE Trans. Geosci. Remote Sens.* (2025). 2
- [WZC*23] WANG S., ZENG W., CHEN X., YE Y., QIAO Y., FU C.: Actfloor-gan: Activity-guided adversarial networks for human-centric floorplan design. *IEEE Trans. Vis. Comput. Graph.* 29, 3 (2023), 1610–1624. 2
- [WZH*25] WANG X., ZHENG J., HU Y., ZHU H., YU Q., ZHOU Z.: From 2D CAD drawings to 3D parametric models: A vision-language approach. In *Proc. AAAI Conf. Artif. Intell.* (2025), pp. 7961–7969. 2
- [WZLC24] WANG L., ZHOU X., LIU J., CHENG G.: Automated layout generation from sites to flats using GAN and transfer learning. *Autom. Constr.* 166 (2024), 105668. 1
- [XJL*23] XU X., JAYARAMAN P. K., LAMBOURNE J. G., WILLIS K. D. D., FURUKAWA Y.: Hierarchical neural coding for controllable CAD model generation. In *Proc. Int. Conf. Mach. Learn.* (2023), pp. 38443–38461. 3
- [YSW*24] YANG Y., SUN F., WEIHS L., VANDERBILT E., HERRASTI A., HAN W., WU J., HABER N., KRISHNA R., LIU L., CALLISON-BURCH C., YATSKAR M., KEMBHAVI A., CLARK C.: Holodeck: Language guided generation of 3D embodied AI environments. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (2024), pp. 16277–16287. 2
- [YXHZ25] YANG L., XIAO Z., HUANG W., ZHONG X.: Storyllava: enhancing visual storytelling with multi-modal large language models. In *Proc. 31st Int. Conf. Comput. Linguistics* (2025), pp. 3936–3951. 2, 3
- [YYZ*25] YU Y., YUAN J., ZHONG X., ZHAO Q., LUO W., MAI L.: Sparse mixture of mambas for domain generalized atomic electron tomography augmentation. *IEEE Trans. Neural Netw. Learn. Syst.* (2025). 2
- [ZGL*25] ZENG P., GAO W., LI J., YIN J., CHEN J., LU S.: Automated residential layout generation and editing using natural language and images. *Autom. Constr.* 174 (2025), 106133. 1

- [ZSD24] ZHANG H., SAVOV A., DILLENBURGER B.: Maskplan: Masked generative layout planning from partial input. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.* (2024), pp. 8964–8973. [2, 5, 6](#)
- [ZZT24] ZONG Z., ZHAN Z., TAN G.: Housellm: LLM-assisted two-phase text-to-floorplan generation. *arXiv preprint arXiv:2411.12279* (2024). [2](#)

Supplementary Materials

Appendix A: Additional Results and Visualizations



Figure 12: Additional qualitative comparison on RPLAN (Part I).

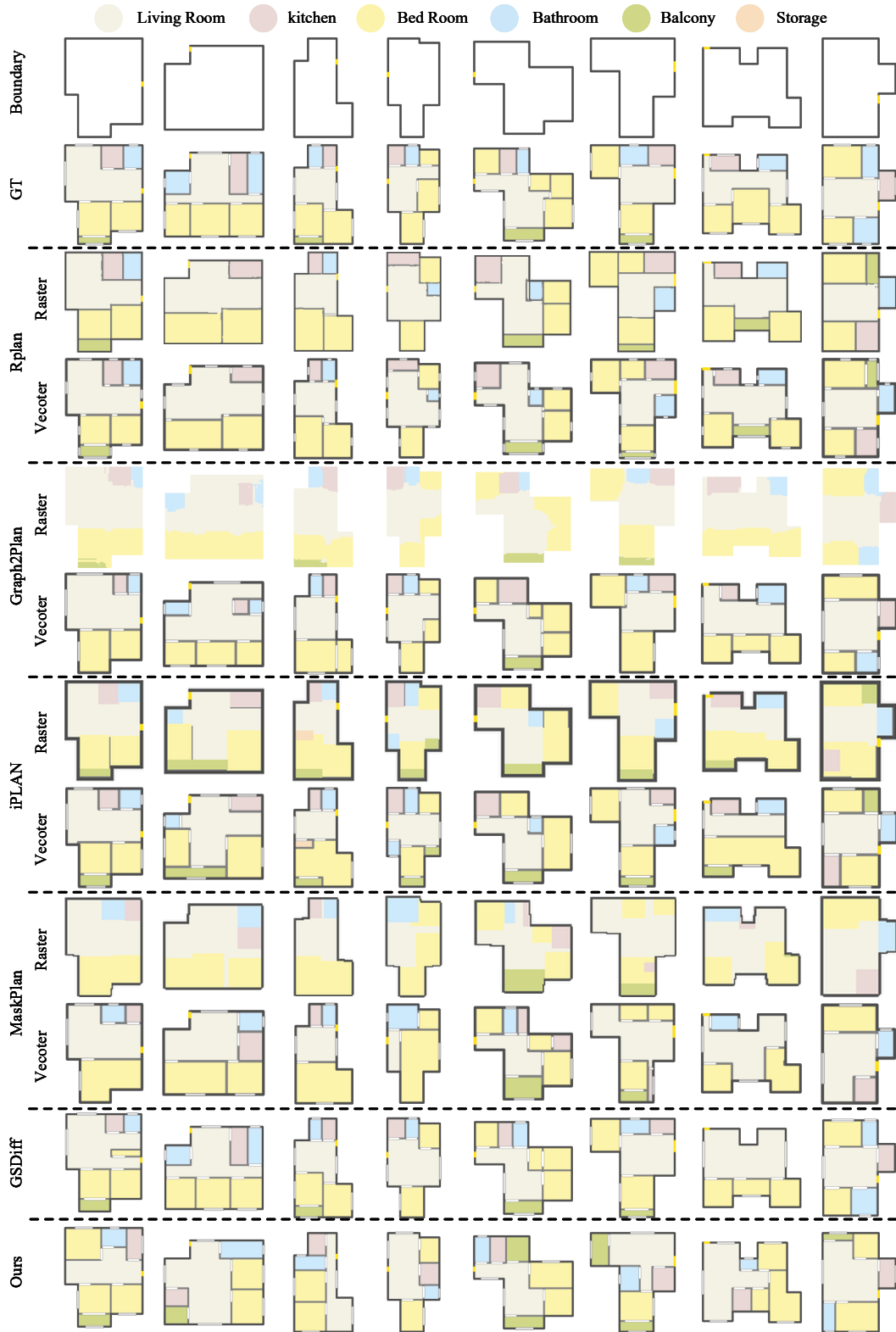


Figure 13: Additional qualitative comparison on RPLAN (Part II).



Figure 14: Diverse generations from a single boundary on RPLAN (Part I).



Figure 15: Diverse generations from a single boundary on RPLAN (Part II).