Supplementary Document for
CG-Floor: Centroid-Guided Diffusion for Large-Scale Floorplan Generation

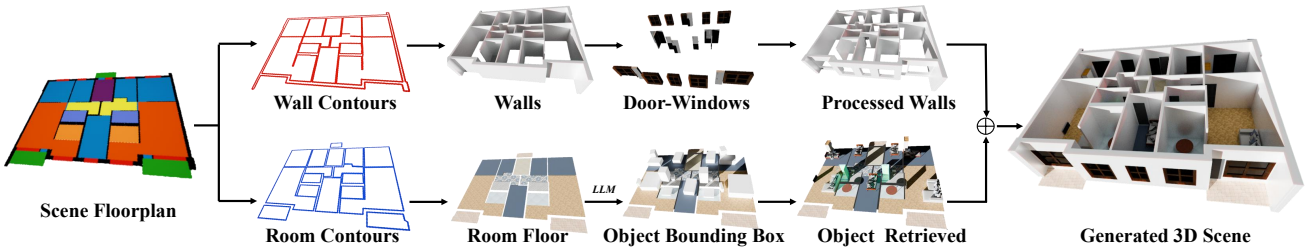

Figure 1. The generated 2D floorplans can be converted into 3D scenes, to demonstrate the effectiveness and practicality of our framework's output.

In this document, we provide the following supplementary contents:

- 3D Scene Generation Details in Section A
- Text-to-Graph Generation Details in Section B
- User Study Details in Section C
- Small-Scale Floorplan Results in Section D
- Additional Experimental Details in Section E
- Additional Results in Section F
- Limitations in Section G

We also provide a demo video along with this document.

A. 3D Scene Generation Details

To enhance the visual fidelity of generated results, as shown in Figure 1, our pipeline transforms the generated floorplan into a 3D scene. The process begins by vectorizing the floorplan to extract the contours of rooms, walls, and door-window structures. These contours are then converted into corresponding 3D meshes. Next, an LLM acting as a virtual interior designer, generates the categories and placements of 3D objects for each room based on its semantics. These assets are then retrieved, positioned accordingly, and integrated with the structural meshes to form a final 3D scene. The following sections describe each step of this pipeline in detail.

Vectorization. For generated floorplan $\hat{F}$, we extract room contours E_k via border-following algorithm [15], then simplify vertices using RDP algorithm [3]:

$$\widetilde{E}_k = \mathrm{RDP}(E_k), \quad |\widetilde{E}_k| \ll |E_k| \tag{1}$$

Door-windows layout is also extracted with connectivity components, but fitted into a rectangular shape.

Scaling: Since the floorplan is normalized to a unit square, a K-Nearest Neighbors (KNN) model is employed to predict the actual scale factor α_k by referencing similar floorplans in the training dataset, enabling appropriate scaling of the contours $\widetilde{E}_k^{3D} = \alpha_k \widetilde{E}_k$.

3D Conversion. We extrude each scaled vectorized contour $\widetilde{E}_k^{3D}$ to form a floor mesh, extrude wall contours along the vertical axis by height of $H = 3.5$ meters to form wall meshes, perform boolean subtraction of rectangular door/window openings, and accurately place the door and window meshes according to the rectangular shape and preset height.

Object Retrieval and Placement. For room R_k, generate 3D asset set via LLM:

$$O_k = \left\{ o_k^i = (c_k^i,\, p_k^i,\, s_k^i,\, r_k^i,\, d_k^i) \mid i = 1, \ldots, N_k \right\} \tag{2}$$

where c, p, s, r, d denote class, position, scale, rotation, and description. We retrieve each asset via CLIP-guided search (e.g. in Objaverse [2]), pick the top match asset $\hat{m}_k^i$, then transform it by (p_k^i, s_k^i, r_k^i). The final scene merges floor, wall, door/window meshes and all placed assets $\{\hat{m}_k^i\}_{i=1}^{N_k}$.

Specifically, the prompt we provide to the LLM follows the prompt format proposed in LayoutGPT [4], as shown in Figure 2. It includes a task description, an output format specification, several examples, and input conditions. For each room, we apply a corresponding prompt that takes the room type and dimensions as input to generate the furniture layout. Subsequently, similar to the approach used in Holodeck [19], we retrieve and place the most semantically

Instruction

Synthesize the 3D layout of an indoor scene based on the type and size of the room. The generated 3D layout should follow the CSS style, where each line starts with the furniture category and is followed by the 3D size, orientation, absolute position, and a short description. The description is for the appearance and style of this object. Make sure that no objects overlap or collide with each other in the layout.

Formally, each line should follow the template:
FURNITURE {length: ?px; width: ?px; height: ?px; left: ?px; top: ?px; depth: ?px; orientation: ? degrees; description: "..."}
All values are in pixels but the orientation angle is in degrees.

Example

Condition:
Room Type: bedroom
Room Size: max length 556px, max width 351px
Layout:
DESK {length: 100px; width: 50px; height: 75px; left: 50px; top: 250px; depth: 75px; orientation: 90 degrees; description: "Compact oak study desk with a small drawer and clean modern lines."}
• • •
CHAIR {length: 45px; width: 45px; height: 90px; left: 160px; top: 260px; depth: 90px; orientation: 90 degrees; description: "Ergonomic desk chair with a mesh back and cushioned seat in black."}
• • •

Condition:
Room Type: bedroom
Room Size: max length 100px, max width 150px
Layout:
BED {length: 90px; width: 60px; height: 60px; left: 5px; top: 45px; depth: 60px; orientation: 0 degrees; description: "Small minimalist bed with a light wooden frame and white bedding."}
NIGHTSTAND {length: 30px; width: 30px; height: 40px; left: 65px; top: 45px; depth: 40px; orientation: 90 degrees; description: "Compact nightstand with a single open shelf, in white."}

Condition:
Room Type: bedroom
Room Size: max length 120px, max width 160px
Layout:

Figure 2. Prompt for generating 3D assets using LLM.

Instruction

Generate a valid connectivity graph for a building floorplan, where rooms are connected based on predefined architectural rules. Each room has a specific type represented by an integer ID from the provided dictionary. The goal is to create a graph where nodes represent rooms and edges represent direct connections (doorways or open passageways) between them. Follow the constraints strictly to ensure realistic and plausible indoor layouts.

Output Format:
nodes = [room_type_1, room_type_2, ..., room_type_n]
edges = [[i, j], [k, l], ...]

Constraints:
Two Kitchens cannot directly connect.
• • •
Corridors can connect to each other.

Room Class Dictionary:
{"Bedroom":1, "Livingroom":2, "Kitchen":3, "Dining":4,'Corridor':5, 'Stairs':6, 'Storeroom':7, 'Bathroom':8, 'Balcony':9}

Example

Condition:
The scene has 4 bedrooms, 2 livingrooms, 2 kitchens, 2 corridors, 1 staircase, 4 bathrooms, 2 balcony.
Graph:
nodes =[1, 1, 1, 1, 2, 2, 3, 3, 5, 5, 6, 8, 8, 8, 8, 9, 9]
edges = [[0, 4], [1, 4], [2, 5], • • •, [9, 4], [9, 10]]
• • •

Condition:
The scene has 7 bedrooms, 2 kitchens, 1 corridor, 1 staircase, 2 bathrooms, 2 balcony.
Graph:
nodes = [1, 1, 1, 1, 1, 1, 1, 3, 3, 5, 6, 8, 8, 9, 9]
edges = [[0, 9], [1, 9], [2, 9], • • •, [6, 10], [9, 10]]

Condition:
The scene has 4 bedrooms, 2 livingrooms, 2 kitchens, 2 corridors, 1 staircase, 4 bathrooms, 2 balcony.
Graph:

Figure 3. Prompt for generating graph from text descriptions using LLM.

relevant 3D assets from a 3D asset library by computing CLIP-based similarity using asset names and descriptions.

B. Text-to-Graph Generation Details

Directly specifying a scene graph, particularly in large-scale scenes, is often impractical. To address this, we introduce an optional pipeline during the inference stage that allows the scene graph to be generated from a simplified textual description using an LLM. Users only need to provide high-level parameters such as room types and their counts, while the LLM infers the room connectivity based on statistical and architectural priors to produce a structured scene graph. This significantly reduces the burden of manual input. An example prompt is shown in Figure 3.

The prompt defines the generation task, the expected output format, and a set of optional constraints, such as preventing direct connections between kitchens or allowing corridors to interconnect. A room-type dictionary and multiple illustrative examples are also provided to guide the model. Based on the given room composition, the LLM generates the corresponding list of room nodes and connection edges, ensuring a plausible and functionally coherent

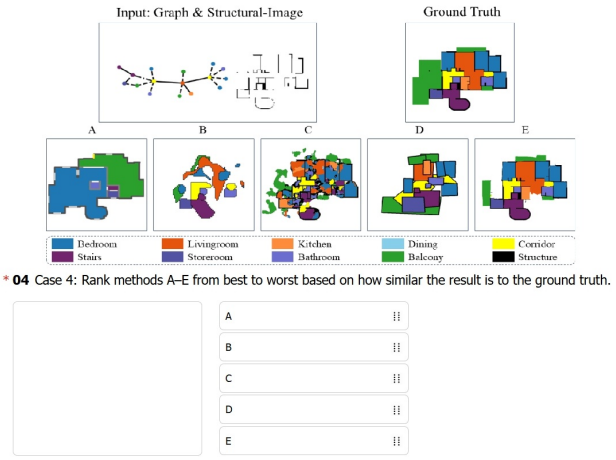

Figure 4. Illustration of the user study interface used in our experiment.

layout.

This approach greatly reduces the manual effort required to construct scene graphs and in most cases produces structurally sound results suitable as input for downstream tasks

related to floorplan generation.

C. User Study Details

The user study interface is illustrated in Figure 4. Specifically, we evaluated three criteria: Ground Truth Similarity, Consistency, and Real-World Similarity. For each criterion, participants were shown five comparison cases. In each case, they were asked to rank five options (A to E) from best to worst. We then computed the proportion of times each method was ranked first under each criterion.

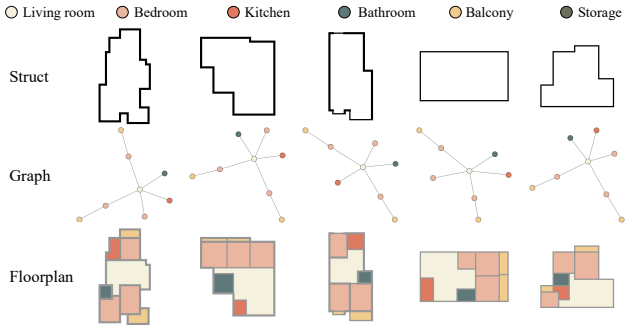

Figure 5. Example floorplan results generated by our method on the RPLAN dataset. The method produces reasonable layouts even in small-scale scenes.

We collected a total of 88 responses from 30 female and 58 male participants via an online survey. Among them, 83 were between the ages of 18 and 40, 4 were between 40 and 60, and 1 participant was over 60. The results show that our method was the most preferred overall, achieving a first-place ranking in over 90% of the comparisons under each criterion.

D. Small-Scale Floorplan Results

To further verify the effectiveness of our method, we conducted training and testing on the small-scale floorplan dataset RPLAN [18]. We used exterior walls merged with Front doors as the structural image. Our approach to extracting the topology graph differs slightly from GSDiff [7]: while GSDiff connects rooms that share a wall, we define an edge between two rooms only if they are connected by a door.

The testing results of our method are shown in Figure 5. The results demonstrate that our method remains effective on small-scale scenes, producing reasonable floorplans.

Although our input differs slightly from that of GSDiff, the comparison results are shown in Figure 6. The results for House-GAN++ [11], HouseDiffusion [14], and GSDiffff [7] are taken directly from the GSDiff paper. We can see that our method can also generate relatively reasonable results. Overall, for the RPLAN dataset, all methods perform quite well, and there is no obvious difference among them. Due to time constraints, we did not re-run the baselines ourselves.

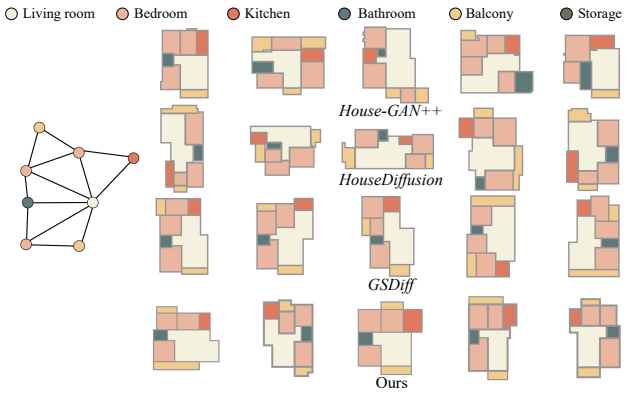

Figure 6. Comparison of floorplan generation methods on the RPLAN dataset. Results for House-GAN++, HouseDiffusion, and GSDiff are taken from the GSDiff paper. Our method also generates relatively reasonable results.

E. Additional Experimental Details

E.1. Additional Implementation Details

For completeness, we further provide several implementation specifications not covered in the main text. The Graph Transformer contains $L = 10$ layers with $H = 8$ attention heads in each layer. The floorplan output predicts $\mathcal{C} = 11$ categories (10 types of rooms plus background), while the door–window layout branch predicts $\mathcal{C}_{dw} = 4$ classes (two types of doors, one type of window, and background). The diffusion transformer backbone consists of 24 layers, and graph information is injected via cross-attention every $M_s = 6$ layers. Training the VQ-VAE component takes approximately one day on 4×RTX 4090 GPUs, while training the diffusion-based generation model requires about two days. At inference time, producing a single floorplan sample takes roughly 30 seconds.

E.2. Baselines and Training Protocols

In selecting baseline methods for comparison, we prioritize state-of-the-art approaches that are widely adopted in the literature. While several recent works such as Cons2Plan [5], Graph Transformer GANs [16], Complex Layout Generation [10], and the architecture-oriented method [21] are highly relevant, these methods are not publicly available and thus could not be included in our experiments.

For text-conditioned floorplan generation, existing methods such as Tell2Design [9], ChatHouseDiffusion [12], and Intelligent Home 3D [1] are limited to small-scale inputs (typically fewer than 10 rooms), making them unsuitable for evaluating performance on large-scale floorplans with

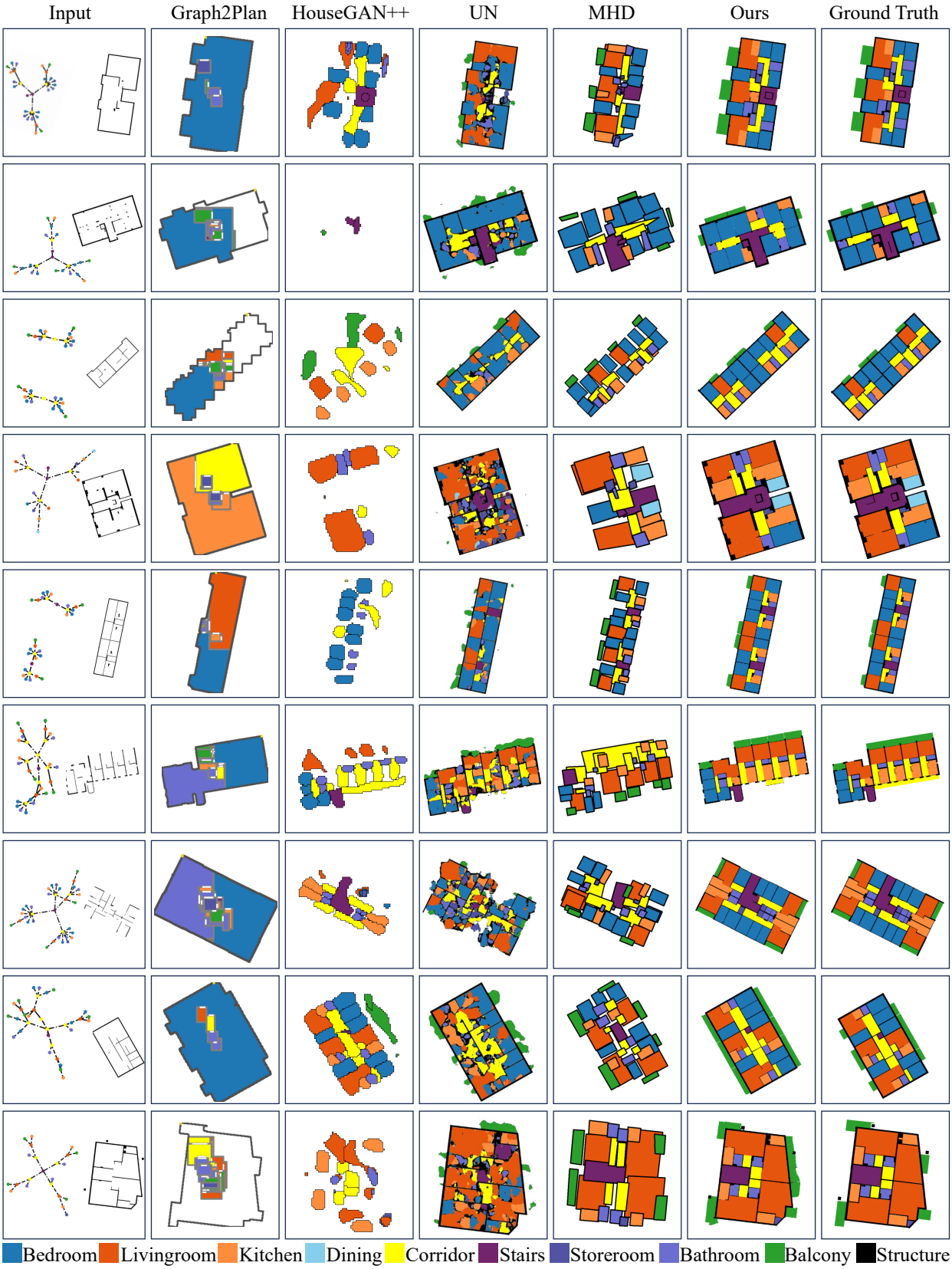

Figure 7. Additional Graph-Conditioned Floorplan Results.

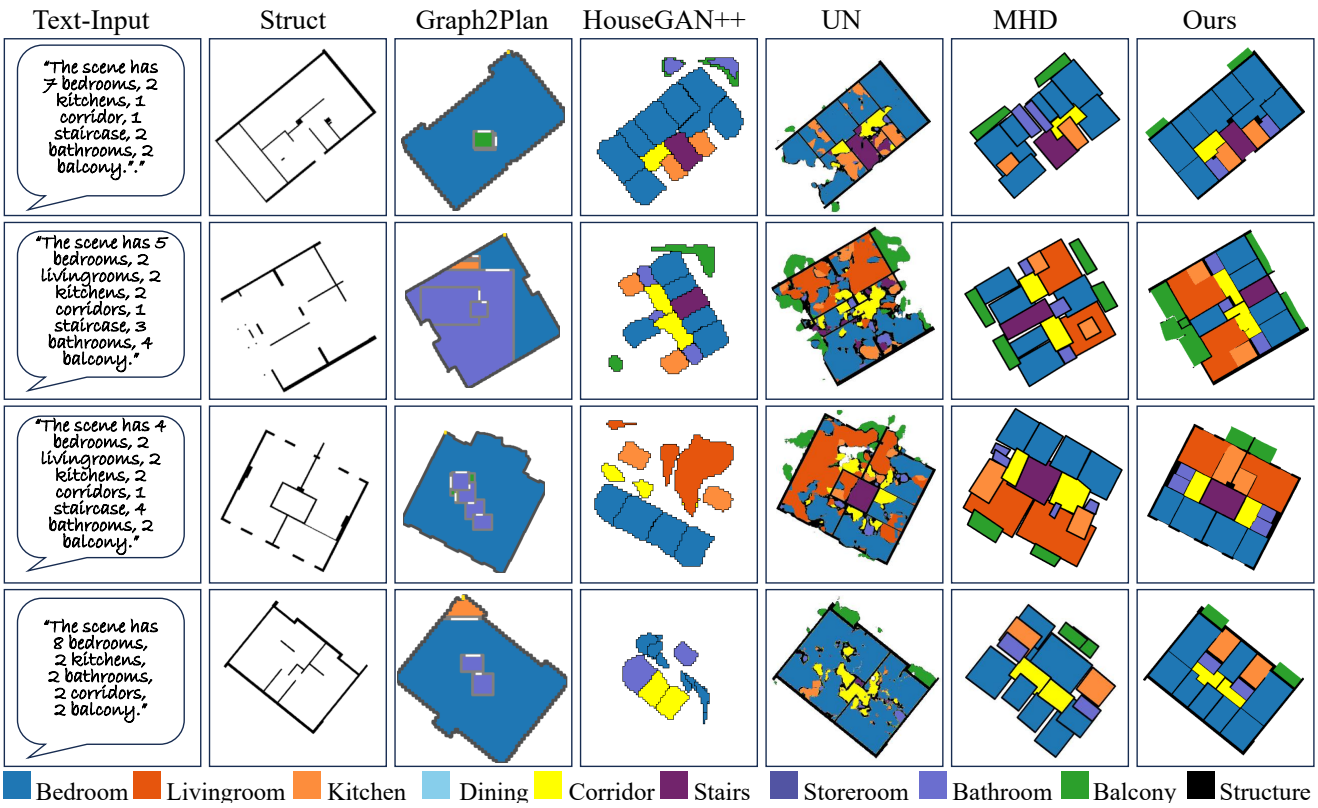

Figure 8. Additional Text-Conditioned Floorplan Results.

multiple rooms. More recent approaches like HouseLLM (HouseTune) [22] and FloorPlan-DeepSeek [20] have not released code or models, and thus cannot be included as baselines.

Therefore, we adopt two official baselines from the MSD benchmark: UN and MHD, where MHD is a modified version of HouseDiffusion [14] tailored for MSD settings. Additionally, we include two widely used graph-to-layout methods, Graph2Plan [6] and HouseGAN++ [11], as representative and reproducible baselines.

For a fair and unified comparison, UN and MHD are implemented following the official settings provided in the MSD benchmark, with a minor adaptation: the original region-type connectivity graph is replaced by a room-type connectivity graph to align with our task definition. As for Graph2Plan and HouseGAN++, we preprocess the MSD dataset into the required formats for each method, including connectivity graphs, adjacency matrices, bounding shapes, etc. All four baselines are trained from scratch until convergence under their respective default training protocols.

All baselines are further extended to a unified two-input setting (scene graph and structure image) for fair comparison. For methods requiring mask-based inputs, the structure image is converted into the corresponding representations. All implementations follow official codebases and recommended training pipelines, and are retrained on the MSD dataset.

We observe that HouseGAN++ performs relatively worse on MSD, which is likely due to its representation being designed for small-scale floorplans. The long-tailed distribution of room counts in MSD introduces significant padding in adjacency matrices, leading to less efficient optimization.

E.3. Evaluation Metric

Due to the high computational cost of certain graph compatibility metrics, such as Graph Edit Distance (GED), especially when evaluating graphs with dozens of edges.

We use the graph compatibility metric introduced in the MSD [17] dataset to evaluate the consistency between the predicted and ground truth floorplans. This metric compares the structure of two graphs: the ground truth room graph, which reflects actual connectivity based on access (typically defined by door locations), and the predicted adjacency graph, which is constructed algorithmically based on spatial proximity.

Since some baseline methods [8] do not support door prediction, and to ensure a consistent evaluation protocol, we extract the adjacency graph from the predicted room layout for all methods. Specifically, following the MSD setup,

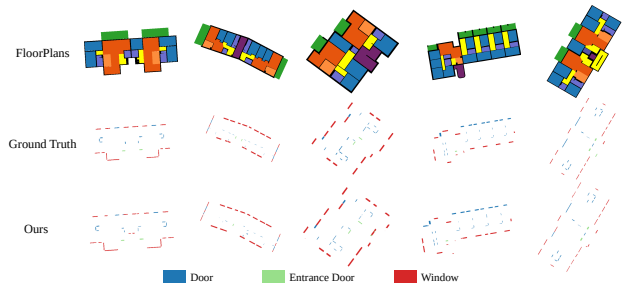

Figure 9. Door-Window Layout Results.

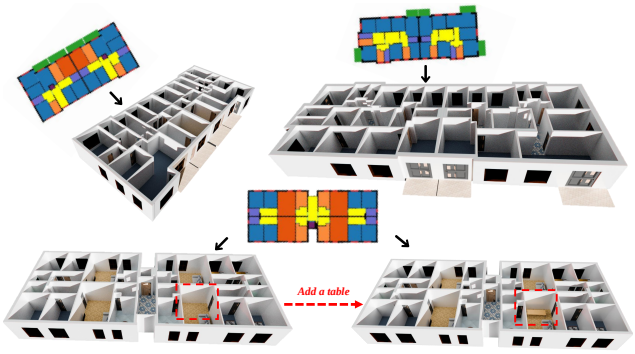

Figure 10. Additional 3D Scene Results.

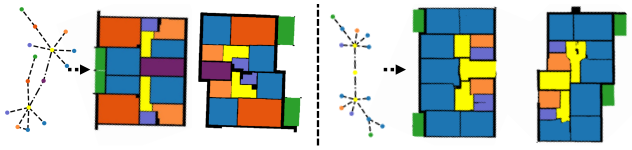

Figure 11. Floorplans generated using only the graph input.

we add an edge between two rooms if the minimum distance between them is below a predefined threshold, referred to as the buffer distance (set to 5 for 512×512 images). The graph compatibility score is computed as the ratio of the number of ground truth edges that also appear in the predicted adjacency graph to the total number of edges in the ground truth graph. This provides a quantitative measure of how well the predicted spatial relationships align with the actual room connectivity.

Therefore, if the evaluated method cannot clearly match the generated rooms to the nodes in the input graph, this metric cannot be computed.

F. Additional Results

F.1. Additional Floorplan Results

Figure 7 and Figure 8 present additional visual results of our method on 2D floorplan generation, showing the model's ability to generalize in terms of structural coherence and spatial layout.

Figure 12. Comparison between Infinigen Indoors [13] and our method

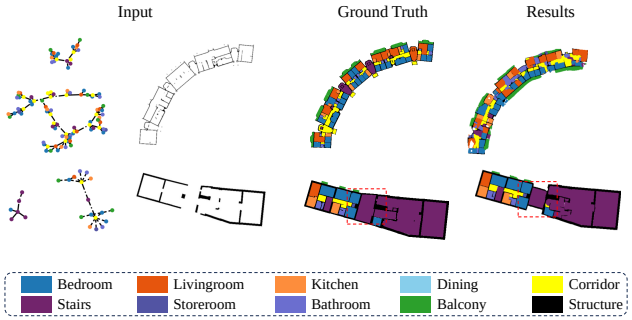

Figure 13. Failure cases.

F.2. Door-Window Layout Results

Our method introduces a separate U-Net model to decouple the generation of doors and windows from the overall floorplan generation process. This decoupling makes the prediction of door and window layouts more flexible and supports the downstream task of 3D scene construction. As shown in Figure 9, the U-Net is able to generate door and window layouts that match the given floorplan. Although some artifacts may appear in the results, these can be removed with simple post-processing rules. For example, a straightforward rule is that doors and windows should only appear along walls.

F.3. Scene Graph Only Input

The use of a structure image aligns with practical floorplan design workflows, where layouts typically start from coarse structural outlines. The structure image provides weak constraints, such as exterior boundaries or load-bearing walls, without specifying detailed room geometry, and is easy to obtain. In addition, our method can also generate floorplans using only the scene graph input after a fine-tuning stage, as shown in Figure 11.

F.4. Comparison with Rule-based Generation

Rule-based methods such as Infinigen Indoors [13] can produce diverse floorplan layouts in certain scenarios. As illustrated in Figure 12, we provide a qualitative comparison between rule-based and learning-based approaches. We observe that rule-based methods tend to generate diverse layouts but often require longer generation time, while learning-based methods better capture data-driven spatial priors from real-world datasets. Exploring the use of rule-based systems such as Infinigen as a complementary train-

ing source for floorplan generation is an interesting direction for future work.

F.5. Additional 3D Scene Results

Our method also provides a simple framework to convert the generated floorplan into a 3D scene, making it useful for various downstream applications. Figure 10 shows the 3D scene constructed from our pipeline, where we use a large language model to support semantic editing of furniture in the scene. This demonstrates that our system not only has strong capabilities in structural generation, but also supports extensible editing.

G. Limitations

Our method still has limitations when generating large-scale floorplans. As shown in Figure 13, the model struggles in scenes with a very large number of rooms (e.g., around 100), leading to structural confusion such as misconnected rooms and overlapping areas. In addition, our method sometimes fails to accurately fit rare or unusual room shapes in the MSD [17] dataset. These issues may be partly caused by the limited size of the training data. Although MSD is currently the only large-scale floorplan dataset, it contains only about 5,000 samples, which restricts the model's ability to learn complex structures.

In future work, we plan to integrate optimization techniques to improve the model's robustness in long-tail scenarios and its ability to handle uncommon structures. We also hope that larger and more diverse floorplan datasets will be developed by the research community to further support this line of work.

References

[1] Qi Chen, Qi Wu, Rui Tang, Yuhan Wang, Shuai Wang, and Mingkui Tan. Intelligent home 3d: Automatic 3d-house design from linguistic descriptions only. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 12625–12634, 2020. 3

[2] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsanit, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2023. 1

[3] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica*, 10(2):112–122, 1973. 1

[4] Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[5] Shibo Hong, Xuhong Zhang, Tianyu Du, Sheng Cheng, Xun Wang, and Jianwei Yin. Cons2plan: Vector floorplan generation from various conditions via a learning framework based on conditional diffusion models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3248–3256, 2024. 3

[6] Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver Van Kaick, Hao Zhang, and Hui Huang. Graph2plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics (TOG)*, 39(4):118:1–118:14, 2020. 5

[7] Sizhe Hu, Wenming Wu, Yuntao Wang, Benzhu Xu, and Liping Zheng. Gsdiff: Synthesizing vector floorplans via geometry-enhanced structural graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1–10, 2025. 3

[8] Yuntae Jeon, Dai Quoc Tran, and Seunghee Park. Skip-connected neural networks with layout graphs for floor plan auto-generation. *arXiv preprint arXiv:2309.13881*, 2023. 5

[9] Sicong Leng, Yang Zhou, Mohammed Haroon Dupty, Wee Sun Lee, Sam Joyce, and Wei Lu. Tell2Design: A dataset for language-guided floor plan generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 14680–14697, 2023. 3

[10] Zhengyang Lu, Yifan Li, and Feng Wang. Complex layout generation for large-scale floor plans via deep edge-aware gnns: Complex layout generation for... *Applied Intelligence*, 55(6), 2025. 3

[11] Nelson Nauata, Sepidehsadat Hosseini, Kai-Hung Chang, Hang Chu, Chin-Yi Cheng, and Yasutaka Furukawa. House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13632–13641, 2021. 3, 5

[12] Sizhong Qin, Chengyu He, Qiaoyun Chen, Sen Yang, Wenjie Liao, Yi Gu, and Xinzheng Lu. Chathousediffusion: Prompt-guided generation and editing of floor plans. *arXiv preprint arXiv:2410.11908*, 2024. 3

[13] Alexander Raistrick, Lingjie Mei, Karhan Kayan, David Yan, Yiming Zuo, Beining Han, Hongyu Wen, Meenal Parakh, Stamatis Alexandropoulos, Lahav Lipson, Zeyu Ma, and Jia Deng. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21783–21794, 2024. 6

[14] Mohammad Amin Shabani, Sepidehsadat Hosseini, and Yasutaka Furukawa. Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5466–5475, 2023. 3, 5

[15] Satoshi Suzuki and Keiichia Be. Topological structural analysis of digitized binary images by border following. *Computer Vision Graphics & Image Processing*, 30(1):32–46, 1985. 1

[16] Hao Tang, Ling Shao, Nicu Sebe, and Luc Van Gool. Graph transformer gans with graph masked modeling for architec-

tural layout generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(6):4298–4313, 2024. 3

[17] Casper Van Engelenburg, Fatemeh Mostafavi, Emanuel Kuhn, Yuntae Jeon, Michael Franzen, Matthias Standfest, Jan van Gemert, and Seyran Khademi. Msd: A benchmark dataset for floor plan generation of building complexes. In *European Conference on Computer Vision*, pages 60–75. Springer, 2024. 5, 7

[18] Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, and Ligang Liu. Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019. 3

[19] Yue Yang, Fan-Yun Sun, Luca Weihs, Eli VanderBilt, Alvaro Herrasti, Winson Han, Jiajun Wu, Nick Haber, Ranjay Krishna, Lingjie Liu, Chris Callison-Burch, Mark Yatskar, Aniruddha Kembhavi, and Christopher Clark. Holodeck: Language guided generation of 3d embodied ai environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16227–16237, 2024. 1

[20] Jun Yin, Pengyu Zeng, Jing Zhong, Peilin Li, Miao Zhang, Ran Luo, and Shuai Lu. Floorplan-deepseek (fpds): A multimodal approach to floorplan generation using vector-based next room prediction. *arXiv preprint arXiv:2506.21562*, 2025. 5

[21] Haolan Zhang and Ruichuan Zhang. Generating accessible multi-occupancy floor plans with fine-grained control using a diffusion model. *Automation in Construction*, 177:106332, 2025. 3

[22] Ziyang Zong, Guanying Chen, Zhaohuan Zhan, Fengcheng Yu, and Guang Tan. Housetune: Two-stage floorplan generation with llm assistance. *arXiv preprint arXiv:2411.12279*, 2025. 5