EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Floorplan Generation by Alternating Geometry and Semantics Optimization

Wenming Wu[1,2] Sizhe Hu[1] Ligang Liu[3] Liping Zheng[4,1] † Xiao-Ming Fu[3]

[1]Hefei University of Technology, China
[2] Anhui Province Key Laboratory of Industry Safety and Emergency Technology (Hefei University of Technology), China
[3]University of Science and Technology of China, China
[4]Anhui University of Technology, China

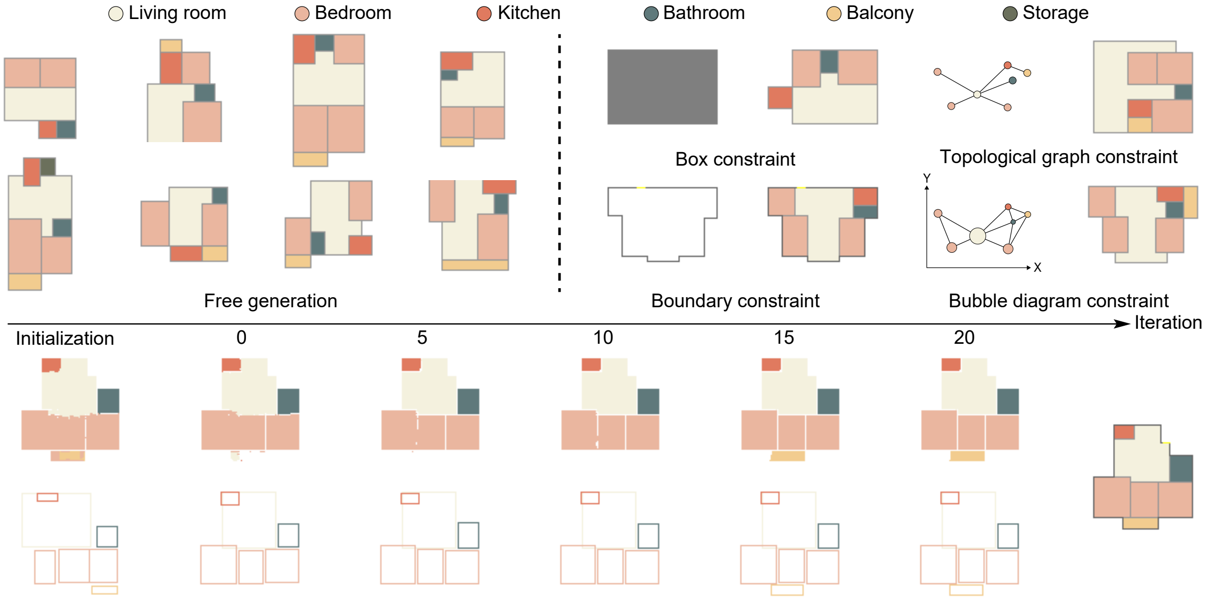


Figure 1: *Our method generates vector floorplans either unconstrained, enabling free exploration of diverse designs, or constrained, ensuring specific conditions are satisfied. A detailed optimization process for one generated floorplan under the boundary constraint, showing the semantics maps and room layouts. The semantics optimization and geometry optimization are achieved by refining each other progressively.*

Abstract

Creating floorplans lays the foundation for architectural design and scene modeling. We propose a novel framework for generating diverse high-quality floorplans under predefined constraints. Central to our method is an iterative refinement process for optimizing the bounding boxes of rooms and the floorplan semantics image, which defines a vector floorplan together. Vector floorplans can be generated through a learning-based refinement process. Our framework supports various constraints, such as floorplan boundaries, topological graphs, and bubble diagrams. Extensive experiments demonstrate that our method is superior to state-of-the-art techniques, particularly in generating a wider variety of solutions that cater to various architectural needs.

CCS Concepts

• ***Applied computing*** → ***Computer-aided design;*** • ***Computing methodologies*** → ***Shape modeling;***

1. Introduction

Generating floorplans is crucial in architectural design and scene modeling. It involves determining the layout of rooms, walls, and

† The corresponding author: zhenglp@hfut.edu.cn (Liping Zheng)

other architectural elements while adhering to design constraints. Creating such vector floorplans with reasonable structure and function is inherently challenging. It requires careful consideration of factors such as room size, shape, function, and logical arrangement, balancing aesthetics, user needs, and practical constraints.

To address these challenges, recent advances in floorplan generation have been largely driven by deep learning techniques. Some methods [WFT*19; NCC*20; NHC*21] focus on generating rough floorplan images, which are then refined through post-processing. These image-based methods, however, typically fail to directly generate realistic vector representations of floorplans, which are required for practical applications in architecture and design. More recent works [HHT*20; SWL*22; SHF23] have sought to improve floorplan generation by incorporating structured representations, such as wall graphs [SWL*22] and room polygons [SHF23]. These methods transform structured representations into floorplans, providing more control over the generated layouts and improving their spatial coherence. However, despite these advancements, several limitations remain in existing methods.

Existing methods typically can not directly generate aesthetically pleasing floorplans, necessitating an additional optimization phase. In practice, the optimization usually relies on heuristic metrics and parameters, demanding parameter tuning or complicated optimization. Although parameter tuning or complicated optimization is used, their output leaves room for improvement in global layout rationality and local semantics accuracy. Moreover, the generation result lacks diversity, failing to offer users a variety of choices. *ATISS* [PKS*21] has shown promise in room layout generation using deep learning. It struggles to produce complete and precise vector floorplans. Incorporating CNN-based components to improve structural accuracy is a promising strategy. However, only treating refinement as a separate task or post-processing limits the synergy between layout generation and semantic refinement. This disconnect fails to fully address the interdependent challenges of generating coherent and functional floorplans.

This paper proposes a novel method for generating vector floorplans that are both functional and capable of meeting specific design constraints. We adopt a simple structure–semantic representation of floorplans, where a floorplan is decomposed into two key components: (1) a room layout, representing the geometric arrangement of rooms, and (2) a semantics map, detailing the types of rooms. This formulation converts the floorplan generation problem into the creation of these two components, enabling various constrained generations by controlling both the room layout and the semantics map. Our approach begins with a deep generative model to produce diverse initial room layouts. Then the room layouts and semantics maps are refined alternately through an iterative optimization. The reasons are twofold. First, end-to-end generating the room layout and semantics map is challenging. Second, the two components can promote each other. Room layouts provide global geometric guidance to structure the semantics map, while semantics maps offer localized labeling to refine the precision of room layouts. This interplay enhances the accuracy and organization of both components in the design process.

We have two observations: (1) determining the semantics map from the room layout is pixel-wise labeling, which can be regarded as an image segmentation problem; (2) computing the room layout from a semantics map is to detect the objects (i.e., rooms) in the labeled images. It is an object detection problem. Thus, our method can be implemented with existing elegant neural networks, increasing the algorithm's robustness and significantly reducing the implementation difficulty. Vector floorplans are obtained via a learning-based refinement process, rather than heuristic, threshold-based post-processing or explicit optimization, as shown in Figure 1. The semantics optimization can be viewed as a local optimization tool, converting the rough room layout into a more accurate semantics map to obtain detailed room shapes and establish specific room boundaries. The semantics map of the individual room gradually becomes regular and neat, and the jaggedness of the semantic pixels gradually decreases. Meanwhile, the semantics map also provides more accurate room locations and more reasonable room sizes for improving the room layout. The geometry optimization can therefore be considered a form of global optimization, gradually adjusting the size and position of rooms, and even merging and decomposing rooms, to match the semantics map from a global perspective. Our process differs from traditional optimization-based approaches. Instead of explicit gradient-based or heuristic optimization, we use neural networks to implicitly optimize geometry-to-semantics and semantics-to-geometry through forward predictions from the previous geometry and semantics, while maintaining the original geometric and semantic information.

Our method supports constrained floorplan generation under various constraints, such as bounding boxes, topological graphs, floorplan boundaries, and bubble diagrams (Figure 1). Extensive experiments, including qualitative and quantitative analyses, ablation studies, and perceptual evaluations, have been conducted to compare our method against state-of-the-art techniques. Our contribution lies in integrating standard deep learning components within a unified framework that reformulates floorplan generation as an alternating geometry–semantics optimization problem to address a complex generative task. Unlike prior approaches that rely on heuristic or threshold-based post-processing, our method formulates refinement as a learning-based, iterative process in which geometry and semantics mutually reinforce each other.

2. Related work

2.1. Layout generation in computer graphics

In the community of computer graphics, layout generation generally means arranging elements in a space or on a canvas in a visually pleasing and functional manner. Different design elements correspond to distinct tasks of layout generation, including urban planning [PYW14; PYB*16], indoor scene synthesis [PKS*21; TNM*23], and graphic design [GLA*21; IKS*23]. Our work focuses on generating vector floorplans. Floorplan generation is a significant research problem in layout generation, designing the layout of rooms, spaces, and other elements within a building or structure.

2.2. Optimization-based Floorplan generation

Given high-level requirements, [MSK10] integrates machine learning with architectural principles and stochastic optimization techniques to translate bubble diagrams into detailed floorplans. [LYAM13] presents an interactive floorplan design tool to conform to functional constraints, design considerations, and fabrication constraints. [WFLW18] introduces a coarse-to-fine framework for generating building interiors with mixed integer quadratic programming. *G2PLAN* [BSU*22] is an automated floorplan generation method based on graph theory and linear optimization techniques. These methods rely on strict constraint modeling and complex, parameter-sensitive optimization, whereas our approach replaces such heuristics with a learning-based refinement framework.

2.3. Deep predictive model for floorplan generation

Unlike early work using procedural or optimization methods, current research trends explore deep learning techniques, especially using deep predictive models for floorplan generation. *RPLAN* [WFT*19] presents a two-stage framework to obtain visually coherent floorplans. However, *RPLAN* relies on heuristic post-processing to convert predicted semantic maps into vector floorplans. *Graph2Plan* [HHT*20] is a framework to convert bubble diagrams to floorplans. However, the output of *Graph2Plan* is a set of discrete room boxes, which require post-processing, such as alignment, to obtain more plausible floorplans. [CWT*20] proposes to create floorplans and 3D houses from linguistic descriptions. Room boxes are generated using neural networks to generate floorplan, similar to *Graph2Plan*. However, post-processing is inevitable. *WallPlan* [SWL*22] introduces a wall-oriented method where the wall graph generation process involves post-processing operations. However, the graph generation process within *WallPlan* still requires heuristic post-processing operations. Another key limitation of existing methods is their single deterministic output with limited diversity. Our framework integrates predictive precision with generative diversity, producing high-quality, flexible, and adaptable vector floorplans through a learning-based refinement process rather than heuristic, threshold-based post-processing. MaskPLAN [ZSD24] enables user-guided floorplan synthesis from partial inputs via graph-structured masked autoencoding, but focuses on attribute completion rather than explicit geometric refinement. Some methods [LWKF17; CLWF19] attempt to reconstruct vector floorplans from rasterized images through semantic segmentation. These methods heavily depend on additional post-processing and optimizations to obtain the structure and improve recognition accuracy. It is worth noting that such approaches are not generative methods but rather focus on recognition tasks.

2.4. Deep generative model for floorplan generation

The generative adversarial network (GAN) was first introduced in exploring multi-floorplan generation [Cha20]. However, the generated results suffer from semantic inconsistencies, jagged edges, and disorganized structural information, undermining the practicality of the generated floorplans. Subsequently, the *House-GAN* series [NCC*20; NHC*21] has emerged as the representative work in this field. These methods take a topological graph as input to generate floorplans, but have similar limitations inherent to GAN-based models [Cha20]. Further post-processing is still required since the generated floorplan is represented as a raster image. *Floorplan-GAN* [LH22] uses GANs with differentiable rendering to generate vector-based floorplans, but requires an area table, increasing user burden. Its outputs, a set of room boxes, often face alignment issues, overlaps, and lack a defined room order. Moreover, it relies on heuristic post-processing, whereas our method formulates refinement as a learning-based post-processing stage. [TSSV24] learns graph-structured architectural layouts using a Transformer-based GAN with masked modeling, but it remains graph-centric and does not directly generate or refine vectorized floorplans. Recently, *HouseDiffusion* [SHF23] introduces a diffusion model for floorplan generation, representing rooms and doors as polygons. However, it requires predefined room shape topologies (the number of polygon edges) that remain fixed, which often results in misaligned polygons, as the model does not dynamically adapt the topology to better fit the desired layout. [PGK*21] proposes to generate floorplan layouts using a Transformer-based generative model. However, additional layout optimization is still needed to implement the restrictions. In contrast, our method generates room layouts and semantic maps under design constraints and refines them through a unified, learning-based post-processing framework, rather than heuristic, threshold-based optimization. This yields structured outputs with strong spatial coherence and semantic accuracy under diverse design constraints.

3. Method

3.1. Overview

Given the design constraints as inputs, such as the architectural boundary and bubble diagram, we aim to generate a set of plausible vector floorplans (Figure 2). For simplicity in description, we will only introduce the architectural boundaries with a front door as input, with other constraints introduced in Section 4. A floorplan $\mathcal{F}$ can be decomposed into two components (Figure 3): (1) a semantics map $\mathcal{I} \in \mathcal{R}^{H\times W}$ (H and W denote the resolution of the floorplan image), which uses image pixels to describe the semantic categories of elements within the floorplan, and (2) a room layout $\mathcal{L} = \{r_i\}_{i=1}^{M} = \{(l_i, x_i, y_i, w_i, h_i)\}_{i=1}^{M}$, which represents the geometric arrangement of floorplan elements using a set of bounding boxes. Here, l_i is the label of the room category, (x_i, y_i) denotes the center of the room box, and w_i and h_i represent its width and height. The task of floorplan generation can be converted to creating $\mathcal{I}$ and $\mathcal{L}$ under constraints. Once obtaining $\mathcal{I}$ and $\mathcal{L}$, we can easily generate the vector floorplan $\mathcal{F}$ as follows: (1) use $\mathcal{I}$ to determine room labels for overlapping regions of boxes; (2) extract the regions with the same label to form the final room.

Key idea. We observe that deriving the semantics map $\mathcal{I}$ from the room layout $\mathcal{L}$ resembles image segmentation, while extracting the room layout $\mathcal{L}$ from the semantics map $\mathcal{I}$ parallels object detection. These complementary processes enhance each other through iterative refinement. Room layouts can globally guide the generation of semantics maps by offering rough floorplan geometry. This guidance helps create more structured and coherent semantics maps where each space's general function and category are delineated. Conversely, semantics maps contribute to the refinement of room layouts by providing detailed area labeling that serves as local guidance. According to the semantic indications, the specific dimensions and arrangement of room boxes are optimally placed. Therefore, our key idea is first to generate a wide variety of initial room layouts, and then use neural networks to optimize the

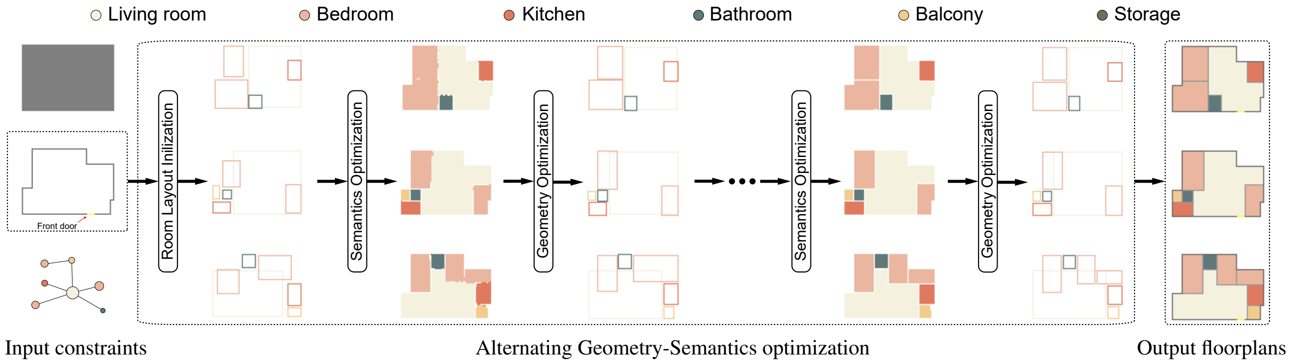


Figure 2: *Overview. Our method automatically generates multiple vector floorplans based on design constraints like architectural boundaries. We show the generation process of alternating Geometry-Semantics optimization, with real results of generated floorplans. The semantics network optimizes the semantics map, while the geometry network refines the floorplan's geometry. This process continuously improves the semantics map and room layout, resulting in vector floorplans derived from these maps and layouts.*

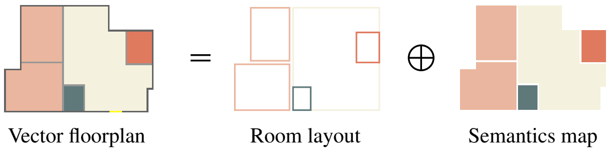


Figure 3: *Floorplan representation. A vector floorplan can be represented by a room layout and a semantics map.*

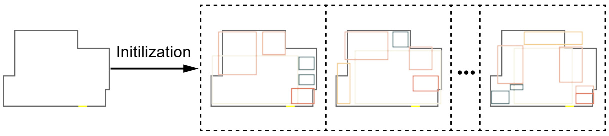


Figure 4: *Initilization. Several initial room layouts are generated, given the architectural boundary as the input constraint.*

room layouts $\mathcal{L}$ and semantics maps $\mathcal{I}$ alternately to improve them continuously. We view our refinement module as a learning-based post-processing stage that replaces hand-crafted geometric heuristics and threshold selection with data-driven refinement.

Pipeline. As shown in Figure 2, given the constraint, we first generate a wide variety of initial room layouts using a room layout generative model (Section 3.2). For each initial room layout $\mathcal{L}$, an image segmentation network is developed to predict the semantics map $\mathcal{I}$ from the room layout $\mathcal{L}$ (Section 3.3). Next, an object detection network is used to refine the room layout $\mathcal{L}$ based on the semantics map $\mathcal{I}$ (Section 3.4). We iteratively conduct these two steps until convergence. This alternating process balances the floorplan's local semantics and global geometry, continuously improving the semantics map and room layout. The floorplan extraction from the optimized semantics map and room layout is simple.

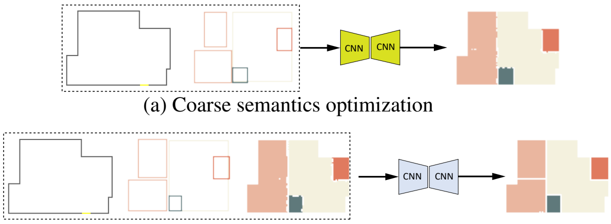
3.2. Initializing room layouts

Our framework first generates initial room layouts based on design constraints. We adopt an autoregressive approach. The generation process unfolds sequentially, and each room box is generated conditioned on the previously generated. We propose a generative model, *BoxGenerator*, which enables the acquisition of detailed attributes for room boxes with diverse possibilities. *BoxGenerator* learns the distribution of attributes for the i-th room box, represented as $\mathcal{P}(r_i \mid \mathcal{B}, r_{<i}) = \mathcal{P}(l_i, x_i, y_i, w_i, h_i \mid \mathcal{B}, r_{<i})$, where $\mathcal{B}$ is the given architectural boundary and $r_{<i} = \{r_0, r_1, ..., r_{i-1}\}$ denotes the room boxes already present in the scene. We follow the framework introduced by *ATISS* [PKS*21] to generate initial room layouts (Figure 4). See more details in the supplementary material.

Generation ordering. We adopt an autoregressive approach to generate room boxes. This raises the issue of generation ordering. For floorplan generation, the generation ordering of room boxes directly impacts the final results. For example, the placement of the living room can affect the planning of bedrooms and kitchens. We have two observations: (i) When two room boxes overlap, the smaller room box is usually placed partially or completely inside the larger room box, such as a larger living room containing a smaller kitchen. Generating larger room boxes first can provide better guidance for the placement and adjustment of smaller overlapping rooms, which ensures a coherent and realistic arrangement of rooms in the generated floorplan. (ii) The occurrence frequency of different categories of rooms varies. Room categories that appear more frequently are more likely to be generated in floorplans. For example, living rooms are almost essential in all floorplans. Hence, prioritizing the generation of high-frequency rooms helps ensure the functionality and plausibility of floorplans. In the end, we propose the following training strategy: For each training sample, we first sort the room boxes by area and then by the frequency of room occurrence in the dataset. This establishes the generation order of room boxes for each training sample.

3.3. Optimizing semantics maps

Given a room layout and the input constraints, generating a semantics map of the floorplan involves assigning each point in the floorplan generation domain the room label to which it belongs. It can be regarded as an image segmentation task.



(a) Coarse semantics optimization

(b) Fine semantics optimization

Figure 5: *Semantics optimization. (a) Coarse semantics optimization is used to obtain an initial semantics map. (b) Fine semantics optimization continuously improves the result.*

3.3.1. Network

We propose a coarse-to-fine optimization strategy for optimizing semantics maps: (1) *Coarse semantics optimization* (Figure 5 (a)): with input constraints and perturbed room layouts, we train a coarse semantics network. (2) *Fine semantics optimization* (Figure 5 (b)): given input constraints, perturbed room layouts, and the predicted semantics maps from the coarse semantics optimization step, we train a fine semantics network. A modified *ASPP* [CPK*17] is used for our semantic networks. We represent the input boundary as a (128×128) image. Therefore, the input to the coarse semantics network is a (128×128) multi-channel image (defaults to 0), including the mask image of the boundary, the front door, room boxes from the room layout, and the room center:

- *Inside-Boundary mask*: taking a value of 1 for the interior, 0.5 for the exterior walls, and 0.75 for the front door.
- *Front-door mask*: same to used in *BoxGenerator*.
- *Room-box mask*: taking a value of the room category for the room box mask.
- *Room-center mask*: taking a value of 1 for the room-center mask. Using a (10×10) square centered on the room box.

These are used as the input for the first time in semantic optimization, i.e., coarse semantics optimization. For the fine semantics optimization, in addition to these channels, the input also includes the semantics map in the last iteration:

- *Previous-map mask*: the semantics map from the last iteration.

The output is a semantic segmentation image with room labels. More details can be found in the supplementary material.

3.3.2. Training and inferring

To train the semantics network, the room layout and semantics map must be well-matched in supervised learning. However, we lack the paired room layout and semantics map to train each step of the iteration process. To address this issue, we introduce perturbations to the ground-truth room layouts (Figure 6). The ground truth semantics map and the perturbed room layouts form the pairs of training samples, reducing the difference in data distribution between training and inference. We construct the perturbation data in two steps. First, for each ground-truth room box, we expand its length and width by a factor of two while maintaining the central location. With the four corners of the ground-truth room box as centers, the enlarged room box is divided into four sub-areas. Second, the perturbation is achieved by adjusting the top-left and bottom-right vertices within their sub-areas. Alterations are achieved in central locations and room sizes for each ground-truth room box. This ensures that the perturbed room layouts vary in both position and size while remaining reasonably close to the ground truth.

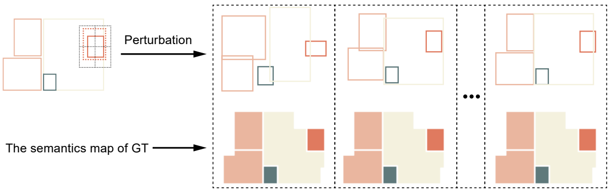


Figure 6: *Training data construction for the semantics network. We simulate iterative optimizations with varying perturbations to generate training samples of room layouts. The corresponding semantics map samples remain the ground truth.*

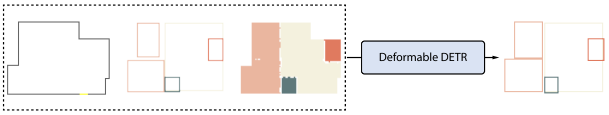


Figure 7: *Geometry optimization. During training, we apply the semantics network to obtain a semantics map as the input to the geometry network. The semantics map from the previous iteration is used as the input during inference.*

The semantics networks are trained to predict the ground-truth semantics map. Since the semantics networks perform a semantic segmentation task, we train the semantics networks using pixel-wise cross-entropy loss and Adam optimizer [KB14]. The coarse and fine semantics networks are not trained together. Given the constraints and an initial room layout, we first train the coarse semantics network to generate a semantics map. Then we train the fine semantics network, using the input constraints, the previous room layout, and the previous semantics map to produce a more accurate semantics map. The additional semantic input allows the fine semantics network to refine the coarse predictions effectively.

3.4. Optimizing room layouts

We adopt a dedicated geometry network because initial semantic maps are often noisy, and directly extracting bounding boxes can lead to unstable layouts with split or overlapping rooms, which destabilize iterative refinement. By learning geometric priors such as relative positions, typical shapes, and proportions, the geometry network produces clean and stable room boxes, enabling reliable convergence. The room layout optimization is a kind of global geometry optimization. The newly generated room layout from the geometry optimization matches the input semantics map, making it more accurate in terms of both the location and size compared to the previous room layout. Therefore, optimizing room layouts can be viewed as generating a new room layout from an updated semantics map. Given a semantics map, generating a room layout involves finding the rooms' bounding boxes and labels. This is a classic object detection task. Note that the optimization of room

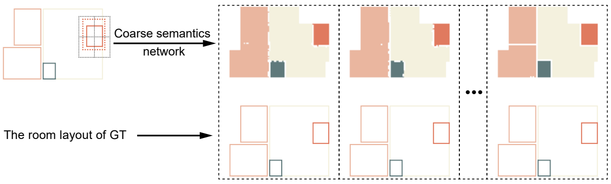


Figure 8: *Training data construction for the geometry network. We use the coarse semantics network to generate semantics maps from perturbed room layouts, thus obtaining training samples of semantics maps. The corresponding training samples of room layouts remain the ground truth.*

layouts adjusts the room boxes based on the input semantics map. This adjustment may involve adding, removing, or modifying room boxes to better align with the detected semantics regions.

3.4.1. Network

We develop a specialized *geometry network*, which extracts a global room layout from the semantics map of a floorplan based on *Deformable DETR* [ZSL*20]. The geometry network also processes the input of a 128 × 128 multi-channel image, the same as the fine semantics networks, including *Inside-Boundary mask*, *Front-door mask*, *Room-box mask*, *Room-center mask*, and *Previous-map mask*. The output is a new room layout predicted from the previous semantics map. As shown in Figure 7, given a semantics map of the floorplan, we first employ *ResNet-50* [HZRS16] as the backbone to extract multi-scale image features. These features are then fed into the Transformer encoder, which utilizes multiple layers of self-attention mechanisms and feedforward neural networks to capture contextual information within the input features, enabling a deep understanding of the input. The Transformer decoder takes a fixed number of learnable positional embeddings as input, and the output feature embedding is passed to the Transformer encoder. The Transformer decoder outputs the same number of embeddings, aggregating information from both the box queries and the image features to capture long-range dependencies among room boxes. The output embeddings can be used for room attribute prediction by passing them to the prediction heads of multiple shared feedforward networks.

3.4.2. Training and inferring

Similar to semantics networks, we need the paired semantics map and room layouts for training (Figure 8). We use the coarse semantics network (Section 3.3), which uses a perturbed room layout as input to generate a semantics map as input to the geometry network, and the paired room layout is the ground-truth room layout.

The geometry network is trained to predict the ground-truth room layout based on the input semantics map and the input constraints. Since the geometry network performs an object detection task, we train the network using the loss introduced in *Deformable DETR* [ZSL*20] and use Adam optimizer [KB14] for training. Given the constraint and the semantics map of the last iteration, we update the room layout using the geometry network.

3.5. Alternating optimization

We have introduced two optimization models focusing on semantic and geometric aspects, respectively. In the following, we link these two models to alternately optimize the room layout and semantics map, continuously improving them. The process starts with an initial room layout, refined by the semantics optimization model to adjust room labels and details. Next, the geometry optimization model refines the layout by adjusting room locations and sizes based on the updated semantics map. Figure 9 illustrates this process.

The refinement framework alternates between optimizing semantics and layout, ensuring progressive improvement through a bilevel optimization process. The combined loss function is: $Loss = \mathscr{L}_{\text{sem}}(\mathcal{I},\mathcal{L}) + \mathscr{L}_{\text{geo}}(\mathcal{I},\mathcal{L})$, where $\mathscr{L}_{\text{sem}}$ enforces semantic consistency and $\mathscr{L}_{\text{geo}}$ improves geometric accuracy. The alternating optimization iteratively minimizes $Loss$ with respect to $\mathcal{I}$ (semantics map) and $\mathcal{L}$ (room layout) as follows:

1. Fix $\mathcal{L}$ and minimize $\mathscr{L}_{\text{sem}}(\mathcal{I},\mathcal{L})$ to update $\mathcal{I}$.
2. Fix $\mathcal{I}$ and minimize $\mathscr{L}_{\text{geo}}(\mathcal{I},\mathcal{L})$ to update $\mathcal{L}$.

Semantics refinement adjusts pixel-level details to meet user constraints, while layout refinement optimizes room positions, sizes, and overlaps for structural coherence. This interplay reduces misalignments and inconsistencies by leveraging local semantic guidance and global layout adjustments. The method is robust to poor initialization. If the layout is misaligned, the semantics map guides adjustments, and if the semantics map is noisy, layout refinement enforces constraints like non-overlapping rooms and size distributions, ensuring high-quality floorplans even from suboptimal starts.

Our iterative process terminates when any of the following conditions are met: (1) The amount of (pixel) change in the semantics map before and after iteration is less than a specified N_{max}; (2) the maximum iteration number n_{max} is reached. In our experiments, we set $N_{\text{max}} = HW/10000$ and $n_{\text{max}} = 20$.

3.6. Floorplan extraction

Our alternating optimization can be viewed as a *learning-based post-processing* stage, which replaces heuristic and parameter-sensitive refinement with data-driven predictions. During inference, the framework produces two intermediate representations: a pixel-level semantics map predicted by the semantics network, and a room layout predicted by the geometry network in the form of labeled bounding boxes. To obtain vector floorplans suitable for rendering and evaluation, we apply only deterministic and lightweight regularization steps, rather than hand-crafted energy functions or threshold-sensitive heuristic rules.

Due to the probabilistic nature of neural networks, minor geometric deviations may occur in the predicted room layouts. In practice, we extract room regions from the optimized semantics map and compute their bounding boxes to obtain well-aligned room geometry. Since the semantics map is produced by a supervised segmentation network, spatial continuity and region consistency are already strongly enforced, and the prediction strictly respects architectural boundaries, eliminating the need for additional clipping or geometric correction. The semantic network uses the input boundary as an explicit spatial mask, ensuring all predictions occur within

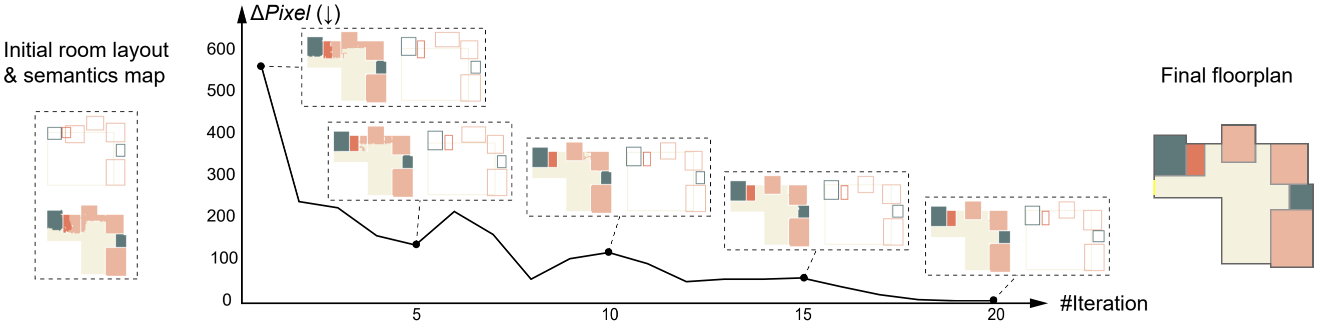


Figure 9: *The optimization process of generating one floorplan under the boundary constraint. The plot shows the average amount of pixel change in the semantics map before and after each iteration (ΔPixel). We show the results for the initial, 1st, 5th, 10th, 15th, and 20th iterations, as well as the final floorplan.*

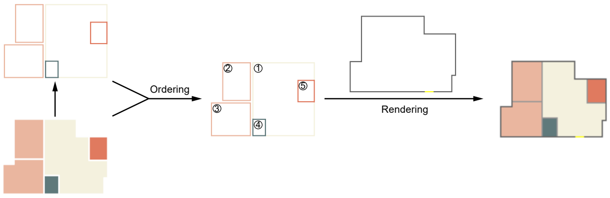


Figure 10: *Vector floorplan extraction. We determine the room box order from the semantics map and the room layout. Then the vector floorplan can be directly extracted under the boundary constraint.*

the given boundary. This structurally enforces boundary adherence, which we observe consistently in generated results. The semantics map is further used to determine the ordering of room boxes in overlapping regions to ensure structural consistency. We emphasize that our method does not explicitly perform box merging as a post-processing step. When adjacent small rooms with identical semantics are merged, this behavior arises naturally from the geometry network's object detection predictions rather than from manually designed rules. The geometry optimization follows standard object detection practice and applies several fixed thresholds (such as the confidence threshold) to filter valid room boxes, which is a conventional component rather than a tunable heuristic parameter.

4. Experiments

4.1. Implementation

We have implemented the proposed framework using PyTorch and trained our networks on an NVIDIA GeForce GTX 4090 GPU. Our framework involves multiple networks. More implementation details can be found in the supplementary material. These networks are trained separately. At test time, the alternation applies pretrained networks iteratively. Given the boundary constraints, our method takes around 0.34 seconds to generate a floorplan with 10 iterations of Geometry-Semantics optimization and 0.57 seconds with 20 iterations. We have used the *RPLAN* dataset [WFT*19] processed by *Graph2Plan* [HHT*20] for training, which contains more than 80K residential floorplans with dense annotation. We convert floorplans into a 128 × 128 semantics map with fixed wall thickness (1 pixel) for training. The data distribution for training, validation, and testing is set to 70%, 15%, and 15%, respectively.

Constrained floorplan generation. Our method is highly scalable for different design constraints. By adjusting the input according to specific restrictions during training and testing, we can achieve various constraint-based generations (Figure 11). See more details on implementation in the supplementary material. The free generation without any inputs can create floorplans with boundaries that may not have appeared in the dataset, yet maintain logical and functional room layouts. The box-constrained generation requires the generated floorplans to lie within a given box, which allows the creation of floorplans with unique boundaries but reasonable dimensions. The graph-constrained generation offers a more flexible and versatile approach to control floorplan generation. Note that nodes of the topological graph have only one attribute, the room category label. The topological graph only constrains the connectivity between rooms and does not impose any constraints on non-connectivity. In contrast, nodes in a bubble diagram include the room category label and encompass attributes like the room's central location and size. Edges denote the adjacency between rooms, which includes not only room pairs connected by doors but also spatially adjacent room pairs. It can be seen that the bubble diagram constraint provides significantly more generative control than the topological graph constraint. Although no boundary constraints are imposed, the generated room layouts and architectural boundaries are similar in almost all results. The boundary-constrained generation with a front door provides stronger control over the generation without a front door, especially for the generation of living rooms. The hybrid-constrained generation shows a stronger capability for control over the generation process, especially when bubble diagram constraints are combined with other types of constraints. This similarity suggests that the design space is significantly smaller. For floorplan box, architectural boundary, or bubble diagram, we use *ResNet* [HZRS16] to extract constraint features. For the topological graph, we use *GCNConv* [KW16] for graph encoding.

4.2. Qualitative evaluation

Figure 12 illustrates the improvements achieved through the iterative optimization process. The initial room layout consists of rough rectangular bounding boxes that often lack alignment, exhibit over-

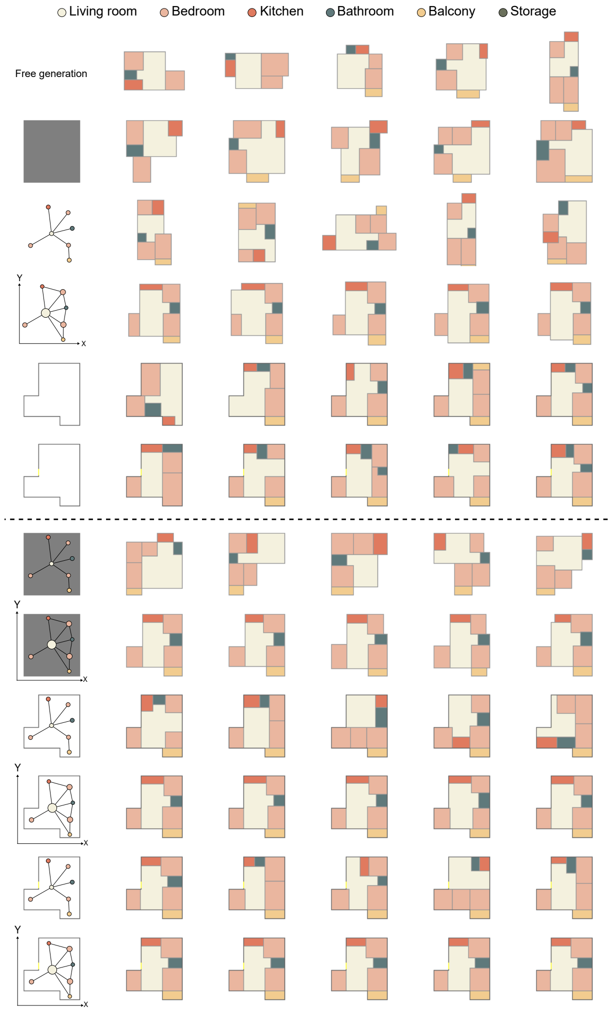


Figure 11: *Floorplan generation.* ***Upper:*** *free generation, box-constrained generation, topological-graph-constrained generation, and bubble-diagram-constrained generation.* ***Bottom:*** *six kinds of hybrid-constrained generations.*

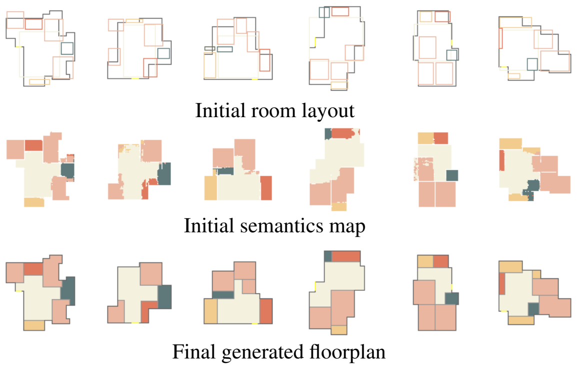


Figure 12: *The difference between the initial room box/semantics map and the final generated floorplan.*

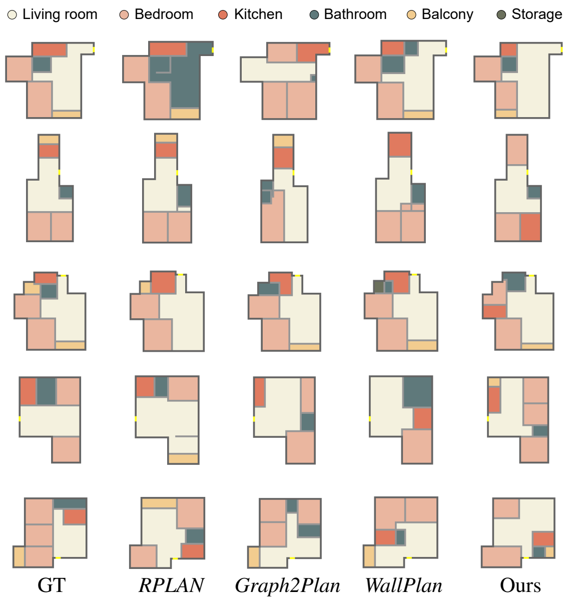


Figure 13: *Comparison to the state-of-the-art techniques on the boundary-constrained generation. Each row shows the generated floorplans by different methods applied to the same architectural boundaries. We demonstrate the results of the ground truth, RPLAN [WFT*19], Graph2Plan [HHT*20], WallPlan [SWL*22], and our method, respectively.*

laps, and fail to form coherent boundaries. The initial semantics map provides a preliminary labeling of room types but frequently shows irregular boundaries, semantic inconsistencies, and jagged edges. After iterative optimization, the final floorplans exhibit significant improvements: room layouts are more aligned and spatially coherent, with overlaps removed and boundaries refined. The semantics maps also show cleaner, more accurate room shapes and consistent labeling. While the BoxGenerator provides a good starting point, our contributions lie in the learning-based refinement beyond initialization. Our method generates high-quality floorplans even from poor initializations. While a strong initialization helps, our framework ensures consistent refinement for a superior final result. The alternating optimization makes minor adjustments at each iteration, but the iterative process allows substantial refinement over time. Even with poor initializations, the continuous interaction between geometry and semantics improves both, producing high-quality floorplans that overcome initialization limitations.

4.2.1. Boundary-constrained generation

We compare with *RPLAN* [WFT*19], *Graph2Plan* [HHT*20], and *WallPlan* [SWL*22] on the boundary-constrained generation (Figure 13). *RPLAN* generates floorplans by predicting and post-processing a semantic map. Its results show significant semantic inaccuracies, such as incorrect bathroom placements and omissions, highlighting weaknesses in its semantic map generation. *Graph2Plan* shows improvement over *RPLAN*, with better room functionality. However, it relies on room box predictions and heuristic post-processing, leading to issues like misplaced or missing elements, such as bathrooms and kitchens. *WallPlan* combines graph-based geometric structuring with semantic mapping.

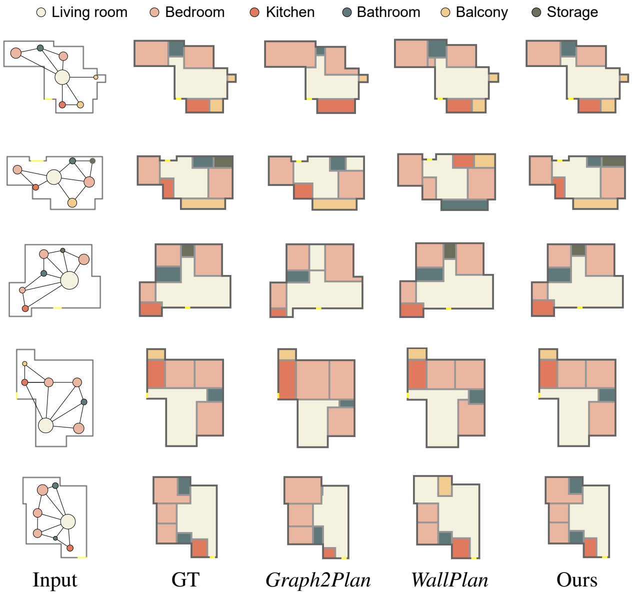


Figure 14: *Comparison to the state-of-the-art techniques on the bubble-diagram-constrained generation. Each row shows the generated floorplans by different methods applied to the same bubble diagrams and architectural boundaries. We illustrate the results of the ground truth, Graph2Plan [HHT*20], WallPlan [SWL*22], and our method, respectively.*

Despite integrating these aspects, they often conflict, causing problems like poorly placed bathrooms and disproportionate room sizes. Our method produces results that are more aligned with ground-truth floorplans, indicating a more robust approach to boundary-constrained floorplan generation.

4.2.2. Bubble-diagram-constrained generation

We compare with *Graph2Plan* [HHT*20] and *WallPlan* [SWL*22] in generating floorplans from bubble diagrams and architectural boundaries (Figure 14). The bubble diagram of the ground truth served as the input constraint. *Graph2Plan* exhibited significant issues across all samples, failing to accurately include essential spaces like balconies and storage, and misplacing rooms, which disrupted the logical flow, such as a bedroom blocking access between the kitchen and living room. These problems stem from heuristic, parameter-sensitive optimization, which can lead to overfitting or unstable refinements. *WallPlan* showed improvement over *Graph2Plan* but still encountered issues like semantic mislabeling and missing components, highlighting method-specific limitations in semantic prediction and continuity. Our method, in contrast, closely mirrored the ground-truth floorplans, demonstrating superior integration of global information and maintaining high accuracy. This comparison underscores the robustness of our approach in bubble-diagram-constrained floorplan generation.

4.2.3. Topological-graph-constrained generation

We compare with *House-GAN++* [NHC*21] and *HouseDiffusion* [SHF23], which generate floorplans from topological graphs (Figure 15). For a fair comparison, we used the comparison approach in *WallPlan*, retrieving dataset samples based on the input topological graph and using the building box of that sample as an additional constraint. *House-GAN++* had the poorest performance, with uneven and jagged room boundaries and chaotic topological connections that impacted room connectivity. *HouseDiffusion* shows improvement with smoother room boundaries, but it struggles with alignment issues of the generated rooms and lacks generation diversity due to its reliance on predefined room shape topologies (fixed number of polygon edges), such as uniformly shaped hexagonal living rooms in Figure 15. In contrast, our method outperformed both, producing more natural and realistic floorplans with streamlined room boundaries and better-aligned rooms.

Table 1: *Quantitative evaluation of the constrained generation. We calculate the FID and KID scores. For Functionality and Connectivity, we calculate the average over the entire dataset. The best results are bolded for emphasis.*

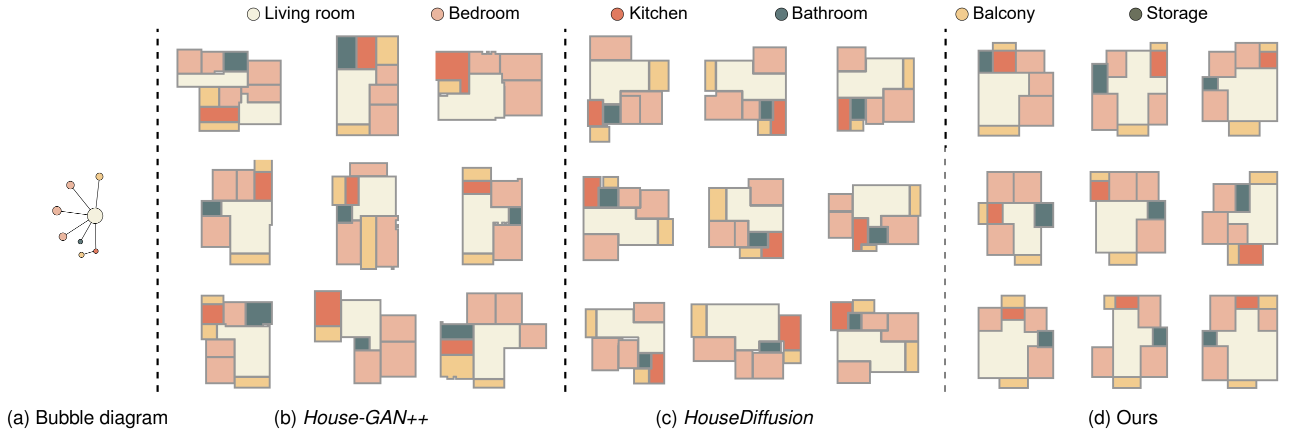
Constraint	Method	FID (↓)	KID (↓)	Functionality (↑)	Connectivity (↑)
Boundary	*RPLAN* [WFT*19]	2.80	2.51	0.95	0.91
	Graph2Plan [HHT*20]	2.07	0.89	0.97	0.97
	WallPlan [SWL*22]	1.67	0.84	**0.99**	0.97
	Ours	**1.47**	**0.81**	**0.99**	**0.98**
Bubble diagram	*Graph2Plan* [HHT*20]	1.45	0.59	**0.99**	0.97
	WallPlan [SWL*22]	0.54	0.10	**0.99**	**0.99**
	Ours	**0.36**	**0.09**	**0.99**	**0.99**
Topological graph	*House-GAN++* [NHC*21]	43.24	50.44	0.98	0.96
	HouseDiffusion [SHF23]	21.49	23.10	**0.99**	0.98
	Ours	**13.26**	**15.55**	**0.99**	**0.99**

4.3. Quantitative evaluation

4.3.1. FID comparison

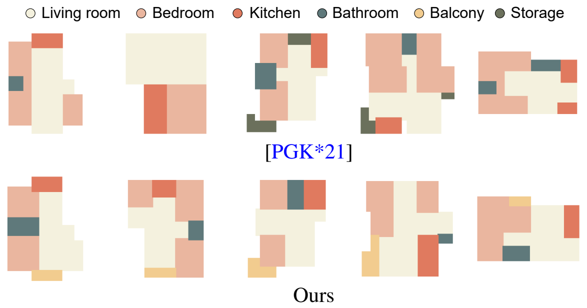
FID (*Fréchet Inception Distance*) [HRU*17] and KID (*Kernel Inception Distance*) [BSAG18] are metrics that compare the statistical distribution of features extracted from a real image dataset with those from a generated image dataset. For the constrained generation, we generate 6000 floorplans for each state-of-the-art technique and calculate their FID and KID scores regarding the ground truth dataset of the corresponding 6000 floorplans. Table 1 shows that our method achieves the best scores compared to other methods, Given only the boundary as input. When adding the ground-truth bubble diagram as the constraint, the scores for our method are particularly impressive, as we obtain a minimal FID = 0.36 and a minimal KID = 0.09 compared to other methods. We also compute the FID and KID scores to quantitatively evaluate our method with state-of-the-art techniques on the topological-graph-constrained generation. We randomly selected ten topological graphs from the dataset as the input constraints and generated 5000 samples for each method (i.e., 500 samples for each topological graph). We directly calculate their scores regarding the ground truth dataset. Our method (FID = 13.26, KID = 15.55) shows a considerable advantage over *House-GAN++*and *HouseDiffusion*.

[PGK*21] proposes to generate room boxes directly using a Transformer-based generative model, followed by optimization-based post-processing. We also compare this method. We utilize the testing data generated by [PGK*21] as introduced in *WallPlan* to compute the FID score between the testing dataset and the ground truth dataset. The results are shown in Table 2. Additionally, Figure 16 presents a side-by-side comparison of the floorplans gener-



(a) Bubble diagram (b) *House-GAN++* (c) *HouseDiffusion* (d) Ours

Figure 15: *Comparison to the state-of-the-art techniques on the topological-graph-constrained generation. (a) Given the topological graph, we show the results of (b) House-GAN++ [NHC*21], (c) HouseDiffusion [SHF23], and (d) our method, respectively. For a fair comparison, we remove the generated doors from the results of House-GAN++ and HouseDiffusion and keep the drawing style.*



[PGK*21]

Ours

Figure 16: *Comparison to [PGK*21] with the same boundary. We adopt a drawing style similar to [PGK*21].*

Table 2: *Given the architectural boundary as the generation constraint, we calculate the FID scores of the floorplans generated by [PGK*21] and our method, respectively.*

Method	[PGK*21]	Ours
FID	14.94	**10.74**

ated by both [PGK*21] and our method. The comparative results demonstrate that our method surpasses [PGK*21] in terms of generation quality. For [PGK*21], relying solely on linear programming to obtain the final results can lead to local geometric and semantic issues in the generated floorplans.

4.3.2. Evaluation for free generation

Existing methods [WFT*19; HHT*20; NHC*21; SWL*22; SHF23] typically rely on constraints such as boundaries, bubble diagrams, or topology graphs for floorplan generation. To our knowledge, no current method achieves truly unconditional generation. We have generated 6,000 floorplans for unconditional generation and conducted quantitative evaluations on these floorplans against the corresponding ground truth dataset. The results show an FID of 5.03 and a KID of 4.88, underscoring the high quality and diversity of the generated floorplans. Furthermore, inspired by point cloud evaluation metrics [YHH*19], we calculated an MMD of 0.0002 and a COV of 50.27%. These metrics demonstrate the capability of our method to generate diverse and high-quality floorplans without relying on predefined constraints, validating its effectiveness and versatility in addressing unconditional floorplan generation.

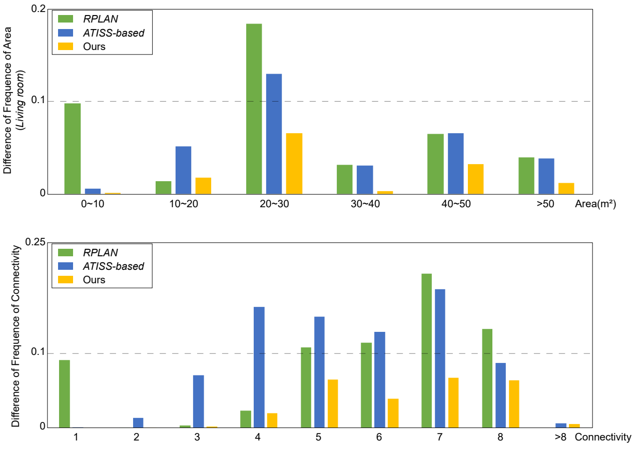


Figure 17: *Difference of Frequencies. We visualize the absolute difference between the distributions of generated and real floorplans for the living room area (top) and room connectivity (bottom). Lower is better; our method shows consistently smaller distribution gaps than* RPLAN *and* ATISS*-based initialization.*

4.3.3. Statistics comparison

FID score alone might capture only some aspects of what makes a floorplan practical or aesthetically pleasing. To improve our evaluation, some statistics of the generated floorplans by different methods are conducted. We first consider the functionality, primarily reflected by the categories of rooms present. A standard floorplan

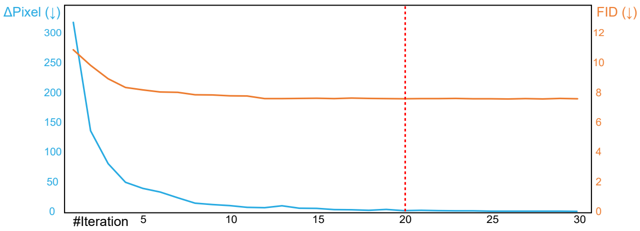


Figure 18: *The optimization process of 500 samples generated by our method. We show (1) the average amount of pixel change in the semantics map before and after each iteration (ΔPixel) and (2) the FID score calculated by the generated floorplan set of 500 samples with the GT dataset for each iteration.*

should encompass four basic categories of rooms: *living room*, *bedroom*, *kitchen*, and *bathroom*. Thus, we define a Functionality metric as the completeness of these four room types within the floorplan, calculated as the number of these basic room types divided by four. We also assess the connectivity between rooms, which indicates the accessibility of different rooms within the floorplan and serves as a crucial metric for identifying potential design flaws. Therefore, we define a measure of connectivity as the ratio of the number of possible doors to the total number of rooms. We calculate the number of possible doors as follows: (1) the living room is directly associated with the main entrance; (2) any room sharing a common wall with the living room should have exactly one door; (3) balconies and bathrooms, not sharing a wall with the living room but with a bedroom, should also have exactly one door. We use the data from the FID comparison to calculate the above statistical metrics (Table 1). The results generated by our method show better functionality and connectivity compared to other methods on both the boundary-constrained generation and bubble-diagram-constrained generation. These results show that our method follows the design constraints while maintaining the rationality of the generated results, demonstrating its validity.

We further show the histograms (Figure 17) of living room area (see more histograms of other room types in the supplementary material) and room connectivity, defined as the number of doors connecting a room to other spaces. Following *ATISS*, we visualize the absolute difference between the frequency distributions of generated and the ground-truth (i.e., Difference of Frequencies), where lower values indicate better alignment with the ground truth. Compared with *RPLAN* and *ATISS*-based initialization, our method consistently exhibits smaller distributional gaps for both room size and connectivity. These results demonstrate that iterative geometry–semantics optimization effectively corrects systematic geometric and topological biases present in the initial generation.

4.3.4. Convergence analysis

To evaluate the convergence of our floorplan generation process, we have generated 500 samples using our method with different architectural boundaries, with the front door as the generation constraint. The result from Figure 18 shows that the optimization process tends to converge after several iterations. In our experiments, 20 iterations of optimization balance effectiveness and efficiency. Although the optimization is not guaranteed to be strictly monotonic at every iteration, empirical results show stable convergence within a small number of iterations.

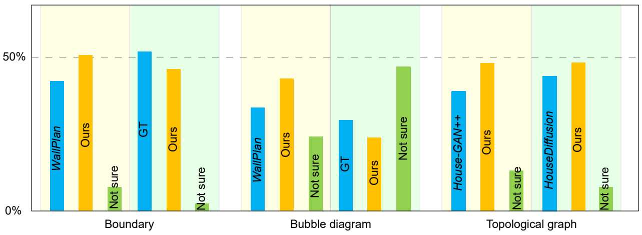


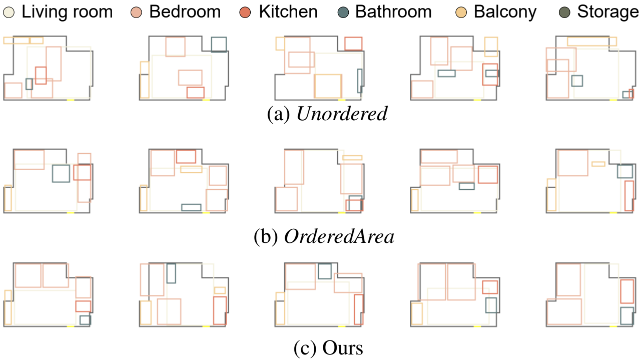
Figure 19: *Perceptual studies. For the boundary-constrained generation and bubble-diagram-constrained generation, we compare our method with WallPlan and GT. For the topological-graph-constrained generation, we compare with House-GAN++ and HouseDiffusion. For each task, we record the voting results of a competitor, our method, and "Not sure", respectively.*

Table 3: *Given the architectural boundary, we show the FID comparison of the ablation study for ordering generation.*

Method	*Unordered*	*OrderedArea*	Ordered (Ours)
FID	2.31	1.53	**1.47**

4.4. Perceptual study

To evaluate the realism of the generated floorplans, we conduct perceptual studies. We compare with *WallPlan* and the ground truth for the boundary-constrained and bubble-diagram-constrained generation. To evaluate the topological-graph-constrained generation, we compare with *House-GAN++* and *HouseDiffusion*. In total, there are six sets of perception studies. Each set consists of 15 comparison tasks, where each pair of floorplans is generated by our method and the competitor. In each task, participants are asked to compare a pair of floorplans and select the one that is better than the other or "*Not sure*", considering the aesthetics, functionality, and connectivity of floorplans. We have recruited 20 participants (Undergraduate students) to complete all of the perception studies. Therefore, for each group of the perceptual study, there are 300 tasks, with 15 tasks for each of 20 participants. The results are shown in Figure 19. In the boundary-constrained generation, the quality of our generated floorplans is higher but slightly weaker than that of the real floorplans. In the bubble-diagram-constrained generation, the quality of our generated floorplans is comparable to that of real floorplans. On the topological-graph-constrained generation, our method is somewhat ahead of both *House-GAN++* and *HouseDiffusion*, and in particular, shows an advantage when compared to *House-GAN++*. The results of perceptual studies indicate that our method produces floorplans that are overall superior to state-of-the-art methods in terms of aesthetics, functionality, and connectivity, achieving results comparable to the ground truth floorplans.



(a) *Unordered*

(b) *OrderedArea*

(c) Ours

Figure 20: *Ablation study for ordering generation. Given the floorplan boundary, we show several generated initial room layouts as well as the final floorplans based on these initial room layouts. (a) Unordered generation, (b) Ordered generation by the room area, and (c) Ordered generation of ours. Our ordered generation produces better floorplans than the other generation strategies.*

Table 4: *Ablation study comparing our full framework with* ATISS*-based initialization across different settings.*

Constraint	Method	FID (↓)	KID (↓)
Free generation	*ATISS*-based inilization	19.77	23.94
	Ours	**5.03**	**4.88**
Boundary	*ATISS*-based inilization	15.69	18.71
	Ours	**1.47**	**0.81**

4.5. Ablation study

We have performed ablation studies to verify the effectiveness of our framework design. More implementation details of ablation studies can be found in the supplementary material.

4.5.1. Ablation study for box generation

Ordered generation guides the generation process. To confirm this, we conduct an ablation study for ordering generation. To achieve unordered generation (*Unordered*), we randomly disrupt ordering room boxes in floorplans when training *BoxGenerator*. On the other hand, we also perform an ablation study on the ordered generation with only sorting by the room area (*OrderedArea*). The results are shown in Figure 20. We generate 6000 floorplans for each ablation method for the ablation study and calculate their FID score regarding the ground truth dataset of the corresponding 6000 floorplans. In Table 3, the large gap between the ordered generation of our method (1.47) as well as ordered by the room area (1.53), and the ablated method Unordered (2.31) illustrates the effectiveness of the ordered generation. Our method still outperforms the ordered generation by the room area, indicating that our strategy in training and generation is relatively appropriate.

To verify that the performance gains of our method are not solely due to a stronger initialization, we compare our full framework with an *ATISS*-based initialization that directly outputs floorplans without iterative refinement. As shown in Table 4, the *ATISS*-based initialization alone yields substantially higher FID and KID scores across all settings: free genratio and boundary-constrained generation. In contrast, our method consistently achieves large improvements after applying iterative geometry–semantics optimization. These results demonstrate that the proposed learning-based refinement is essential for correcting geometric and semantic inconsistencies, and that the performance gains primarily come from iterative optimization rather than the *ATISS*-based initialization. More details on the *ATISS*-based initialization and additional qualitative results are provided in the supplementary material.

Table 5: *Given the architectural boundary, we show the FID comparison of the ablation study for our semantics optimization.*

Method	*NoCoarseOpt*	*NOFineOpt*	*NoSemanticsOpt*	Ours
FID	1.70	2.48	2.53	**1.47**

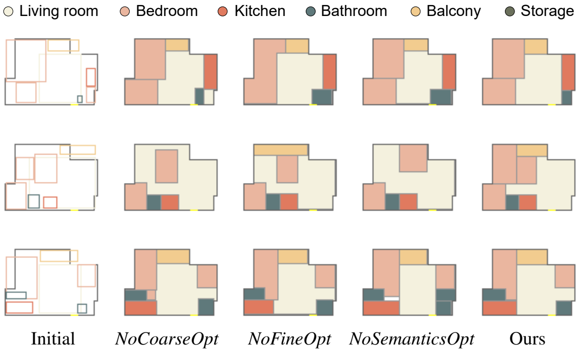


Figure 21: *Ablation study for our semantics optimization. Given the initial room layout, floorplans are generated using NoCoarseOpt, NoFineOpt, NoSemanticsOpt, and our method, respectively. Our method produces significantly better floorplans than the other ablation methods.*

4.5.2. Ablation study for semantics optimization

Our full method can be represented as: *BoxGenerator* + *Coarse Semantics Optimization* + *Fine Semantics Optimization* + *Geometry Optimization*. To show the effectiveness of our semantics optimization, we have conducted several ablation studies (Figure 21): (1) *NoCoarseOpt*: *BoxGenerator* + *Fine Semantics Optimization* + *Geometry Optimization*. (2) *NoFineOpt*: *BoxGenerator* + *Coarse Semantics Optimization* + *Geometry Optimization*. (3) *NoSemanticsOpt*: *BoxGenerator* + *Geometry Optimization*. We generate 6000 floorplans for each ablated method and calculate their FID score regarding the ground truth dataset of the corresponding 6000 floorplans. Table 5 shows the improvement in our FID score (1.47) compared to *NoSemanticsOpt* (2.53), suggesting the importance of our semantics optimization. The FID score of *NoCoarseOpt* (1.70) is lower than the FID score of *OneFineOpt* (2.48), showing the importance of iterative fine semantics optimization.

4.5.3. Ablation study for geometry optimization

To show the effectiveness of our alternating optimization, we have conducted several ablation studies (Figure 22): (1) *BoxCoarseOpt*: *BoxGenerator* + *Coarse Semantics Optimization*. (2) *BoxFineOpt*:

Table 6: *Given the architectural boundary, we show the FID comparison of the ablation study for geometry optimization.*

Method	*BoxCoarseOpt*	*BoxFineOpt*	*NoGeometryOpt*	*SAOpt*	Ours
FID	4.31	3.45	1.95	2.82	**1.47**

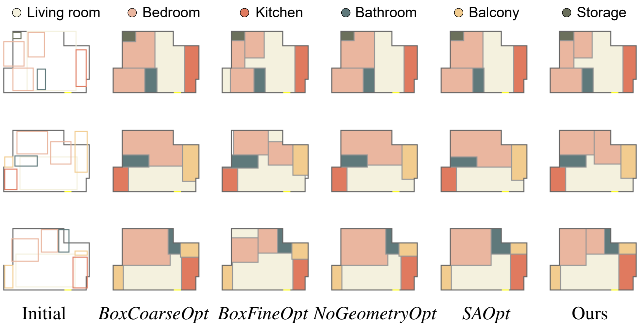


Figure 22: *Ablation study for geometry optimization. Given the initial room layout, floorplans are generated using BoxCoarseOpt, BoxFineOpt, NoGeometryOpt, SAOpt, and our method, respectively. Our method produces significantly better floorplans than the other ablation methods.*

BoxGenerator + Fine Semantics Optimization. (3) *NoGeometryOpt*: *BoxGenerator + Coarse Semantics Optimization + Fine Semantics Optimization*. (4) *SAOpt*: *BoxGenerator + Coarse Semantics Optimization + Fine Semantics Optimization + Simulated Annealing*, which replaces the geometry network with a Simulated Annealing algorithm for geometry optimization. We generate 6000 floorplans for each ablated method and calculate their FID score regarding the ground truth dataset of the corresponding 6000 floorplans. Table 6 shows a noticeable improvement in our FID score compared to others, suggesting the importance of our alternating geometry and semantics optimization. Our FID score (1.47) is lower than the FID score of *SAOpt* (2.82), showing the effectiveness of our geometric network for geometry optimization.

4.5.4. Extreme test for geometry-semantics optimization

Our geometric-semantic optimization does not heavily rely on the quality of the room layout initialization. Even if the initial generation creates rooms with incorrect adjacencies and wrong sizes, our geometric-semantic optimization can produce somewhat reasonable results. We have conducted extreme tests on our geometric-semantic optimization, as shown in Figure 23. Our optimization still yields high-quality floorplans.

5. Conclusion

We propose an alternating semantics–geometry optimization framework for generating diverse, high-quality floorplans under a wide range of design constraints. By iteratively coupling a semantics segmentation network with a geometry prediction network, the framework progressively refines both functional semantics and global layout structure. From a broader perspective, our approach can be viewed as a learning-based post-processing framework. Unlike prior methods that rely on hand-crafted geometric heuristics or threshold-based refinement, we employ data-driven neural networks to regularize and correct layouts produced by an initial generative model, improving robustness and structural consistency while retaining post-processing flexibility.

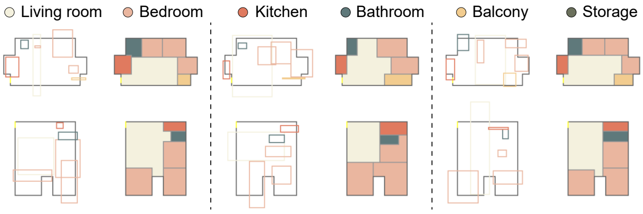


Figure 23: *Extreme tests for geometry-semantics optimization. We apply significant perturbations to the ground-truth room layouts to obtain poor-quality initial room layouts.*

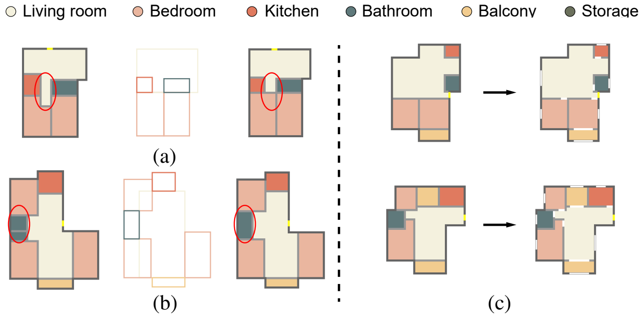


Figure 24: *(a) Our method fails to produce a long corridor connecting the living room to the bedroom. (b) It may merge two small adjacent bathrooms into one large restroom. The generated room layout for each case is shown. (c) We add windows and internal doors to the floorplan using heuristics.*

Our method has several limitations. Owing to the room-box-based representation, complex room geometries can only be approximated by stacked boxes, which restricts the generation of long corridors (Figure 24(a)). In addition, the geometry network may occasionally merge adjacent small rooms of the same category into a larger room (Figure 24(b)). The current framework also generates only axis-aligned layouts. Doors and windows are not synthesized during generation, as they are typically added after the room layout is finalized. As in prior work [WFT*19; HHT*20; SWL*22], they can be incorporated via lightweight heuristic rules (Figure 24(c)). Integrating door and window generation into the learning-based refinement process, as well as extending the representation to richer primitives such as room polygons, are promising directions for future work. Beyond floorplans, the proposed refinement framework is general and may be applied to other layout generation tasks, including graphic design and typography. Integrating AI generation with human editing is important, and a promising direction for future work is to further enhance the controllability and extensibility of our iterative framework to better support interactive design.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive suggestions and comments. This work was supported by the National Natural Science Foundation of China (Grant No. 62372152), the Fundamental Research Funds for the Central Universities of China (Grant No. PA2025GDSK0035).

References

[BSAG18] Bińkowski, Mikołaj, Sutherland, Danica J, Arbel, Michael, and Gretton, Arthur. "Demystifying mmd gans". *arXiv preprint arXiv:1801.01401* (2018) 9.

[BSU*22] Bisht, Sumit, Shekhawat, Krishnendra, Upasani, Nitant, et al. "Transforming an adjacency graph into dimensioned floorplan layouts". *Computer Graphics Forum*. Vol. 41. 6. Wiley Online Library. 2022, 5–22 3.

[Cha20] Chaillou, Stanislas. "ArchiGAN: Artificial Intelligence x Architecture". *Architectural Intelligence*. Springer, 2020, 117–127 3.

[CLWF19] Chen, Jiacheng, Liu, Chen, Wu, Jiaye, and Furukawa, Yasutaka. "Floor-sp: Inverse cad for floorplans by sequential roomwise shortest path". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, 2661–2670 3.

[CPK*17] Chen, Liang-Chieh, Papandreou, George, Kokkinos, Iasonas, et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), 834–848 5.

[CWT*20] Chen, Qi, Wu, Qi, Tang, Rui, et al. "Intelligent home 3d: Automatic 3d-house design from linguistic descriptions only". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, 12625–12634 3.

[GLA*21] Gupta, Kamal, Lazarow, Justin, Achille, Alessandro, et al. "Layouttransformer: Layout generation and completion with self-attention". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 1004–1014 2.

[HHT*20] Hu, Ruizhen, Huang, Zeyu, Tang, Yuhan, et al. "Graph2plan: Learning floorplan generation from layout graphs". *ACM Transactions on Graphics (TOG)* 39.4 (2020), 118–1 2, 3, 7–10, 13.

[HRU*17] Heusel, Martin, Ramsauer, Hubert, Unterthiner, Thomas, et al. "Gans trained by a two time-scale update rule converge to a local nash equilibrium". *Advances in neural information processing systems* 30 (2017) 9.

[HZRS16] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. "Deep residual learning for image recognition". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 770–778 6, 7.

[IKS*23] Inoue, Naoto, Kikuchi, Kotaro, Simo-Serra, Edgar, et al. "LayoutDM: Discrete Diffusion Model for Controllable Layout Generation". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, 10167–10176 2.

[KB14] Kingma, Diederik P and Ba, Jimmy. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980* (2014) 5, 6.

[KW16] Kipf, Thomas N and Welling, Max. "Semi-supervised classification with graph convolutional networks". *arXiv preprint arXiv:1609.02907* (2016) 7.

[LH22] Luo, Ziniu and Huang, Weixin. "FloorplanGAN: Vector residential floorplan adversarial generation". *Automation in Construction* 142 (2022), 104470 3.

[LWKF17] Liu, Chen, Wu, Jiajun, Kohli, Pushmeet, and Furukawa, Yasutaka. "Raster-to-vector: Revisiting floorplan transformation". *Proceedings of the IEEE International Conference on Computer Vision*. 2017, 2195–2203 3.

[LYAM13] Liu, Han, Yang, Yong-Liang, AlHalawani, Sawsan, and Mitra, Niloy J. "Constraint-aware interior layout exploration for pre-cast concrete-based buildings". *The Visual Computer* 29.6 (2013), 663–673 3.

[MSK10] Merrell, Paul, Schkufza, Eric, and Koltun, Vladlen. "Computer-generated residential building layouts". *ACM SIGGRAPH Asia 2010 papers*. 2010, 1–12 2.

[NCC*20] Nauata, Nelson, Chang, Kai-Hung, Cheng, Chin-Yi, et al. "House-gan: Relational generative adversarial networks for graph-constrained house layout generation". *European Conference on Computer Vision*. Springer. 2020, 162–177 2, 3.

[NHC*21] Nauata, Nelson, Hosseini, Sepidehsadat, Chang, Kai-Hung, et al. "House-GAN++: Generative Adversarial Layout Refinement Network towards Intelligent Computational Agent for Professional Architects". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, 13632–13641 2, 3, 9, 10.

[PGK*21] Para, Wamiq, Guerrero, Paul, Kelly, Tom, et al. "Generative layout modeling using constraint graphs". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 6690–6700 3, 9, 10.

[PKS*21] Paschalidou, Despoina, Kar, Amlan, Shugrina, Maria, et al. "Atiss: Autoregressive transformers for indoor scene synthesis". *Advances in Neural Information Processing Systems* 34 (2021), 12013–12026 2, 4.

[PYB*16] Peng, Chi-Han, Yang, Yong-Liang, Bao, Fan, et al. "Computational network design from functional specifications". *ACM Transactions on Graphics (TOG)* 35.4 (2016), 1–12 2.

[PYW14] Peng, Chi-Han, Yang, Yong-Liang, and Wonka, Peter. "Computing layouts with deformable templates". *ACM Transactions on Graphics (TOG)* 33.4 (2014), 1–11 2.

[SHF23] Shabani, Mohammad Amin, Hosseini, Sepidehsadat, and Furukawa, Yasutaka. "Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, 5466–5475 2, 3, 9, 10.

[SWL*22] Sun, Jiahui, Wu, Wenming, Liu, Ligang, et al. "Wallplan: synthesizing floorplans by learning to generate wall graphs". *ACM Transactions on Graphics (TOG)* 41.4 (2022), 1–14 2, 3, 8–10, 13.

[TNM*23] Tang, Jiapeng, Nie, Yinyu, Markhasin, Lev, et al. "Diffuscene: Scene graph denoising diffusion probabilistic model for generative indoor scene synthesis". *arXiv preprint arXiv:2303.14207* (2023) 2.

[TSSV24] Tang, Hao, Shao, Ling, Sebe, Nicu, and Van Gool, Luc. "Graph transformer GANs with graph masked modeling for architectural layout generation". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46.6 (2024), 4298–4313 3.

[WFLW18] Wu, Wenming, Fan, Lubin, Liu, Ligang, and Wonka, Peter. "MIQP-based Layout Design for Building Interiors". *Computer Graphics Forum*. Vol. 37. 2. Wiley Online Library. 2018, 511–521 3.

[WFT*19] Wu, Wenming, Fu, Xiao-Ming, Tang, Rui, et al. "Data-driven interior plan generation for residential buildings". *ACM Transactions on Graphics (TOG)* 38.6 (2019), 1–12 2, 3, 7–10, 13.

[YHH*19] Yang, Guandao, Huang, Xun, Hao, Zekun, et al. "Pointflow: 3d point cloud generation with continuous normalizing flows". *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, 4541–4550 10.

[ZSD24] Zhang, Hang, Savov, Anton, and Dillenburger, Benjamin. "Maskplan: Masked generative layout planning from partial input". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, 8964–8973 3.

[ZSL*20] Zhu, Xizhou, Su, Weijie, Lu, Lewei, et al. "Deformable detr: Deformable transformers for end-to-end object detection". *arXiv preprint arXiv:2010.04159* (2020) 6.