Kalpa Publications in Computing
Volume 22, 2025, Pages 744–755
Proceedings of The Sixth International Conference on Civil and Building Engineering Informatics

Efficient Accessibility Integrated Multi-Occupancy Building Layout Generation Using Constrained Diffusion Models

Haolan Zhang[1] and Ruichuan Zhang[1]
[1] Virginia Polytechnic Institute and State University, Blacksburg, VA, US.
haolanz@vt.edu, ruichuanz@vt.edu

Abstract

Designing complex, multi-occupancy building layouts requires considering complicated spatial arrangements, design constraints, and strict accessibility requirements. Despite advancements in automatic building layout generation, existing methods struggle to address the complexities of these layouts and often overlook critical accessibility features. This paper presents a novel deep learning-based approach for generating complex, multi-occupancy building layouts that meet both architectural and accessibility standards. To improve the efficiency of training, non-corridor rooms are approximated by minimum rotation rectangles, while a graph neural network (GNN) predicts the number of corners for corridors. To address the quadratic complexity of transformers, we incorporate FlashAttention, to enhance computational efficiency. Accessibility features are integrated into the model by enforcing geometric requirements, including room size ratios and maximum corner distances to account for travel distance to egress. Additionally, a distance penalty is introduced in the loss function to ensure compliance with wheelchair clearance requirements. Experimental results show that our approach outperforms baseline models in generating realistic, complex layouts while ensuring compliance with design and accessibility constraints, making it a robust solution for generating multi-occupancy building layouts.

1 Introduction

Designing building layouts is a critical, resource-intensive aspect of building design, involving iterative collaboration between designers and clients to balance client needs and regulatory constraints such as accessibility requirements. Automatic building layout generation has the potential to revolutionize architectural design by enabling architects and designers to produce layouts that meet specific criteria using algorithms and machine learning models.

Early efforts in this domain relied heavily on rule-based systems and optimization algorithms to generate layouts (Dincer et al., 2014; Nisztuk and Myszkowski, 2019), which, while effective for simpler designs, struggled to scale up for more complex architectural demands. Recent advances in machine learning, especially generative models such as Generative Adversarial Networks (GANs), have provided more sophisticated tools for tackling these challenges. GANs and other deep learning models can generate layouts by learning patterns from large datasets (Nauata et al., 2020; Aalaei et al., 2023; Luo and Huang, 2022). These methods excel at creating diverse and realistic layouts but have primarily been limited to small or single-unit layouts.

However, real-world designs, especially multi-occupancy building layouts, are significantly more complex than single-unit layouts. Multi-occupancy building layouts contain more rooms, more complex spatial arrangements, and stricter accessibility requirements. These accessibility requirements, such as ensuring adequate clearance for wheelchair, and adhering to egress distance requirements, add an additional layer of complexity to the design process. Such constraints are essential to meet building codes, such as the Americans with Disabilities Act (ADA), and to ensure inclusivity in the built environment. These complexities present unique challenges that existing models struggle to address and require more scalable model designs and the integration of constraints throughout the generative process.

To address these challenges, this paper introduces a novel approach for generating accessible multioccupancy building layouts using a combination of graph neural network (GNN) and a constrained diffusion model. To handle the increased size and complexity of multi-occupancy building layouts, we optimized the input encoding and incorporated more advanced attention mechanisms to improve learning efficiency. We also introduced additional inputs, including the maximum travel distance within each room, and modified the model's loss function to account for wheelchair accessibility requirements. The proposed approach consists of three key steps: encoding building layout and input requirements, predicting the number of corners for corridors, and generating building layouts through a constrained diffusion process. We trained and validated our model using the Modified Swiss Dwellings (MSD) dataset, an open-source collection of multi-apartment building layouts. Our test results demonstrate that the proposed method outperforms existing approaches in generating complex, accessible multioccupancy building layouts.

2 Background

Generative design of building layouts leverages algorithms, ranging from rule-based, optimization, and artificial intelligence (AI) and machine learning, to automatically generate and optimize building layouts. Early methods capitalized on rule-based methods and optimization methods. For example, Dincer et al. (2014) encoded the adjacency of spaces as neighboring rules and used cellular automata to generate layouts within 8 by 8 grids. Nisztuk and Myszkowski (2019) utilized evolutionary algorithms to generate designs that adhere the topology requirements. These methods rely on extensive hand-crafted features and predefined rules, and they lack flexibility in adapting changes in rules. comes to change of rules.

Recently, machine learning methods has become more popular due to its better adaptability and performance in creating more realistic building layouts. These methods primarily focus on single-unit residential floor plan layout, and use datasets such as RPLAN (Wu et al., 2019), LIFULL (LIFULL, 2015). For instance, Wu et al. (2019) employed an encoder-decoder model that can generate convincing layouts. To include room connectivity requirements, Nauata et al. (2020) and Nauata et al. (2021) combined graph neural networks with GANs to generate layouts from bubble diagrams, using LIFULL and RPLAN to develop their model. Shabani et al. (2023) proposed using a constrained diffusion model to generate layouts based on bubble diagrams. Relying solely on bubble diagrams to condition the GAN model often results in arbitrary boundaries, which are not sufficiently practical for real-world applications. To control the output boundary of layouts, Zheng et al. (2020), Aalaei et al. (2023), Luo and Huang (2022), Hu et al. (2020) develop their models using RPLAN dataset and incorporated layout boundaries into the GAN model, constraining the output's boundary shape.

There are also a few research focusing on multi-occupancy building layouts generation. For instance, Wang et al. (2023) used a U-Net with a spatial attention mechanism to generate building layouts, given boundary constraints. They mainly focus on higher-level layout and omit the layout within each apartment in the building layout. Kuhn (2024) used diffusion model to generate building layouts given structural walls and room connectivity, however, the generated layouts remain overly simplified because all rooms are represented as rectangles, and their visual quality is low (Van Engelenburg et al., 2024).

While these methods have succeeded in generating realistic building layouts and incorporating basic requirements such as room connectivity and boundary constraints, they still face two significant limitations. First, existing models struggles with generating complex geometries and topologies in building layouts. Figure 1 compares the multi-apartment building layout dataset MSD (Van Engelenburg et al., 2024) with the RPLAN and LIFULL datasets, which are single-unit floor plan layouts. It is evident that multi-apartment building layout are considerably more complex than single-unit plans, as they involve a larger number of rooms, increased spatial complexity, and more intricate connectivity requirements between units. The sheer volume of rooms and connections requires more efficient encoding mechanisms and a more scalable model architecture capable of managing the additional complexity. Second, current methods largely overlook the integration of accessibility requirements, which are critical to ensuring both regulatory compliance and inclusivity in building design. For example, the International Building Code (IBC) and ADA mandate that the travel distance to an egress in a building with a single exit point must not exceed 22.86 meters. Additionally, wheelchair accessibility standards require that corridors have a minimum width of 1.5 meters to accommodate wheelchair users. These interconnected requirements add further complexity to the building layout generation process, as they must be considered holistically to ensure that the resulting designs are functional, compliant, and accessible. Addressing these two gaps is essential for making automated building layout designs more applicable in real-world architectural projects.

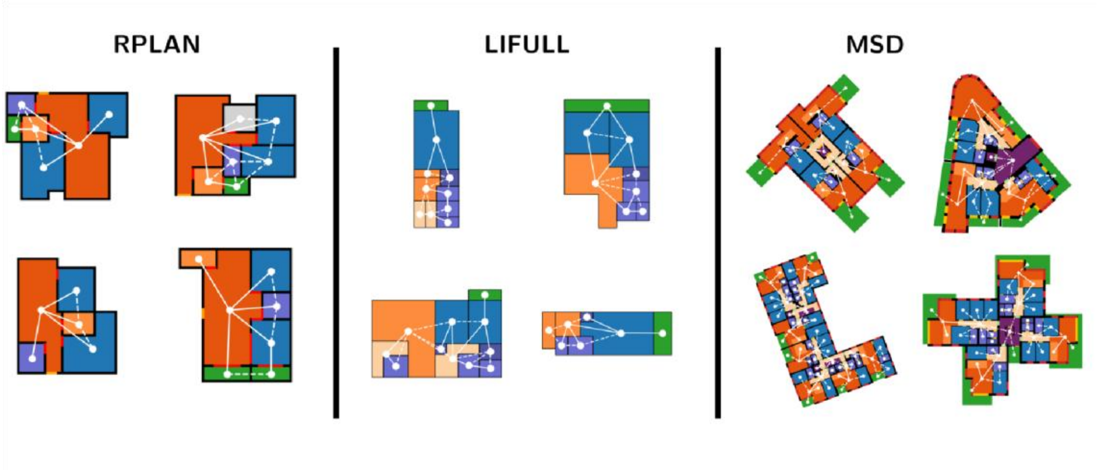

Figure 1: Comparison between floor plan layout datasets (Van Engelenburg et al. 2024)

3 Method

Our proposed approach for multi-occupancy building layout generation employs a GNN and a constrained diffusion model, consisting of three primary steps as depicted in Figure 2.

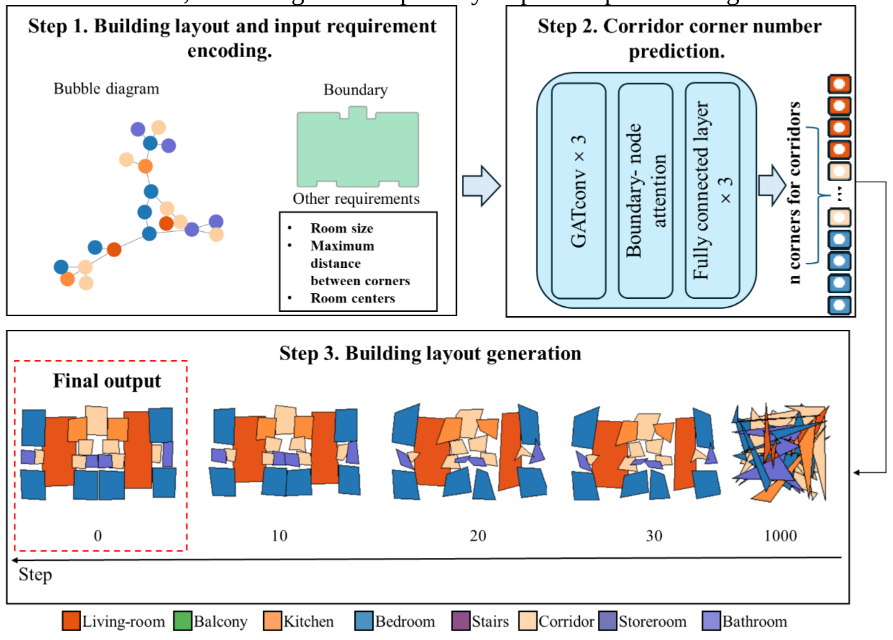

Figure 2: Proposed approach

Step 1. Building layout and input requirement encoding. Each room in a building layout is represented as a polygon, with non-corridor rooms simplified as rectangles with a fixed number of four corners to ensure computational efficiency. The corner number of corridors will be predicted using GNN in Step 2. The input requirement includes a bubble diagram to defines room connectivity and types, building layout's boundary, and size requirements including room sizes and the maximum distances between corners for accessibility compliance. Room center coordinates are also encoded to control the spatial positioning of the generated rooms.

Step 2. Corridor corner number prediction. Corridors, unlike other rooms, require a more flexible representation because their shape depend on the arrangement of adjacent rooms. The corner number for corridors is variable and must be predicted. To handle this, we employ a GNN that predicts the number of corners for each corridor based on the spatial context of the surrounding rooms. The inputs to the GNN include the bubble diagram, room sizes, and room center coordinates, allowing the model to adaptively determine the appropriate corridor corner number, ensuring that it fits seamlessly within the overall building layout.

Step 3. Building layout generation using diffusion model. We use a constrained diffusion model to generate the building layout by progressively refining it through denoising steps. Conditioned on the input encoded in the first step, the model employs a transformer architecture optimized for handling the complexity of multioccupancy building layouts. Several attention mechanisms are incorporated to allow the transformer to learn the intricate relationships between connected rooms, boundaries, and corner coordinates within each room. Furthermore, a modified loss function is introduced to ensuring wheelchair clearance requirements.

3.1 Building layout and Input Requirement Encoding

(1) Building Layout Encoding

Given a building layout F, Let $F = \{R_1, R_2, R_3, \ldots, R_N\}$, where R_N is the room in F. R_N is represented as a polygon, which are a sequence of 2D coordinates. For each R_i in F, $R_i = \{c_{i,1}, c_{i,2}, c_{i,3}, \ldots, c_{i,N}\}$. $c_{i,N}$ is the 2D coordinates and N is the number of corners. The challenge for such encoding lies in determining the number of corners N. A simple approach would be to fix N as the maximum number of corners found in the dataset. While this might work for single-unit layouts, it becomes highly inefficient for complex layouts. For instance, a 40-room layout could require over 1000 corner coordinates, making transformer-based models, with their quadratic complexity, impractical to train. Some of the previous studies (Nauata et al., 2018; Van Engelenburg et al., 2024) treat each room as a rectangle, but this simplification results in a significant loss of detail. To balance efficiency and detail retention, we simplify non-corridor rooms by approximating them using a minimum bounding rectangle, as most non-corridor rooms are close to rectangular. However, corridors, with their irregular and variable shapes, depend heavily on the configuration of surrounding rooms and cannot be effectively approximated as rectangles. For these, we use a GNN to predict the number of corners. This approach reduces the representation to fewer than 200 corner coordinates for a 40-room layout, making it computationally feasible while preserving essential detail. The GNN model is discussed in detail in Section 3.2. Figure 3 illustrates how non-corridor rooms are simplified into rectangles while preserving the original shapes of the corridors (beige color).

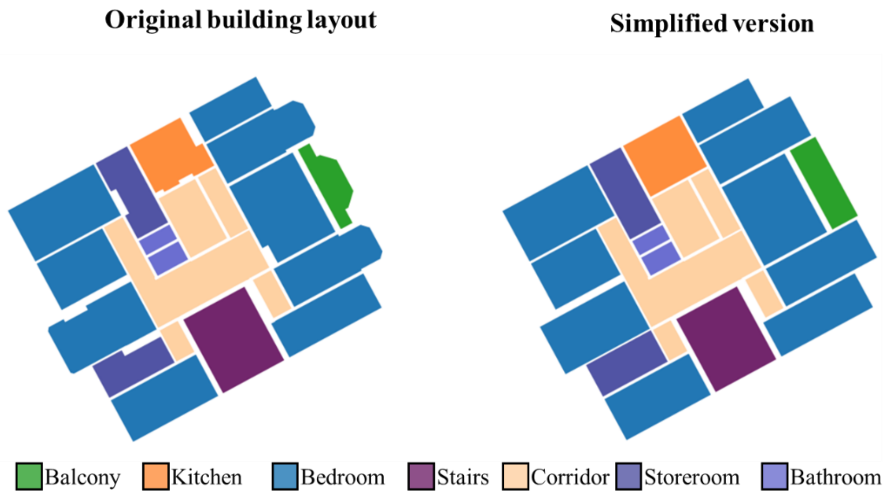

Figure 3: Example of simplified building layout

(2) Boundary Encoding

The building layout boundary defines the overall shape of the layout. Similar to building layout, the boundary is represented as a polygon formed by a sequence of 2D corner coordinates, which can be expressed as $bd = \{C_1, C_2, C_3, \ldots, C_N\}$, where C_N represents the 2D coordinates of the boundary corners.

(3) Bubble Diagram Encoding

The bubble diagram represents room types and their connections, encoded as an adjacency matrix A, where each element A_{ij} being defined by as Equation (1)

$$A_{ij} = \begin{cases} 0 & \text{if room } i \text{ is connected to room } j \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

The adjacency matrix also serves as an attention mask in the transformer model, directing the network's focus during training. A '0' in the matrix indicates a direct connection between two rooms.

(4) Room Size and Room Center Encoding

Room size and center encodings capture additional accessibility and spatial requirements. The room size requirement includes two components: the room size ratio, represented as the ratio of the room area to the total building layout area, ensuring each room meets the specified size constraints, and the maximum distance between corners, which represents the longest possible travel distance within a room. This distance is crucial for calculating the travel distance to the nearest egress and ensuring compliance with egress requirements. The room center is encoded as 2D coordinates, which control the spatial positioning of the generated rooms within the building layout. These encodings ensure that the model maintains spatial coherence and adheres to critical size and accessibility requirements during the generation process.

3.2 Corridor Corner Number Prediction

Since corridors heavily depend on the spatial arrangement of surrounding rooms, we employ a GNN to predict the number of corners for corridors dynamically. To achieve this, we use a Graph Attention Network (GAT) due to its ability to efficiently model the complex relationships between rooms. This is particularly beneficial for corridor corner prediction, as the shape and size of corridors are influenced by internal room connectivity. The node features in the graph are comprised of the room type, room size, and room center coordinates. The edges in the graph represent whether two rooms are connected, based on the adjacency matrix defined in the bubble diagram. As shown in Figure 2, we use three graph attention convolutional (GATConv) layers in our model.

Additionally, an extra input for the building layout boundary is incorporated into the model as the boundary also influence the shape of corridor. The boundary input is processed using a linear layer to align its features with the room-based graph. After this, an attention layer combines the boundary features and the graph features by computing their relevance, which helps the model attend to important boundary information when predicting corridor corner numbers.

3.3 Diffusion Model for Building Layout Generation

(1) Forward and Backward Processes

During the forward diffusion process, Gaussian noise is progressively added to these corner coordinates, gradually transforming the original building layout into random noise. This process follows Equation (2). where γ_t is a noise schedule that transitioning from 1 to 0, and $\epsilon \sim \mathrm{N}(0, \mathrm{I})$ represents Gaussian noise. This process transforms x_0 into random noise x_t.

$$x_t = \sqrt{\gamma_t} x_0 + \sqrt{(1-\gamma_t)}\epsilon \tag{2}$$

The reverse process leverages a transformer model to iteratively remove the noise, gradually reconstructing the building layout from its noisy state. The denoising is conditioned on inputs discussed in Section 3.1 ensuring that the generated building layout adheres to design constraints and accessibility requirements. This iterative refinement allows the model to generate accurate and compliant building layouts.

(2) Feature Embedding

Figure 4 shows the transformer architecture. The model utilized four embeddings: building layout, condition, boundary, and time embeddings. All embeddings are projected into a 512-dimensional feature space for consistency.

Building Layout Embedding. The building layout coordinates are projected into a 512-dimensional space using a linear layer.

Condition Embedding. Room features, including index, type, corner index, size, maximum corner distance, and room center coordinates, are encoded. The index and type are represented as one-hot vectors, room center coordinates are encoded as 2D coordinates, and size and corner distance are encoded as scalars. These features are then concatenated and expanded to 512 dimensions using a linear layer.

Boundary Embedding. The input boundary is encoded and projected into a 512-dimensional vector using a linear layer, ensuring boundary constraints are incorporated during generation.

Time Embedding. The diffusion step is encoded as a scalar and projected into 512 dimensions, allowing the model to track the generation process over time.

Finally, the building layout, condition, and time embeddings are concatenated to form the input for the model.

(3) Transformer Architecture

The transformer model in our approach utilizes multiple new attention mechanisms (Dao et al., 2022), replacing standard attention mechanisms to address the quadratic complexity issue in typical transformer architectures. FlashAttention significantly reduces both computational cost and memory requirements, enabling efficient processing of longer sequences and allowing us to handle complex, multi-occupancy building layouts with greater scalability. This optimization ensure that our model can generate intricate layouts without the prohibitive computational overhead. In our experiments, training time was reduced by half when using FlashAttention compared to traditional attention mechanisms.

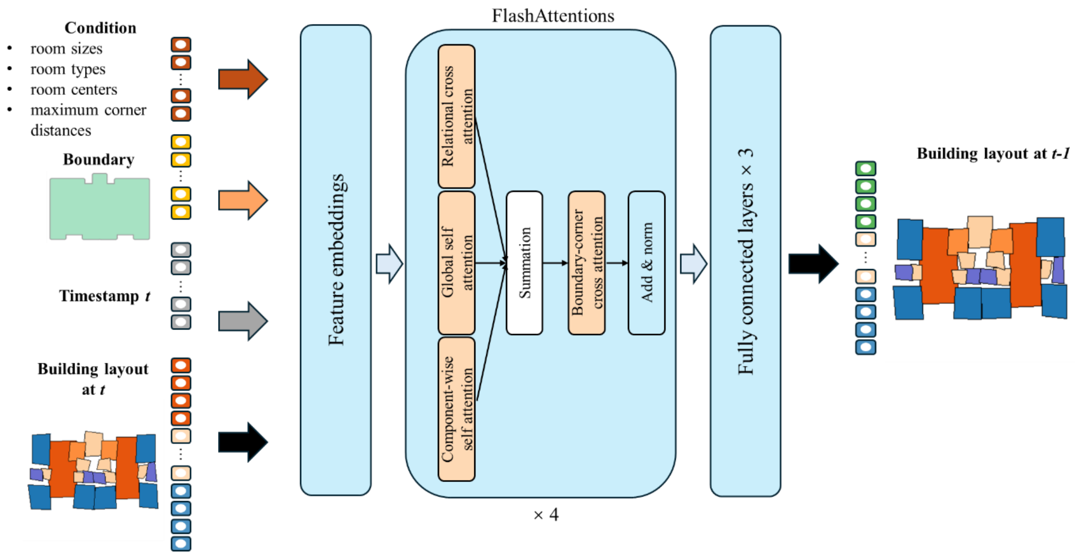

Figure 4: Transformer architecture

We first integrate three attention mechanisms adapted from Shabani et al. (2023): component-wise self-attention within rooms, global self-attention across all corners, and relational cross-attention between connected rooms.

To incorporate the building layout boundary, we use a boundary-to-corner cross-attention module (Equation (3)) to capture interactions between boundary corners and room corners, ensuring proper alignment between the room layout and the external boundary.

$$Attention_{boundary\text{-}to\text{-}corner}\ (Q, K_b, V_b) = softmax\left(\frac{QK_b{}^T}{\sqrt{d_b}}\right)V_b \tag{3}$$

In this equation, Q is derived from a concatenation of the condition embedding, time embedding, and room corner embedding features. K_b and V_b are the key and value matrices from the boundary embeddings, and d_b is the dimensionality of the boundary key vectors.

Additionally, we modified the loss function by incorporating a distance penalty (Equation (4)), which enforces a minimum distance between room corners. This penalty ensures compliance with accessibility requirements by maintaining wheelchair clearance.

$$\text{penalty}_{\text{distance}} = \sum_{i=1}^{n} max\left(0, d_{min} - d(p_i, p_{i+1})\right) \tag{4}$$

Where n is the corner number of room, p_i and p_{i+1} represent consecutive corner points of a room. $d(p_i, p_{i+1})$ is the Euclidean distance between two consecutive corner points. d_{min} is the minimum required distance, in our experiment, we set d_{min} as the smallest minimum distance between room corners in ground truth. The penalty works by penalizing cases where the distance between consecutive corners falls below the specified minimum d_{min}. This encourages the model to generate rooms that maintain sufficient space for wheelchair clearance between their corners.

The final loss function combines the corner coordinate mean squared error (MSE) loss with the distance penalty to balance the precision of corner positioning and adherence to accessibility constraints. As shown in Equation (5).

$$\text{Loss} = (1\text{-}\lambda)\ \text{MSE}_{\text{corners}} + \lambda \cdot \text{penalty}_{\text{distance}} \tag{5}$$

Where λ is a hyperparameter controlling the influence of the distance penalty relative to the corner MSE loss. In our experiment, we set λ = 0.5.

4 Experiments and Results

4.1 Dataset Preparation

For our experiments, we employed the MSD dataset, which contains over 5,300 building layouts of medium- to large-scale building complexes, encompassing more than 18,900 distinct apartments. Each building layout includes annotations for rooms, structural components, and bubble diagrams, offering a diverse range of room shapes. The dataset provides 13 room type categories, essential for capturing the diverse spatial arrangements and functional requirements of real-world designs.

For the cleaning process, we removed building layouts with an excessive number of rooms to ensure consistency and manageability in training. This filtering step helped eliminate overly complex layouts that could introduce noise, ensuring the dataset remained representative of typical multi-apartment designs while maintaining computational efficiency. Additionally, we observed that the original dataset was not axis-aligned. To address this, we rotated the building layouts to align them with the axes, which also served as a data augmentation technique to improve model generalization. Both the original, non-axis-aligned and the newly axis-aligned building layouts were used during training. In total, we used 11,166 building layouts for training, 300 for validation, and 712 for testing.

4.2 Model Training

The model was trained using PyTorch with the Adam optimizer and a batch size of 128. Training lasted for 84,000 steps, starting with a learning rate of 1e-3, which was reduced by a factor of 10 after every 50,000 steps. The diffusion process was structured over 1,000 steps with uniformly sampled time steps.

4.3 Evaluation Metrics

Graph compatibility. This metric evaluates how well the generated building layout matches the input bubble diagram in terms of room connectivity. The graph compatibility is defined in Equation (6).

$$\text{Compatibility}(Q, K) = \frac{1}{S}\sum_{i=1}^{S} \frac{1}{|E_i^K|}\sum_{e \in E_i^K} 1\left[e \in E_i^Q\right] \tag{6}$$

where S represents the total number of test samples, E_i^K represents the set of edges in the graph extracted from the generated building layout K, and E_i^Qrepresents the edges from the input bubble diagram Q. $1\left[e \in E_i^Q\right]$ is an indicator function that equals 1 if the edge e exists in both $E_i^K$and E_i^Q, and 0 otherwise.

Boundary mean intersection over union (MIoU). This metric evaluates the overlap between the input and generated building layout boundaries. The MIoU is defined in Equation (7).

$$\text{MIoU}_{\text{boundary}}\left(B_{\text{in}}, B_{\text{out}}\right) = \frac{1}{S}\sum_{i=1}^{S} \frac{|B_{\text{in}} \cap B_{\text{out}}|}{|B_{\text{in}} \cup B_{\text{out}}|} \tag{7}$$

where B_{in} represents the input boundary, B_{out} represents the output boundary, and the IoU measures the overlap between the input and generated boundaries. S is the number of samples.

Size requirement MSE. The MSE is used to measure the combined difference between the predicted room sizes and the maximum distances between room corners, compared to the ground truth, as shown in Equation (8).

$$\text{MSE}_{\text{size}} = \frac{1}{S}\sum_{i=1}^{S} \left[(\hat{s}_i - s_i)^2 + \left(\hat{d}_{\max,i} - d_{\max,i}\right)^2\right] \tag{8}$$

where s_i is the actual room size, $\hat{s}_i$ is the room size calculated from generated building layouts, $d_{\max,i}$ is the actual maximum corner distance, and $\hat{d}_{\max,i}$ is the maximum corner distance calculated from generated building layouts. S is the number of samples.

Wheelchair clearance compliance percentage. We measure the percentage of rooms that comply with the required wheelchair clearance. This metric is defined in Equation (9).

$$\text{Compliance} = \frac{\text{Number of compliant rooms}}{\text{Total number of rooms}} \times 100\% \tag{9}$$

This metric calculates the proportion of rooms that meet the accessibility requirements for wheelchair clearance.

4.4 Experimental Results and Discussion

Figure 5 showcases sample output building layouts alongside the ground truth, demonstrating the performance of our proposed approach. The generated plans closely adhere to the input requirements, with room locations and sizes highly similar to the ground truth. Although boundary alignment is not always perfect, the model consistently produces layouts that are remarkably close to the ground truth boundaries.

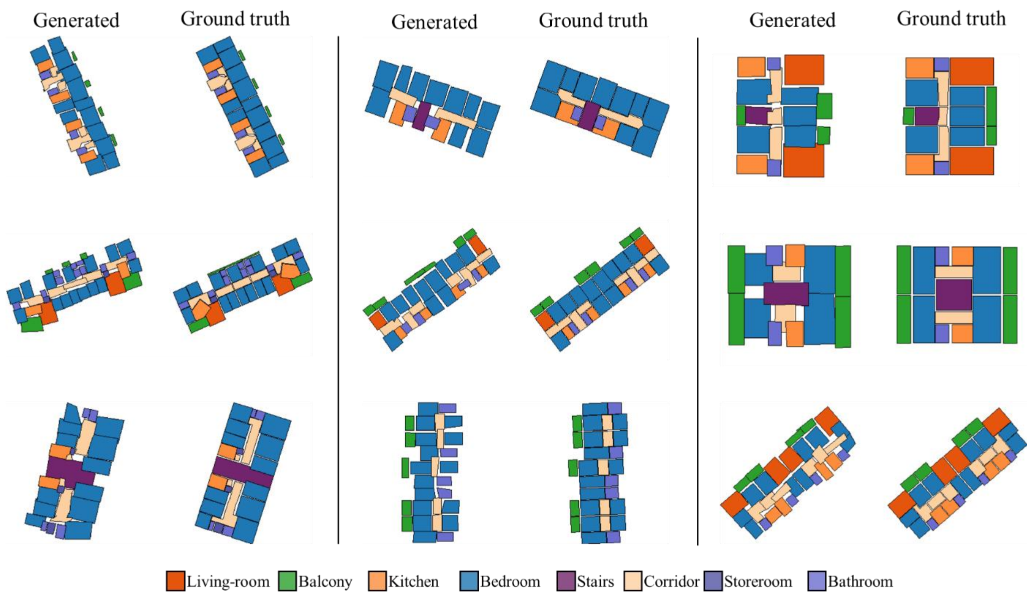

Figure 5: Generated samples

Table 1 presents the graph compatibility and boundary MIoU for our proposed approach compared to the baseline model across different room count intervals. The baseline model is a simplified version that does not incorporate data augmentation or boundary inputs. In terms of graph compatibility, our approach consistently outperforms the baseline in graph compatibility across all room count intervals, with an average improvement of 3%. These results demonstrate the effectiveness of data augmentation in capturing the room connectivity relationships. Regarding boundary MIoU, our approach surpasses the baseline by an average of 16%. These improvements suggest that the proposed approach better maintains alignment between the generated and input boundaries, benefiting from the inclusion of boundary input in the training process.

Room count interval	Proposed approach		Baseline	
	Graph compatibility ↑	Boundary MIoU ↑	Graph compatibility ↑	Boundary MIoU ↑
10-20	**0.78**	**0.73**	0.74	0.58
20-30	**0.79**	**0.74**	0.76	0.59
30-40	**0.82**	**0.73**	0.8	0.58
40-50	**0.82**	**0.74**	0.79	0.56

Note: bold=best performance; ↑ indicates that higher values are better. ↓ indicates that lower values are better.

Table 1: Graph compatibility and boundary MIoU test results

Table 2 compares the Size requirement MSE and wheelchair clearance compliance percentage for our proposed approach and the baseline model across different room count intervals. The baseline model does not include the maximum length input or the distance penalty in the loss function, which impacts its ability to enforce accessibility constraints. For the room size and corner distance MSE, the baseline has comparable results in the lower room count intervals. However, as the room count increases, our proposed approach shows improvement, achieving 0.01 of improvements. For wheelchair clearance compliance percentage, our proposed approach outperforms the baseline in the latter three room count intervals and with an average improvement of 1.7%. This demonstrates that the inclusion of the maximum length input and distance penalty leads to better accessibility compliance, especially in larger and more complex layouts.

Room count interval	Proposed approach		Baseline	
	Size requirement MSE ↓	Wheelchair clearance compliance percentage ↑	Size requirement MSE ↓	Wheelchair clearance compliance percentage ↑
10-20	**0.07**	**0.94**	**0.07**	**0.94**
20-30	**0.06**	**0.96**	**0.06**	0.95
30-40	**0.05**	**0.97**	0.06	0.95
40-50	**0.04**	**0.97**	0.05	0.95

Note: bold=best performance; ↑ indicates that higher values are better. ↓ indicates that lower values are better.

Table 2: Size requirement MSE and wheelchair clearance compliance percentage

5 Conclusions

In this paper, we introduced a novel approach for generating complex multi-occupancy building layouts by combining a graph neural network (GNN) with a constrained diffusion model. To improve the efficiency of training on complex building layouts, we optimized the building layout encoding by approximating non-corridor rooms with minimum bounding rectangles and using a GNN to predict the number of corners for corridors. Additionally, to address the quadratic complexity of transformers, we employed FlashAttention, which significantly reduces computational cost and memory usage, making the model more efficient. To incorporate accessibility features, we introduced room size requirements, including room size ratios and maximum corner distances, alongside fundamental constraints such as layout boundaries and bubble diagrams. We also designed a distance penalty to ensure compliance with wheelchair clearance regulations.

Our model significantly outperforms the baseline across several key metrics, including graph compatibility, boundary alignment, and wheelchair clearance compliance. Specifically, the proposed approach improves graph compatibility by an average of 3%, boundary alignment by 16%, room size and corner distance MSE by 0.01, and wheelchair clearance compliance by 1.7%. These results demonstrate the model's capability to generate building layouts that not only adhere closely to design constraints but also meet important accessibility requirements.

Looking forward, we identified three key areas for improvement. *(1) Refining corridor shapes.* The current approach approximates corridors but does not fully capture their complex and irregular geometries. Further refining the model to better represent corridor shapes will enhance the practicality of the generated building layouts. (2) *Addressing overlapping and gaps*. Although our model performs well overall, issues with room overlaps and gaps between rooms still arise in more complex layouts. Developing more robust mechanisms to prevent overlaps and close gaps will improve the usability of the building layouts. (3) *Introducing more accessibility features.* we will expand the scope to include additional accessibility features such as door widths, accessible bathroom layouts and will increase the model's applicability for universal design and compliance with more comprehensive accessibility standards.

References

Aalaei, M., Saadi, M., Rahbar, M., and Ekhlassi, A. (2023). Architectural layout generation using a graph-constrained conditional generative adversarial network (GAN). *Automation in Construction*, *155*, 105053.

Dao, T., Fu, D., Ermon, S., Rudra, A., and Ré, C. (2022). Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35, 16344-16359.

Dincer, A. E., Cagdas, G., and Tong, H. (2014). A digital tool for customized mass housing design. *Fusion, Proceedings of the 32nd eCAADe Conference, Department of Architecture and Built Environment*, pp. 10-12.

Hu, R., Huang, Z., Tang, Y., Van Kaick, O., Zhang, H., and Huang, H. (2020). Graph2plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics (TOG)*, 39(4), 118-1.

Kuhn, E. (2023). Adapting HouseDiffusion for conditional Floor Plan generation on Modified Swiss Dwellings dataset. *arXiv preprint arXiv:2312.03938*.

LIFULL Co., Ltd. (2015). LIFULL HOME'S Snapshot Data of Rentals, Informatics Research Data Repository, *National Institute of informatics.*

Luo, Z., and Huang, W. (2022). FloorplanGAN: Vector residential floorplan adversarial generation. *Automation in Construction*, 142, 104470.

Nauata, N., Chang, K. H., Cheng, C. Y., Mori, G., and Furukawa, Y. (2020). House-gan: Relational generative adversarial networks for graph-constrained house layout generation. *Computer Vision–ECCV 2020: 16th European Conference*, Glasgow, UK, pp. 162-177.

Nauata, N., Hosseini, S., Chang, K. H., Chu, H., Cheng, C. Y., and Furukawa, Y. (2021). 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*, pp. 13632-13641.

Nisztuk, M., and Myszkowski, P. B. (2019). Hybrid evolutionary algorithm applied to automated floor plan generation. *International Journal of Architectural Computing*, *17*(3), 260-283.

Shabani, M. A., Hosseini, S., & Furukawa, Y. (2023). 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*, pp. 5466-5475.

Van Engelenburg, C., Mostafavi, F., Kuhn, E., Jeon, Y., Franzen, M., Standfest, M., ... and Khademi, S. (2024). MSD: A Benchmark Dataset for Floor Plan Generation of Building Complexes. *arXiv preprint arXiv:2407.10121*.

Wang, L., Liu, J., Zeng, Y., Cheng, G., Hu, H., Hu, J., and Huang, X. (2023). Automated building layout generation using deep learning and graph algorithms. *Automation in Construction*, *154*, 105036.

Wu, W., Fu, X. M., Tang, R., Wang, Y., Qi, Y. H., and Liu, L. (2019). Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (TOG)*, *38*(6), 1-12.

Zheng, H., Keyao, A. N., Jingxuan, W. E. I., and Yue, R. E. N. (2020). Apartment floor plans generation via generative adversarial networks. *25th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA 2020): RE: Anthropocene, Design in the Age of Humans*, pp. 601-610.