2024 EC³
2024 European Conference on Computing in Construction
Chania, Crete, Greece
July 15-17, 2024

INTERACTIVE AI FOR GENERATIVE HOUSING DESIGN BASED ON GRAPH NEURAL NETWORKS AND DEEP GENERATIVE MODELS

Tian Xia[1], Alex Ledbetter[1], Alexandru Bobe[1], Jeroen Hofland[1], Berend Krouwels[1], ,Tong Wang[1], Luciano Cavalcante Siebert[1], Paul Chan[1], and Jian Yang[2]
[1]Delft University of Technology, Delft, Netherlands
[2]Shanghai Jiao Tong University, Shanghai, China

Abstract

Nowadays, the complexity of housing design has grown substantially to meet multitudinous requirements (e.g., cost, space, esthetics, sustainability, circularity, and modularity) of diverse stakeholders, which poses great challenges to traditional labor-intensive design processes. Automatic design tools are being developed to assist designers handle tedious work at scale. However, knowledge gaps still exist in harnessing deep learning models to learn from human experience for more efficient design generation while keeping the data understandable and interoperable. Moreover, human-in-the-loop approach is largely neglected in the automatic design tools, which are essential for more customized and user-centered design. This research utilizes graph data to parametrically represent housing designs and graph-representative deep generative models for design generation, which provides an interactive design approach for the users at every step. This method enables deep learning models to semantically understand hidden patterns and knowledge in housing designs and facilitate the human-centered design process by returning feasible and parametric housing design alternatives. All codes can be found at: https://github.com/jlhofland/housing-design.

Introduction

The design process has been identified as one of the most significant elements influencing housing quality (Hamzah et al., 2011). Traditionally, the creation of housing designs has involved labor-intensive, manual processes, relying on the expertise of designers. This conventional approach has been characterized by iterative design revisions and historical references. However, it faces challenges related to scalability and adaptability to evolving societal needs (Williamson and Wong, 2022). In response to these challenges, recent advancements in technology and more specifically in the area of Generative AI have introduced the possibility of partially automating the housing design process (Ko et al., 2023). These advancements have the potential to accelerate design iterations and augment the creativity of the proposed designs (As and Basu, 2021).

While various techniques for automatic housing design generation offer distinct advantages, they also come with some limitations. Generative Adversarial Networks (GANs) have the ability to produce realistic floorplan images (He et al., 2022; Wu et al., 2019). However, the image-based approaches lack interoperability with common architectural applications (e.g., CAD and BIM) (Ko et al., 2023). Moreover, it is hard to incorporate various constraints and user requirements into the image-based floorplan generation models. Graph-based methods like Graph Neural Networks (GNNs) can be a complementary approach which has good interoperability due to its parametric nature and compatibility with state-of-the-art generative models (Guo and Zhao, 2023). Therefore, it is worth noting that a promising approach lies in combining GANs with graph-based methods for their generative ability and interoperability.

Numerous studies have incorporated graphs as semantic representation to guide the floorplan generation with GAN, which could be referred to as graph-constrained Relational Generative Adversarial Networks (RaGAN) (Nauata et al., 2021, 2020; Shabani et al., 2022). By using graphs, the users can more easily input and modify the layout at the semantic level, which provides a user-in-the-loop approach and ensures that the model not only produces functional designs but also aligns closely with the user requirements. Bubble diagrams have long been used as a conceptual tool in housing design for visualizing spatial relationships and zoning (Zheng and Petzold, 2023), which has been widely incorporated as graph input to guide floorplan generation with GANs in existing models (Nauata et al., 2021, 2020; Shabani et al., 2022). However, most of the models only generate floorplans based on bubble diagrams given by users and neglect the room layout arrangement task in the conceptual design stage (Nauata et al., 2021, 2020; Shabani et al., 2022). Moreover, bubble diagrams primarily serve as a conceptual tool in housing design for visualizing numbers and spatial relationships among rooms, which can not incorporate constraints from building boundaries (Zheng and Petzold, 2023). Graph2Plan (Hu et al., 2020) allows for building boundaries as user input and then extracts the layout bubble diagrams and generates floorplans. However, Graph2Plan does not generate the graph but rather retrieves it from a set of similar floorplans from a database. This makes it less flexible as certain input constraints might not coincide with the floorplans in the database. Moreover, the adjacencies between rooms and exterior walls cannot be defined. It is of vital importance to incorporate such conditions if the housing design needs to fit into constraints from existing walls and rooms, which happens a lot in renovation cases. Also, the edge features of the graph are non-configurable, which means it does not allow for interior doors to be specified by the user.

Heterogeneous graphs could offer a more expressive approach for housing design generation by providing a data-driven framework that explicitly encodes the attributes and relationships among different rooms and components (e.g., walls, windows, and doors) (Gan,

2022). Therefore, instead of using bubble diagrams, we develop heterogeneous graphs to capture complex constraints and diverse attributes for more powerful and versatile housing design generation. To achieve higher flexibility, we generate both layout graphs and floorplans by using generative models based on graph representation (Figure 1).

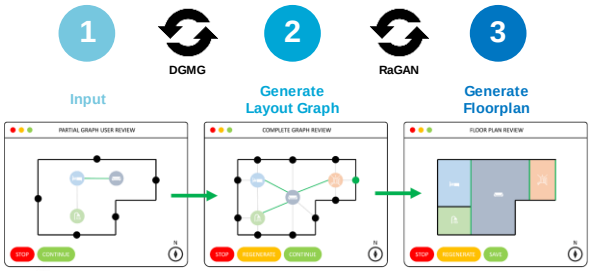


Figure 1: Interactive design generation process where the dots are walls and rooms, and edges represent adjacency. For envisioned graphical interfaces see Figure A4.

Specifically, heterogeneous graph representation will be developed to explicitly encode the attributes and relationships among the exterior walls and rooms which would enable users to input their requirements and constraints (Step 1). Starting from the input, our generative housing design pipeline encompasses layout graph generation model (Step 2) and floorplan generation model (Step 3), which are developed based on DGMG (Li et al., 2018) and RaGAN from Housegan++ (Nauata et al., 2021) because of its wide adaptability and state-of-the-art performance. We made improvements to the models by incorporating edge features in the message passing over heterogeneous layout graphs. On top of this, the user can intervene at each step in the pipeline to make changes to all interim results. By doing this we create a model that proactively keeps the user in the loop and achieves more user-centered design through a collaborative approach compared with end-to-end models (e.g., FloorGAN (Upadhyay et al., 2023)).

Research Methods

Overview

An overview of the pipeline is illustrated in Figure 2, where the top depicts user interaction and the bottom is model training. Starting with the user-input file containing user constraints, augmented DGMG (see next sections) will generate nodes and edges to finish the layout graph. User interaction is available to evaluate the generated layout and make changes along the way. When they are satisfied, the pipeline will pass the layout graph to Hetero-HouseGAN++ (HHGPP, see next sections) for floorplan generation by representing the spatial configuration of rooms through masks, which would then be saved in parametric format. A final user-input step will occur offering to regenerate the plan as necessary.

Initial user input: The user can input the building boundary and specify the location of entrance doors. It is also allowed to provide initial constraints such as the number of rooms for each type. More importantly, the location and adjacency of certain rooms can be specified in the input as partial layout graphs, which is an essential feature for the renovation housing design. It should be noted that only a text-based interface is available for initial user input now, which, however, can be transformed into a graphical interface and embedded into domain applications (e.g., CAD and BIM) in future work. The envisioned graphical interface is depicted in Figure A4.

Generation of layout graph: To generate the customized layout graph based on user input, we adopt a graph generation model based on augmented DGMG (Li et al., 2018). The generated layout graph will explicitly formulate the adjacency relationships of rooms and walls. For instance, a bedroom could be adjacent to the south side of an exterior wall and connected to the north side of the living room through an interior wall with a door. The model will be trained on the large-scale floorplan dataset LIFULL, which was transformed into parametric representation (Nauata et al., 2020). The users can evaluate and modify the generated layout graph or regenerate it based on their preferences.

Generation of floorplan: To acquire the floorplan with precise spatial configurations that fits the layout, we adopt a graph-constraint RaGAN based on Housegan++ (Nauata et al., 2021), which is referred to as HHGPP. The model could aggregate the features of the layout graph through message passing and generate the spatial configuration of rooms as volume masks. The output of

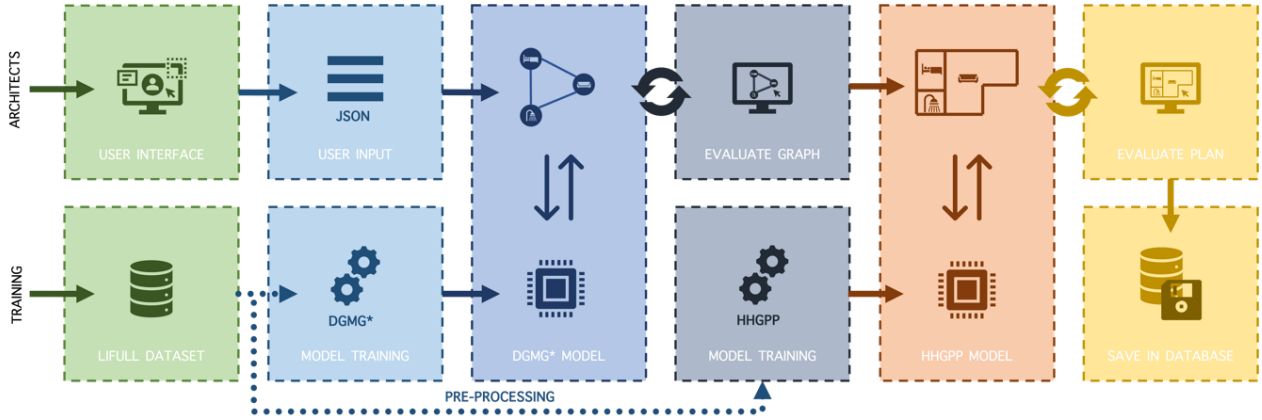


Figure 2: Pipeline for housing design, top: user interaction, bottom: model training

the pipeline will be a parametric floorplan and a graph that depicts the layout arrangement, which can be fed into domain applications for further editing and optimization.

The following sections explain these steps in more detail.

Initial user input

The user could input the requirements and constraints for the housing design(Step 1, Figure 3). The requirements would be represented as a conditioning vector. The constraints would be transformed into a partial graph (Step 2, Figure 3).

User input:

- A vector representing user requirements: the number of various room types, and additional entries indicating if the room quantity is upwards flexible (3 vs 3+ for example).
- The constraints from building boundary and predefined location of rooms (i.e., preferred location of specific rooms or fixed location of existing rooms), transformed into heterogenous-graph representation: exterior walls as nodes (node features with starting and ending vertices, final element indicating with entrance door or not); corners which connect the exterior walls as edges (edge feature represent the corner type); predefine rooms as nodes (node feature represent room types); adjacency between rooms and exterior walls as edges (edge features represent relative direction); adjacency between rooms as edges (edge features represent the relative direction and connection type (i.e. through a wall or a door, or directly connected)).

Generation of layout graph

In this section, we develop the model to generate layout graphs conditioned on user input by completing the input partial graph from user constraints. User requirements such as the number of bedrooms and bathrooms are fed into the model as a conditioning vector (Li et al., 2018) to generate graphs that meet a client's needs. The graph generation model is based on augmented DGMG, which could incorporate edge features in the message passing for heterogeneous layout graph generation. Through training over a large set of real architect-created home layout graphs (Nauata et al., 2020), the model learns to generate graph layouts that respect various implicitly learned architectural design rules and meet user requirements and constraints.

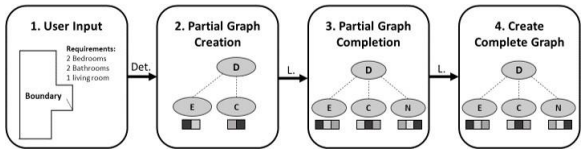


Figure 3 Layout Graph Generation Model. D(decision), E(Add Edge), C(Choose connection), N(Add Node). Det. specifies that the process is deterministic, while, L. states that the step is learned from data.

The layout graph generation process based on augmented DGMG follows an autoregressive generation mode (Li et al., 2018), which is depicted in Steps 3 and 4, Figure 3. The general procedure and our modification are described as follows, and a more detailed illustration can be found in (Li et al., 2018).

Layout graph generation:

(a) Choose to add a new node A (yes/no).

(b) Choose to add a new edge from source node A (yes/no).

(c) If yes:

i. Choose a destination node B from pre-existing nodes in the graph. In this research, we also predict the feature of the added edge in this step.

ii. Perform GNN message-passing to update node representations. In particular, we augment DGMG by incorporating edge features in the message passing.

When the model chooses not to add another node, the graph is complete. The output of the model would be a complete layout graph, which could be evaluated, edited, and regenerated through the envisioned user interface.

Generation of floorplan

To generate the floorplans with precise spatial configuration of rooms that respect the constraints specified in this heterogenous layout graph, a graph-constraint RaGAN is developed based on augmentation of Housegan++ (Nauata et al., 2021) (Figure 4).

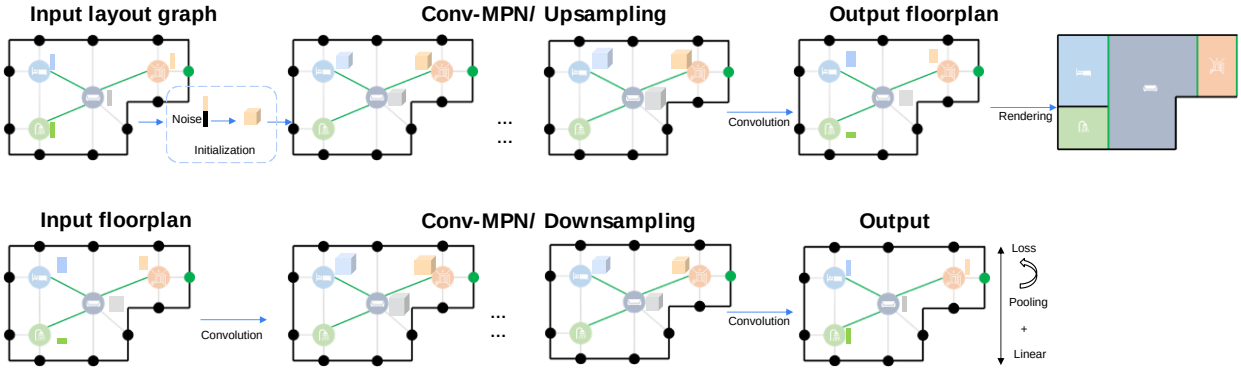


Figure 4 Floorplan generator (top) and discriminator (bottom). ConvMPN is our backbone architecture [9]. The input graph constraint is encoded into the graph structure of their relational networks.

Hetero-HouseGAN++ (HHGPP) is a generative adversarial network whose generator transforms input data (lists of data representing layout graph, e.g. nodes, edges, features, etc.) through Convolutional Message-Passing Network (ConvMPN) into pixel-based room masks. These room masks are later vectorized into parametric representations of floorplans. The discriminator then performs the reverse process, starting with room masks, and resulting in a binary response as to the validity of the room masks it was provided, given the graph data also fed in alongside.

Our improvement in HHGPP compared with Housegan++ includes incorporating new node types, edge types, and node and edge features for graph-constrained floorplan generation. First, GNN message-passing occurs as convolutions over a large data block constructed as the concatenation of the graph node feature volumes, the pooled features volumes of their neighbours which get encoded with the edge type and edge features across these connections, and the pooled feature volumes of the disconnected nodes. These convolutions happen three times, downscaling as we go to result in the final encoded nodal feature volume representations. The second innovation is to use conditional generation whereby some room masks are predefined during mask generation such that the generator learns to paint the remaining masks for the floorplan.

Results and Evaluation

Training data preparation

All the data used for this project are extracted and augmented from the LIFULL dataset (available at: https://www.nii.ac.jp/dsc/idr/en/lifull/), a database of real floor plan designs. Each floor plan is represented as a list of values for room ID and type, room bounding boxes, floorplan edges, edge adjacencies, and IDs of edges with doors. Note that edges here refer to exterior and interior walls, not graph edges.

We filter out floorplans with either completely disconnected rooms or rooms that are only connected at a corner. From this raw data, we create two types of files used for training the layout graph generation model: user-input files and sequences of graph generation. For training the floorplan generation model, we transform the bounding boxes of rooms into the masks (3D volumes) of room nodes to represent their spatial configurations. The pairs of layout graphs and floorplans (graphs with masks for room nodes) will be used to train the generator and discriminator in the HHGPP.

Network training

The layout graph generation model will be trained through teacher forcing throughout the autoregressive generation process of DGMG (Li et al., 2018). The sequences-type files are lists of the ground-truth answers to the sequential generation questions, “should another node be added ?”, “what node should this edge be connected to?,” such that, when followed, producing the precise home layout graph corresponding to the LIFULL homes. Some decisions in the sequence are binary yes/no responses, while others are multi-class prediction-type responses. In either case, decisions are of a discrete number of choices and proceed by the model calculating a prediction for the logits of each response type. This is a numerical value where a highly positive value represents high confidence in that prediction value being the correct one, while a low or highly negative value indicates low confidence in that option.

We then convert these predicted logits to probability mass values (or likelihoods) via the sigmoid (binary) or softmax (multi-class) equations. Finally, our loss accumulated at each decision is equal to the negative log of the predicted likelihood for the correct response. Minimizing this value then seeks to maximize the likelihood of the correct response. The corresponding loss function is illustrated in (1):

$$loss = -log(softmax(predicted - logits)) | correct - response \quad (1)$$

During training, the model will make a predicted answer, and loss will be accumulated if the answers are wrong. After each batch, the loss is backpropagated through the various network layers, and the model parameters are updated to minimize this loss. The loss and generated graphs can be found in Figure A1 and A2.

To train the floorplan generation model, we iterate through the dataset in batches, following a typical training process of GAN models by iteratively training the discriminator, and then the generator, each per batch. Training the discriminator involves generating a set of room masks, and passing these into the discriminator to get a “fake validity” score (binary response, real or fake), then passing in the corresponding real masks to get a “real validity score”. These are summed with fake scores negated, along with a gradient penalty term. This loss is then used to backpropagate and update weights. Training the generator then proceeds by the generator producing masks, the discriminator scoring them, and any predicted as fake are counted up as the loss. Additionally, to train the generator to not change the conditioned masks, an L1 loss between the conditioned and generated masks is added to the discriminator loss. The detailed description of the loss function can be found in (Nauata et al., 2021, 2020). This loss is then backpropagated over the generator and weights are updated (Figure A3).

Evaluation

In this section, the evaluation methods are discussed at different points in the pipeline.

The validity of the generated layout graph is evaluated based on compliance with user input and basic design rules. It should be noted that the quality of the generated layout should be evaluated from a more comprehensive architectural design perspective and with potential human expert evaluation for benchmarking against other models, which will be future works and will be discussed in the next chapter. The validity evaluation criteria for the current research are as follows:

R1. the user input constraints are all fully met.

R2. all rooms have at least one door.

R3. exterior walls connect to two other walls and one room, which ensures no solitary walls exist.

R4. each room connects to at least one other room (or wall), which ensures no solitary rooms exist.

R5. each room has outgoing room adjacencies that cover at least two cardinal directions. This attempts to ensure that a room is bounded in all directions. Ground truth homes show that 99.97% of homes meet this criteria.

Table 1: Evaluation Validity Results for DGMG, 10x100 Graph Runs

	R1	R2	R3	R4	R5
Mean	0.13	0	0.061	0.091	0.337
STD	0.126	0	0.167	0.201	0.192

Table 1 shows the ratio of generated home layout graphs that fail each of the validity criteria. These results come from generating 100 layout graphs each from ten different user-input files, with results averaged over the ten runs. Finally, of the 1000 total graphs generated, they are valid 54.1% of the time with a standard deviation of 25.5% between the ten runs. It should be noted that the failure rate to meet R5 is significantly higher, which implies that the graph-representative model finds it more challenging to capture the directional relationships among rooms. It will be discussed in the next section.

The floorplan generation model is evaluated using the Fre´chet Inception Distance (FID). FID is a metric for image generators that evaluates the realism of the generated images by comparing them to a set of real, given images (Heusel et al., 2017). In our evaluation, the FID is used to assess the quality of the floorplans generated by the model using the floorplans in the LIFULL database.

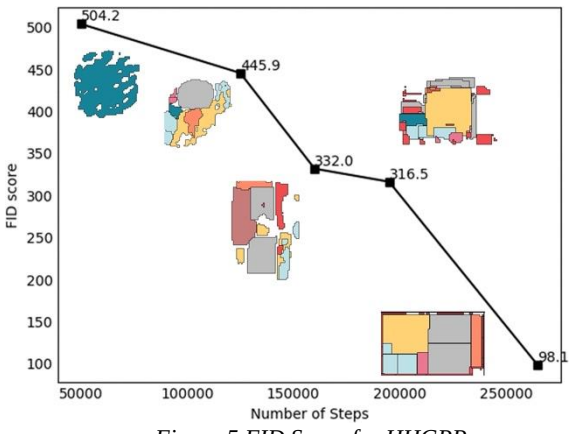


Figure 5 FID Score for HHGPP

Our FID score, Figure 5, is calculated after roughly every 50000 training steps. As the model learned, the floorplan generation model decreased the FID score from 504.2 to a final score of 98.1, indicating more realistic generated images. The score of 98.1 indicates a high similarity between the two distributions of real and generated images. Given enough computing power and training steps, the results could be further improved.

Discussion

We set out to show that housing design assisted by interactive AI based on graph representation is possible. Through the design of the envisioned user interface, review of the workflow, and evaluation of the supporting AI models, we have shown that architects have a new tool at their disposal to generate both home layout graphs and corresponding floorplans pertaining to their needs at higher efficiency. Currently, only a text-based interface is available for users. More advanced graphical interfaces could be developed and embedded into domain applications (e.g., CAD and BIM) based on the graph-based approach.

Both our augmented DGMG framework is a sufficiently capable architecture to learn to generate valid layout graphs that respect the user's input constraints, and the graph-constraint RaGAN architecture present in our HHGPP model is sufficient to learn to translate layout graphs into floorplans that maintain the constraints. This is important because it shows that the user input is not too complex to be learned by generative models in general.

However, the prediction of edge direction in the layout generation model is still imperfect. When the model adds a new node and a new edge from that node, it must then decide in what direction the connection will point. It is important to avoid contradictory constraints. The graph generation model still finds it hard to capture and fulfill all the hard constraints through learning. The reasons for the imperfection can be representation-related, model-related, or training-related. We believe the training process is adequately done considering the loss has dropped to a reasonably low level in different training parameters. The graph representation of the floorplan is quite complex in our current approach, which contains different types of edges and nodes with different attributes and semantic meanings. The tradeoff here is that more expressive graph representation of the housing designs would leave us more possibilities in the interactive design process and in combining computational design approaches. But it could also be more challenging for the models to capture the complex semantic information in the representation of housing designs since the generative models are mostly developed for applications with simple representations and big data (e.g., predictions of proteins or monocular structures).

To tackle this challenge, we will test different graph generation models (e.g., GraphRNN) to potentially identify the one with more expressive power. We would also try other graph representations for the layout to help the model better capture the embedded semantic and spatial relationships. In the meantime, the graph representation enables us to fix the generated layout through hardcoded rules. Also, different computational design methods can be implemented as complementary approach to the generative design by optimizing the results.

Another interesting finding is that small errors in generated layout graphs could result in noticeable contradictions in the corresponding generated floorplans. Even though the graph-based generative housing design pipelines allow more user interaction and control in every intermediate step, it could result in higher propagated error compared with end-to-end models.

Due to time and resource limitations, we only validate our models regarding compliance with user input and basic design rules, and the realism of floorplans. To bring our approach forward, we would scale up the testing and evaluation in two ways, including developing the qualitative matrices to evaluate the architectural quality of the generative housing design through a computational approach and developing a graphical user interface for larger-scale user evaluation. We would also benchmark the performance of our model against other state-of-the-art models in future research.

Since the graph representation is parametric, our graph-based approach could be compatible with various design tasks such as building equipment arrangement (Wang et al., 2019) and structural design (Zhao et al., 2024). It would be a promising direction to investigate how to integrate our graph-based method into a more generic design approach so as to ensure a more realistic and functional design solution.

Conclusions

In this paper, we presented an interactive AI framework for generative housing design, leveraging Graph Neural Networks (GNNs) and Deep Generative Models. Our approach, exemplified by the augmented DGMG framework for layout graph generation and Hetero-HouseGAN++ (HHGPP) for floorplan generation, demonstrates the viability of incorporating user constraints and requirements into the generative design process. Through an interactive interface, architects and stakeholders could actively participate in the design workflow, ensuring the production of feasible and user-centered housing design alternatives.

Moving forward, several avenues for improvement and exploration emerge. Firstly, the model is still insufficient in predicting the edge features and meeting all hard constraints in layout graph generation. We will test different models and representations for more expressive power in the learning for generative design. Future research could also look into computational methods as a complementary approach to fix and optimize the generated design alternatives based on graph representation. Moreover, user feedback and architectural quality metrics will be integrated into the evaluation process, providing a more comprehensive assessment of the generated designs.

This research would facilitate the housing design process through a collaborative and user-centered approach. The graph-based approach can also be integrated into more advanced design tasks (e.g., generative design of modular buildings based on BIM components library) due to its interoperability with the parametric data (e.g., IFC and IFCOWL), which would greatly facilitate the complex design process by harnessing the power of learning-based models and computational design approach. Despite only text-based interface is available for users currently, more advanced graphical interface could be further developed and easily embedded into domain applications (e.g., CAD and BIM) to facilitate the real-world design process.

Acknowledgements

This work was supported by TU Delft AI Labs. All codes can be found at: https://github.com/jlhofland/housing-design.

References

As, I., Basu, P., 2021. AI & architecture. Routledge.

Gan, V.J.L., 2022. BIM-based graph data model for automatic generative design of modular buildings. Autom. Constr. 134, 104062. https://doi.org/10.1016/j.autcon.2021.104062

Guo, X., Zhao, L., 2023. A Systematic Survey on Deep Generative Models for Graph Generation. IEEE Trans. Pattern Anal. Mach. Intell. 45, 5370–5390. https://doi.org/10.1109/TPAMI.2022.3214832

Hamzah, N., Ramly, A., Salleh, H., Tawil, N.M., Khoiry, M.A., Che Ani, A.I., 2011. The Importance of Design Process in Housing Quality. Procedia Eng., 2nd International Building Control Conference 20, 483–489. https://doi.org/10.1016/j.proeng.2011.11.191

He, F., Huang, Y., Wang, H., 2022. iPLAN: Interactive and Procedural Layout Planning. Presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7793–7802.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S., 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium, in: Advances in Neural Information Processing Systems. Curran Associates, Inc.

Hu, R., Huang, Z., Tang, Y., Van Kaick, O., Zhang, H., Huang, H., 2020. Graph2Plan: learning floorplan generation from layout graphs. ACM Trans. Graph. 39, 118:118:1-118:118:14. https://doi.org/10.1145/3386569.3392391

Ko, J., Ennemoser, B., Yoo, W., Yan, W., Clayton, M.J., 2023. Architectural spatial layout planning using artificial intelligence. Autom. Constr. 154, 105019. https://doi.org/10.1016/j.autcon.2023.105019

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., Battaglia, P., 2018. Learning Deep Generative Models of Graphs. https://doi.org/10.48550/arXiv.1803.03324

Nauata, N., Chang, K.-H., Cheng, C.-Y., Mori, G., Furukawa, Y., 2020. House-GAN: Relational Generative Adversarial Networks for Graph-Constrained House Layout Generation, in: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (Eds.), Computer

Vision – ECCV 2020, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 162–177. https://doi.org/10.1007/978-3-030-58452-8_10

Nauata, N., Hosseini, S., Chang, K.-H., Chu, H., Cheng, C.-Y., Furukawa, Y., 2021. House-GAN++: Generative Adversarial Layout Refinement Network towards Intelligent Computational Agent for Professional Architects. Presented at the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13632–13641.

Shabani, M.A., Hosseini, S., Furukawa, Y., 2022. HouseDiffusion: Vector Floorplan Generation via a Diffusion Model with Discrete and Continuous Denoising. https://doi.org/10.48550/arXiv.2211.13287

Upadhyay, A., Dubey, A., Mani Kuriakose, S., Agarawal, S., 2023. FloorGAN: Generative Network for Automated Floor Layout Generation, in: Proceedings of the 6th Joint International Conference on Data Science & Management of Data (10th ACM IKDD CODS and 28th COMAD), CODS-COMAD '23. Association for Computing Machinery, New York, NY, USA, pp. 140–148. https://doi.org/10.1145/3570991.3571057

Wang, K., Lin, Y.-A., Weissmann, B., Savva, M., Chang, A.X., Ritchie, D., 2019. PlanIT: planning and instantiating indoor scenes with relation graph and spatial prior networks. ACM Trans. Graph. 38, 132:1-132:15. https://doi.org/10.1145/3306346.3322941

Williamson, E., Wong, K., 2022. Valuing architectural work: The human effects. Archit. Aust. 111, 58–59. https://doi.org/10.3316/informit.604522451937359

Wu, W., Fu, X.-M., Tang, R., Wang, Y., Qi, Y.-H., Liu, L., 2019. Data-driven interior plan generation for residential buildings. ACM Trans. Graph. 38, 234:1-234:12. https://doi.org/10.1145/3355089.3356556

Zhao, P., Liao, W., Huang, Y., Lu, X., 2024. Beam layout design of shear wall structures based on graph neural networks. Autom. Constr. 158, 105223. https://doi.org/10.1016/j.autcon.2023.105223

Zheng, Z., Petzold, F., 2023. Neural-guided room layout generation with bubble diagram constraints. Autom. Constr. 154, 104962. https://doi.org/10.1016/j.autcon.2023.104962

Appendix

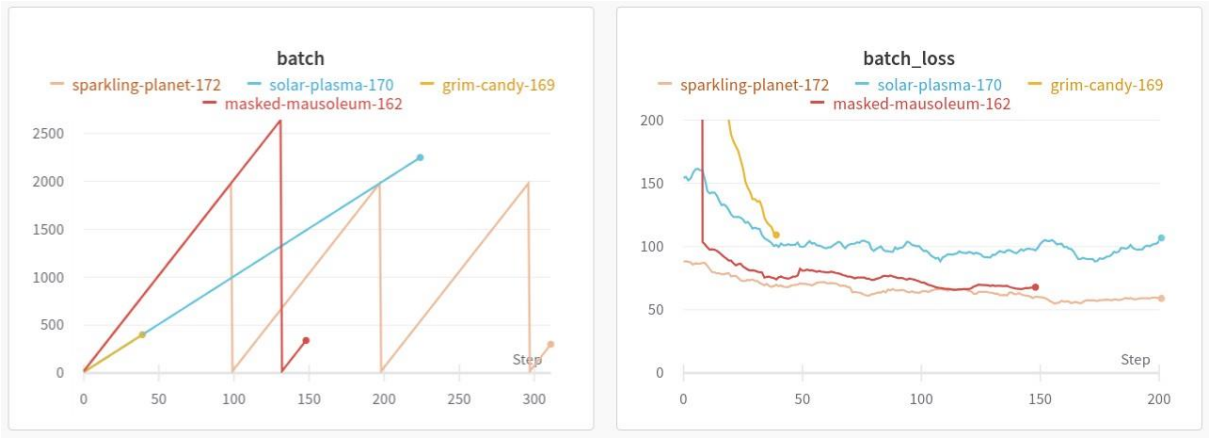


Figure A1: Training of layout graph generation model

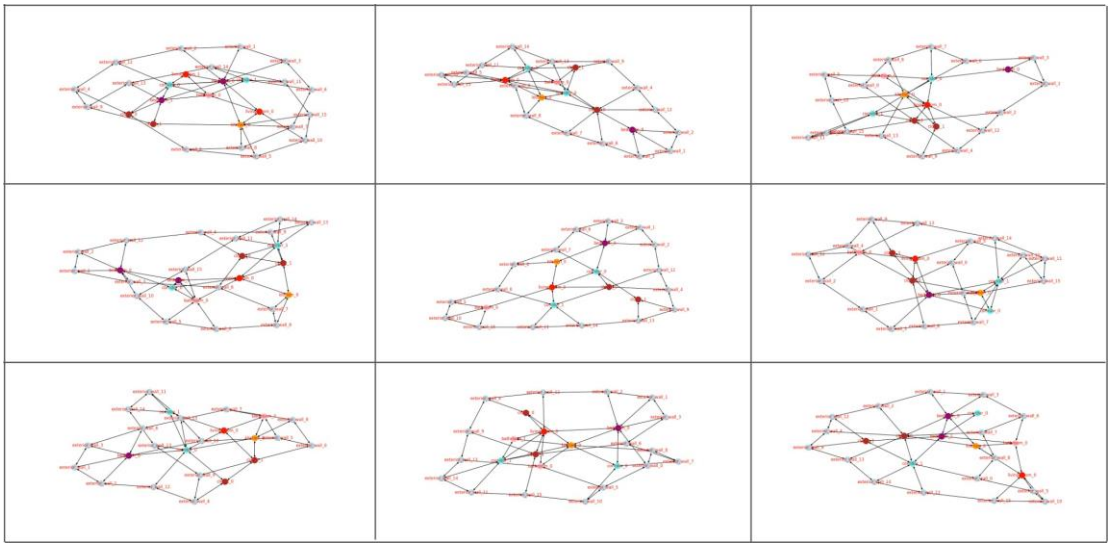
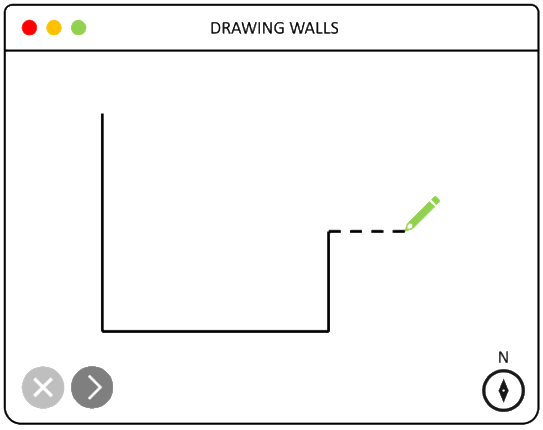
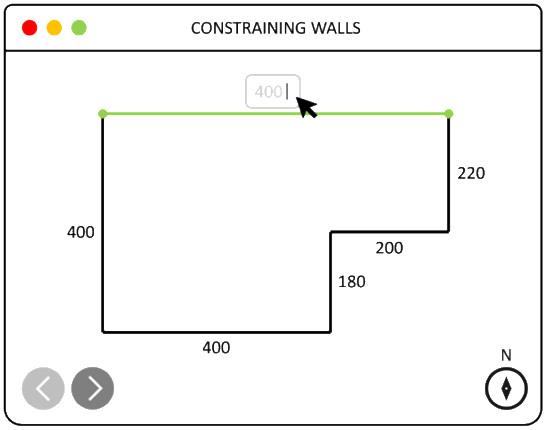


Figure A2: Sample of generated layout graphs

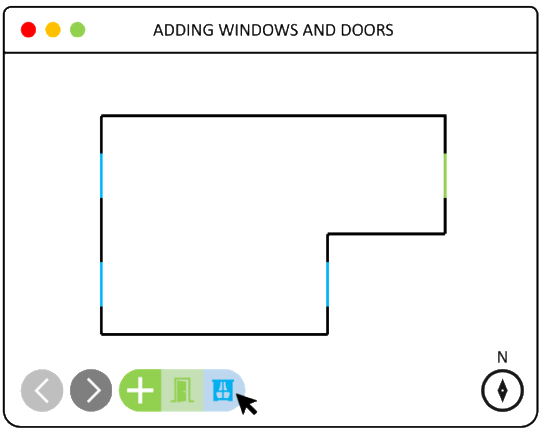
Figure A3: Training of floorplan generation model



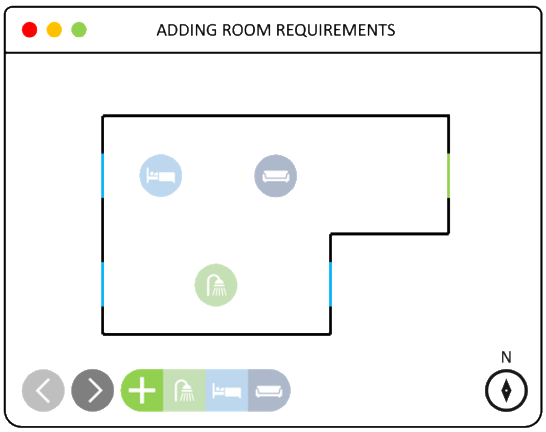
(a) Drawing outline of building using mouse. At each step we can use the buttons in the bottom to navigate through the user input steps.



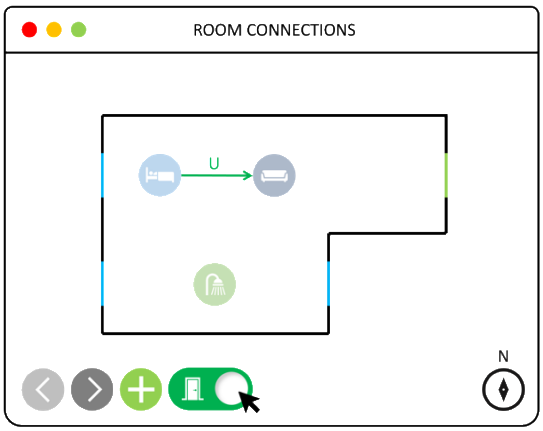
(b) Add length and angular constraints to each wall and corner. The interface could allow for natural angles but currently our model is only trained on angles of 90 degrees.



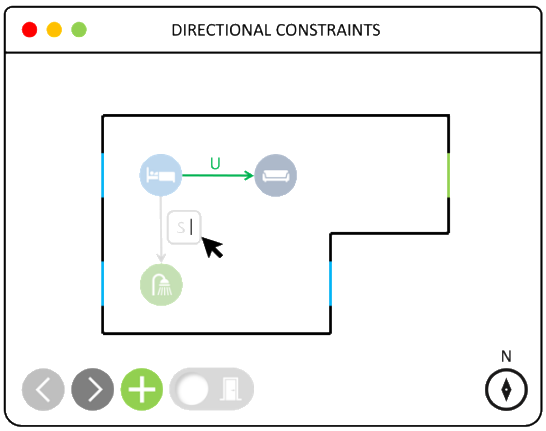
(c) Add facade features like doors and windows. In our case we only allow for the placement of doors as our model does not include window information.



(d) We can now add rooms, notice that the position of the nodes does not represent the spatial location. Our model only constrains the amount of baths, beds and living rooms but can be expanded.



(e) Adding edges (connections) between rooms. We can define wether or not the connection is an adjacency or door connection.



(f) We can also add an adjacency/door between rooms with a constrained relative direction. In the example, we say the bathroom should be south (S) of the bedroom without a door.

Figure A4: Envisioned graphical user interface