VNU Journal of Science: Com. Science & Com. Eng., Vol. 40, No. 2 (2024) 81-90

VNU Journal of Science:
Computer Science and Communication Engineering
Journal homepage: http://www.jcsce.vnu.edu.vn/index.php/jcsce

Original Article*

Deep Learning for Automated 3D Floor Plan Generation

Ma Thi Chau

Faculty of Information Technology, VNU University of Engineering and Technology,
144 Xuan Thuy, Cau Giay, Hanoi, Vietnam

Received 26 May 2024
Revised 29 July 2024; Accepted 10 December 2024

Abstract: This paper presents an optimized method for reconstructing 3D floor plans using user-defined boundaries and constraints for residential structures. This approach allows users to provide architectural constraints such as room types and quantities, as well as manual sketches or existing images of the house boundaries. Advanced deep learning algorithms are used to automatically partition the house boundaries and create a customized, optimized interior layout based on the users' architectural constraints. In the experiment phase, we integrated the Graph2Plan-based deep learning module, which converts the user-provided boundary data and architectural constraints into a structured 2D floor plan, automatically allocating and refining the rooms to ensure a harmonious spatial arrangement. The evaluation of the deep learning model's performance shows that this is a useful and time-saving solution for designers. Then, we utilized graphics and image processing techniques to generate the 3D floor plans. Based on this solution, we have developed a 3D floor plan generation application that provides a flexible and adaptive solution for individual home planning within defined boundaries. The application has been thoroughly tested to demonstrate its features, including the ability to meet users' architectural constraints, provide rapid response times, and offer a convenient user interaction experience.

Keywords: 3DFloorplan, Floorplan generation, layout graph, RPLAN dataset, house plan.

1. Introduction

The design and planning of residential spaces is a critical task that requires significant time and effort from architects and interior designers. Traditionally, the process of creating 3D floor plans for homes has been a manual and labor-intensive endeavor, relying on the

* Corresponding author.
E-mail address: chaumt@vnu.edu.vn

https://doi.org/10.25073/2588-1086/vnucsce.2848

expertise and creativity of human designer[1]. The complete design time is usually estimated to be 10 to 20 days. Within this, the time for designing the floor plan is estimated to be 3 to 5 days. However, advancements in deep learning and computer vision now offer opportunities to automate and optimize this process [1].

In this paper, we propose and discusses a deep learning-based approach for the automated generation of 3D floor plans for residential structures. By leveraging state-of-the-art deep learning algorithms, this method aims to streamline the floor plan design process, allowing users to quickly and easily create customized 3D floor plan that meet their specific architectural constraints and preferences. The key step of this approach lies in the integration of deep learning models that can intelligently analyze user-provided inputs, such as sketches, images, or spatial constraints, and then automatically generate an optimized 2D floor plan.

Our contributions are:

(i) An automated process for a 3D floor plan application, where we fine-tune the Graph2Plan model on the RP_LIF dataset, which is a combination of the RPLAN and LiFULL'sHome datasets. This serves as the deep learning core of the system to generate 2D floor plans.

(ii) A sub-process that generates 3D floor plans from the 2D floor plans by deploying computer graphics and image processing algorithms.

Through extensive experiments and evaluations, we demonstrate the effectiveness and efficiency of our deep learning-based approach in generating high-quality 3D floor plans that closely align with user requirements. The developed application not only reduces the time and effort needed for 3D floor plan design but also promotes greater creativity and flexibility in the design process.

[1] https://omshomesolutions.com/index.php/2021/09/21/thoi-gian-thiet-ke-nha-o/, (accessed on: August 24th2024)

2. Background and Related works

2.1. Background

This section outlines some related basic concepts and the key steps involved in generating a floor plan manually, which provides context for the content discussed in the subsequent sections.

Architectural constraints for a house refer to the various design requirements, limitations, and considerations that must be taken into account when planning and designing a residential building. In this paper, we are interested in the factors related to rooms such as type, number, and area.

A floor plan refers to a scaled, two-dimensional representation of the layout and arrangement of rooms, spaces, and structural elements within a building or a specific floor of a building.

A bubble diagram is a simple, diagrammatic tool used in the early stages of architectural design and space planning. It is a visual representation of the relationships and adjacency between different spaces or functions within a building or site.

A layout graph refers to the arrangement and visualization of spatial relationships and adjacency within a building or design. It is a conceptual and visual tool used in the early stages of the design process to help architects and designers explore, analyze, and communicate the functional and spatial organization of a project.

Besides, when designing a house or an architectural structure, generating a floor plan involves **several key steps**:

Gather information: Understand the project requirements and collect site details.

Create a space program: Determine required spaces, their functions, and desired relationships.

Develop a bubble diagram: Represent spaces as bubbles, arrange to show connections.

Establish circulation and zoning: Identify paths, group related spaces into zones.

Create the initial floor plan: Translate bubble diagram into a detailed layout.

Refine and iterate: Review, adjust, and experiment to optimize the design.

Finalize the floor plan: Add measurements, annotations, and produce final floor plan.

2.2. Related Works

2D Floor plan generating

There are several different approaches that have been explored for generating 2D floor plans from a graph-based layout representation. Here are some of the key methods:

Rule-based layout generation [2]: Define a set of spatial layout rules and constraints based on architectural best practices. Use these rules to guide the translation of the layout graph into a 2D floor plan, placing rooms, walls, doors accordingly. Example rules could be minimum room sizes, adjacency requirements, circulation paths, etc.

Optimization-based approaches [3]: Formulate the floor plan generation as an optimization problem, with the goal of minimizing violations of dimensional, functional, and aesthetic constraints. Use optimization techniques like evolutionary algorithms, simulated annealing, or mixed-integer programming to iteratively refine the floor plan layout. Objective functions can include metrics like space utilization, traffic flow, visual balance, etc.

Data-driven generative models [1, 4, 5]: Train machine learning models like variational autoencoders or generative adversarial networks on large datasets of existing floor plans. Use these generative models to produce new floor plan layouts that capture the statistical patterns and design features learned from the training data. The layout graph can be used as input to condition the generative model and guide the floor plan generation. This approach necessitates a significant amount of data for model training.

Grammar-based approaches [6]: Define a formal grammar that encodes the rules for translating a layout graph into a valid 2D floor plan. Use shape grammars, split grammars, or other grammatical formalisms to recursively generate and refine the floor plan layout. The layout graph provides the high-level structure that guides the application of grammar rules. This approach and the rule-based layout generation demand substantial expert knowledge, and this knowledge must be codified into formal rules.

Hybrid Approaches [7]: Combine multiple techniques, such as using optimization to refine the output of a generative model. Use the strengths of different methods to produce more robust and versatile floor plan generation capabilities. The choice of approach often depends on the specific requirements of the project, the available data, and the desired level of user control and customization. Researchers continue to explore new methods and combinations of these techniques to advance the state-of-the-art in automated 2D floor plan generation.

3D Floor plan generating

Typically, when 2D floor plan are available, specialized software is used to generate 3D floor plans through a series of steps: (i) *Extruding Walls* - The 2D outlines of the walls are extruded vertically to give them height and create the 3D geometry of the rooms and spaces. (ii) *Adding Floors and Ceilings* - The floors and ceilings are added between the extruded walls, defining the vertical dimensions of each room and story. (iii) *Incorporating Architectural Elements* Doors, windows, stairs, and other architectural features are added to the 3D model based on their locations in the 2D plan. (iv) Defining Material Properties - The surfaces of the walls, floors, ceilings, and other elements are assigned material properties like color, texture, and reflectivity to give the 3D model a realistic appearance. (v) Lighting and Rendering - Lighting sources are positioned, and rendering algorithms are applied to generate photorealistic images and walkthrough visualizations of the 3D

floor plan. This process allows the 2D floor plan information to be transformed into a 3D floor plan which provides a much more immersive and detailed representation compared to the initial 2D layout. In recent years, there have been several studies employing deep learning, but a great effort was required to secure a large amount of data for learning. In the present study [8], the authors used 3DPlanNet Ensemble methods incorporating rule-based heuristic methods to learn, and generated 110,000 3D vector data with a wall accuracy of 95% or more from 2D floor plan.

3. Proposal

To employ deep learning in generating 3D floor plan, the process of generating 3D floor plans from the architectural constraints provided by the user includes 3 steps as shown in Fig. 1: layout graph generating, 2D floor plan generating, and 3D floor generating.

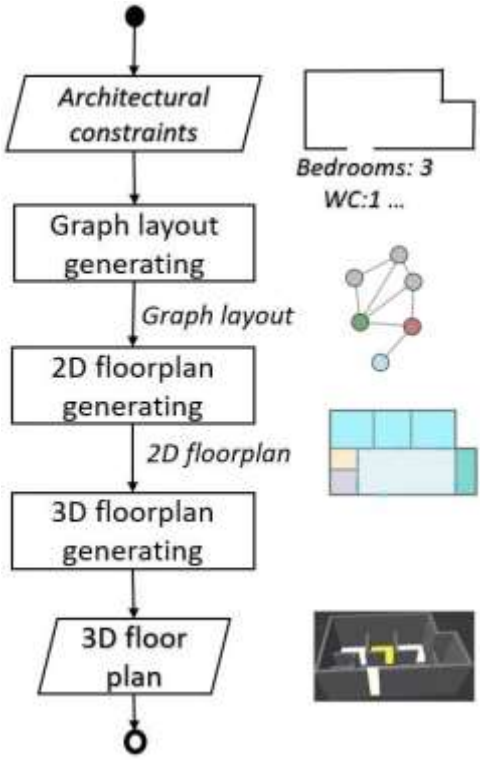

Fig. 1. The process of 3D floor plan generating.

Layout graph generating from user-specified constraints is an active area of research in architectural design automation. When architects create a floor plan, the generation of a bubble diagram is an intermediate step in the design process. For automated design systems, the representation of the user's design requirements is done using a layout graph, which is also referred to as a graph representation. The key idea is to use graph-based representations and generative models to create graph-based representations of the architectural constraints provided by the user. Representing a floor plan as a graph structure is a way of modeling the spaces and relationships between rooms, furnishings, and other components as a graph, with nodes representing individual elements and edges capturing their spatial and functional connections. For example, user requirements such as the number and types of rooms, their approximate sizes, and desired connections can be encoded as attributes of the nodes and edges in the graph. This graph-based approach has the ability to directly incorporate high-level user requirements into the generation process, leading to floor plans that better match user needs and priorities.

2D floor plan generating from the layout graph is a crucial step in this process. The node positions, node attributes, and edge connections in the layout graph are directly used to determine the shapes and placements of the various elements in the 2D floor plan, such as rooms, walls, doors, and other architectural features. However, this initial translation often requires further optimization to ensure the generated floor plan meets the necessary dimensional, functional, and aesthetic requirements. Optimization techniques, such as evolutionary algorithms or mixed-integer programming, are commonly employed to iteratively refine the floor plan layout, adjusting the sizes and positions of rooms and other components to better satisfy the specified design constraints.

This optimization-based approach allows the system to produce 2D floor plans that closely match the user's needs and preferences encoded in the original graph-based representation.

3D floor plan generating from a 2D floor plan is the process of transforming the 2D layout information into a comprehensive 3D digital model of the building. This conversion from 2D to 3D enables a far more immersive and detailed representation of the building design, which can then be utilized for a variety of important purposes- including further analysis and optimization of the layout, visualization and walkthroughs to better understand spatial relationships, construction planning and coordination of building elements, as well as intuitive 3D presentations to stakeholders. By converting the 2D floor plan into a full 3D floor plan, designers and engineers gain a much richer, more complete understanding of the building's layout and features, facilitating more informed decision-making throughout the design and construction phases.

4. Implementation and evaluation

4.1. Layout Graph Generating

The architectural constraints provided by the user include the number and types of rooms, the spatial relationships between them, as well as the overall house boundary. 13 room types are LivingRoom, MasterRoom, S econdRoom, GuestRoom, ChildRoom, S tudyRoom, DiningRoom, Bathroom, Kitchen, Balcony, S torage, Wall_in, and Entrance. The house boundary is then used to generate area constraints for the layout graph representation and to establish spatial geometry limits when generating the floor plan. With the architectural constraints, we initialize the graph G. We employed the *modelC* [1] to generate the layout graph (Fig. 2) from graph G and constraints *CTs*.

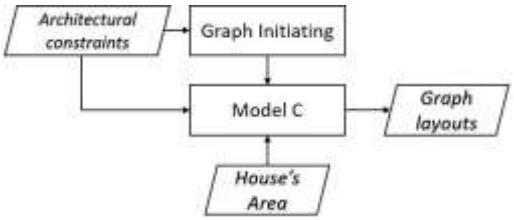

Fig. 2. Process of layout graph generating.

The pseudocode for the algorithm is as follows:

```
function ModelC(G, CT s)
{
# Initialize node positions randomly
pos = initialize_R_pos(G)
# Define the integrated cost function obj_funct = define_I_C_func(G, pos) # Define the constraint functions
c_ f = define_C_func(CT s)
# Solve the constrained optimization problem
opt_pos = solve_C_opt(ob j_ f unct, c_ f , pos)
return opt_pos }

function define_I_C_funct(G, positions)
{
# Define the integrated cost function cost = 0
for u, v in G.edges():
cost += kamada_cost_function(pos[u], pos[v], G[u][v])
cost += modified_cost_function(pos[u], pos[v], G[u][v])
return cost
}
```

The primary advantage of using the modelC is that it enables the user to constrain the positions of nodes within the layout graph and to provide suggested values for node locations. This increased flexibility does not come at the expense of high computational complexity.

4.2. 2D floorplan generating

Graph2Plan (G2P) [5], HouseGAN (HGan) [4] use different powerful architectures specifically designed to convert user input into a floorplan. We have utilized the G2P architecture (Fig. 3), the HGAN architecture (Fig. 4) and retrained them using a combined dataset RP_LIF consisting of RPLAN[2] and LiFULL's Home[3]. This combined dataset has produced the model parameters that we are currently using. To create dataset RP_LIF, we took 80,000 samples from dataset RPLAN that had been preprocessed by [5], including layout graphs and floor plans, and 50,000 samples from dataset LiFULLs Home that had been preprocessed by [4], including bubble diagrams and floor plans. We coded a dual module to generate bubble diagrams from layout graphs (for the RPLAN set) and generate layouts from bubble diagrams (for the LiFULL's Home set). In this way, dataset RP_LIF comprised 130,000 samples of the triplet of bubble diagrams, layout graphs, and floor plans.

G2P model

Input: The input consists of two components - building boundaries (B) and a layout graph (G). The building boundaries B are represented as a 128 128 image with three binary channels, which represent the interior, boundary, and door pixels. The layout graph G encapsulates nodes and edges, where each room i is described as a node ni = [ri, li, si], with ri representing the room category encoding, li the position vector representing the raw image position, and si the size vector conveying the room dimensions at different scales. The edge information eij captures the learned embeddings for pairwise relationships between nodes.

Output: The output of G2P is a 128 128 floor plan image (I) and two sets of room-bounding boxes.

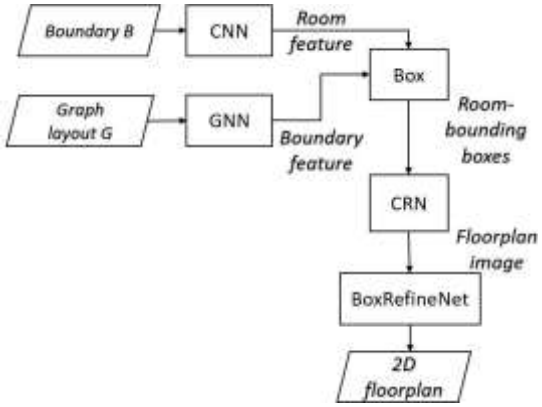

Fig. 3. G2P architecture.

The architecture uses a Graph Neural Network (GNN) to process the G-layout graphs and embed room features. Concurrently, an encoder is applied to the building boundary B to extract boundary features. The combination of these features is used by the Box network to generate the corresponding bounding boxes. Room boxes are expected to guide the compositional generation of room features using a Cascading Refinement Network (CRN) [9] to generate floor plan images I. Overlapping areas use a combination of the respective room's features. To capture global insights, the system uses additional refinement through BoxRefineNet, ensuring accurate room locations and dimensions. In short, G2P integrates GNN, CRN, and screening networks to translate user preferences into well-defined floor plans.

HGan model

Input: Given a bubble diagram, a node for each room is initialized with a 128-dimensional noise vector sampled from a normal distribution, concatenated with a 13-dimensional one-hot encoded room type vector. The result is a 141-dimensional vector.

Output: A feature volume is converted into a room segmentation mask by a 3-layer CNN network. The graph of these segmentation masks

[2] 2https://paperswithcode.com/dataset/rplan

[3] https://paperswithcode.com/dataset/lifull-home-s

will be passed to the discriminator during training. The room mask is fit with the tightest axis-aligned rectangle for each room to generate the floor plan.

4.3. 3D floor Plan Generating

In this section, we present the technique for constructing a 3D floor plan from a 2D floor plan (Fig. 5). The process begins by converting the input 2D floor plan image into a multi-level gray scale image. We then use the find contour algorithm [10] to identify the edges, including the boundary, walls, and doors, within the image. The boundary of the floor plan is determined by identifying the contour with the largest area. Next, we apply the watershed algorithm [11] to determine the edges that represent the walls. We then identify corner features [12] and connect them into line segments to form the rooms, using a threshold of coordinate differences. Additionally, doors are identified by connecting components into rooms using a different threshold.

	Layer	*Specification*
Generator	concat(z,t)	
	linear_reshape1	141 x 1024
	conv_mpn1	16 × 16 × 3 × 3
	upsample1	16 × 16 × 4 x 4
	conv_mpn2	16 × 16 × 4 x 4
	upsample2	16 × 16 × 4 x 4
	conv_leaky_relu1	16 x 256 x 3 x 3
	conv_leaky_relu2	256 x 128 x3 x3
	conv_tank1	128 x1 x3 x3
Discriminator	linear_reshape1(t)	10 x 8192
	concat(t,x)	
	conv_leaky_relu1	9 x 16 x3 x 3
	conv_leaky_relu2	16 x 16 x3 x 3
	conv_leaky_relu3	16 x 16 x3 x 3
	conv_mpn1	
	downsample1	16 x 16 x3 x 3
	conv_mpn2	
	downsample2	16 x 16 x3 x 3
	conv_leaky_relu1	16 x 256 x 3 x 3
	conv_leaky_relu2	256 x 128 x3 x 3
	conv_leaky_relu3	128 x128 x 3 x 3
	pool_reshape_linear1	141 x 1

Fig. 4. HGan architecture.

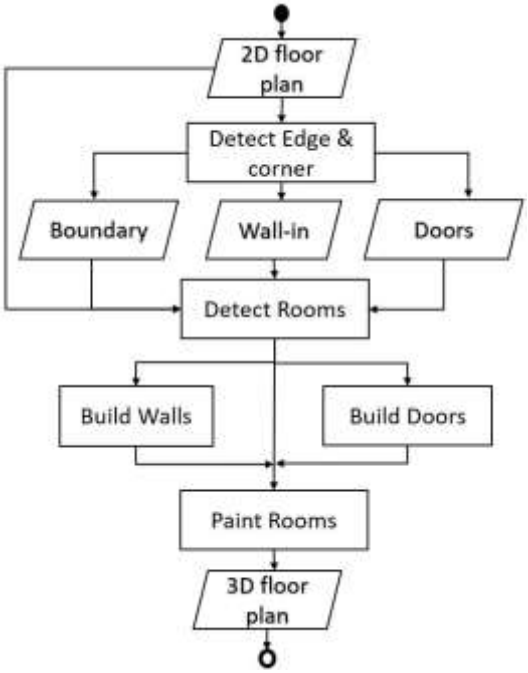

Fig. 5. 3D floor plan generating.

4.4. Evaluation and application

Evaluation: After training the models G2P and HGan with 2 datasets the RPLAN - similar to the one used in the G2P model and the RP_LIL, the results are shown in Table 1 with batch size 20, epoch 101.

The training process ran stably with both datasets. The loss function was relatively good, and the Intersection over Union (IoU) score was around 0.65, which is a quite positive and good result. The IoU metric is commonly used to evaluate the performance of object detection models, with a higher score indicating better alignment between the predicted bounding boxes and the ground truth.

Tab. 1. Results for G2P and Hgan

Test info	**G2P RPLAN** (baseline)	**G2P RP_LIF**	**HGan RP_LIF**
Train time	41 hours	59 hours	65 hours
Loss	0.0001	0.0001	0.0002
IOU	0.64	0.65	0.67
Time per floor_plan	0.4-1s	0.3-1s	0.1-0.6

The use of the combined RP_LIL dataset likely contributed to the model's improved performance compared to using the RPLAN dataset alone. By incorporating a more diverse set of floor plans, the model was able to learn more robust representations and generalize better to unseen data. The results demonstrate the effectiveness of the model architecture and the benefits of training on a comprehensive dataset for this floor plan generation task.

Currently, utilizing advanced computer graphics and image processing techniques, we have created a simple 3D floor plan model featuring enclosing walls, door orientations, and functional room divisions represented by distinct colors.

Application: We have developed an application that automatically generates a 3D floor plan from the architectural constraint requirements provided by users. The application allows users to upload an image of the house boundary or to draw the house boundary themselves. Users also input information about the room types, sizes, and connections to other rooms (Fig. 6). We deploy the 2D floor plan generation module using a pre-trained G2P (as mentioned above) model. The parameters extracted from the boundary, room, and door identification in 2D floor plan are supplemented with height information to prepare the input for 3D construction. Based on the stored parameters from the previous steps, we develop modules for constructing the walls and doors in dedicated 3D software. Along with the parameters used for constructing the walls and doors, and distinguishing colors in different areas of the input image representing different room types, we create a module to color the delineated areas of the different rooms in the final 3D floor plan (Fig. 7). In the Fig. 7, we have automatically generated 3D floor plans with the input of architectural constraints and the house's boundaries. For each 3D floor plan, we have identified the location of doors, the location of windows in each room, colors indicating the type of room for each room, and the scale of the room areas as per the 2D floor plan.

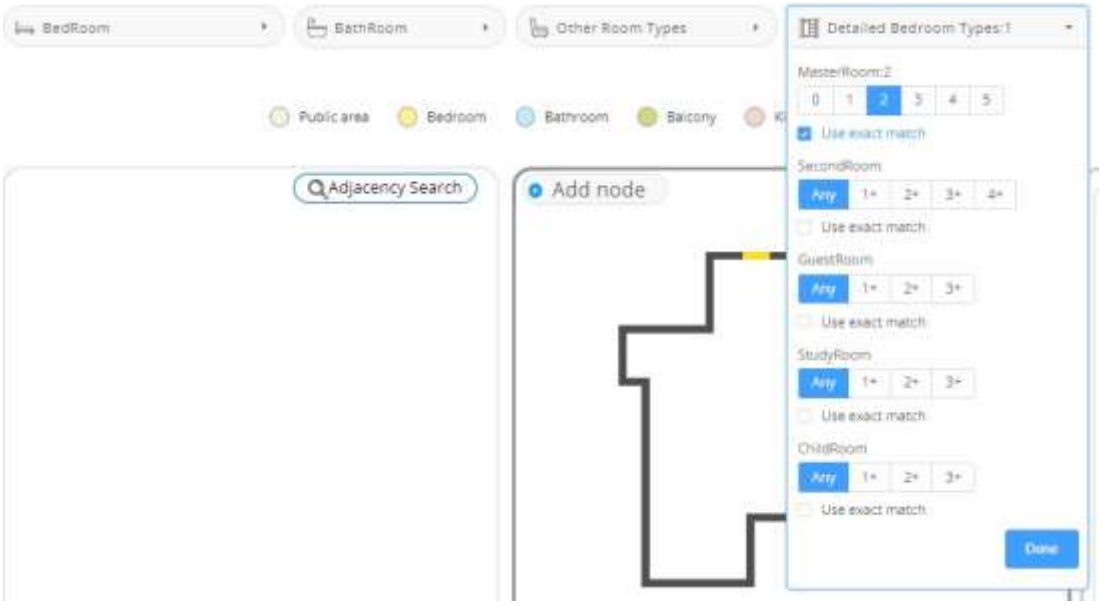

Fig. 6. User-provided architectural constraint descriptions.

5. Conclusion

The proposed approach for generating 3D floor plans from architectural constraints provided by the user represents a promising direction for advancing architectural design automation. The three-step process of graph layout generating, 2D floor plan generating, and 3D floor plan generating leverages graph-based representations and generative models to directly incorporate the user's high-level design requirements into the generation process.

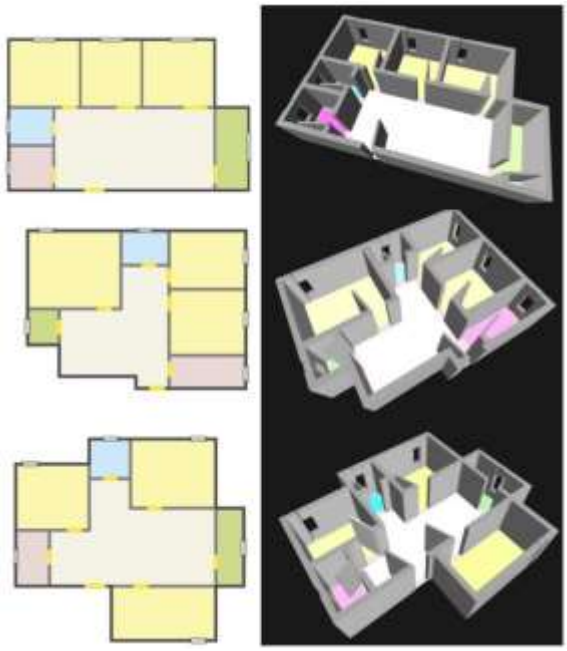

Fig. 7. 3D floorplan results.

The key advantages of this approach are its ability to model the spatial and functional relationships between rooms, and other components using a graph-based representation, and the use of optimization techniques to refine the 2D floor plan layout to better satisfy the specified design constraints and reduce time process. By converting the 2D floor plan into a comprehensive 3D model, designers and engineers can gain a much richer, more immersive understanding of the building's features, enabling more informed decision-making throughout the design and construction phases. With this approach, we built an application that generates 3D floor plans from user-provided architectural constraint descriptions.

In summary, this work demonstrates the potential of combining graph-based representations, generative models, and optimization techniques to automate the generation of 3D floor plans that closely match user requirements. As architectural design continues to evolve, this type of approach could significantly streamline the design process and empower architects, engineers, and stakeholders to explore a wider range of design possibilities more efficiently.

6. Acknowledgement

This work has been supported by VNU University of Engineering and Technology under project number CN24.14 "Solution of 3D floor plan generation".

References

[1] Y. Shi, M. Shang, Z. Qi, Intelligent Layout Generation Based on Deep Generative Models: A Comprehensive Survey, Information Fusion 100 (2023) 101940. doi: https://doi.org/10.1016/j.inffus.2023.101940.

[2] T. Tutenel, R. Bidarra, R. Smelik, K. J. de Kraker, Rule-Based Layout Solving and its Application to Procedural Interior Generation, in: Proceedings of the CASA Workshop on 3D Advanced Media in Gaming and Simulation (3AMIGAS), 2019.

[3] P. Merrell, E. Schkufza, V. Koltun, Computer-Generated Residential Building Layouts, ACM Transactions on Graphics 29 (2010) 1–12. doi:10.1145/1866158.1866203.

[4] N. Nauata, K.-H. Chang, C.-Y. Cheng, G. Mori, Y. Furukawa, House-gan: Relational Generative Adversarial Networks for Graph-Constrained house Layout Generation, in: European Conference on Computer Vision, Springer, 2020, pp. 162–177.

[5] R. Hu, Z. Huang, Y. Tang, O. van Kaick, H. Huang, Graph2plan: Learning Floorplan Generation From Layout graphs, ACM

Transactions on Graphics 39 (2020) 118:1 – 118:14. doi:10.1145/3386569.3392391.

[6] X. Wang, Y. Liu, K. Zhang, A Graph Grammar Approach to the Design and Validation of Floor Plans, The Computer Journal 63 (2020) 137–150. doi:10.1093/comjnl/bxz002.

[7] M. Nisztuk, P. Myszkowski, Hybrid Evolutionary Algorithm Applied to Automated Floor Plan Generation, International Journal of Architectural Computing 17 (2019) 260–283. doi:10.1177/1478077119832982.

[8] S. Park, H. Kim, 3dplannet: Generating 3d Models from 2d Floor Plan Images using Ensemble Methods, Journal of Electronic 10 (2021) ’. doi:10.3390/electronics10222729.

[9] Q. Chen, V. Koltun, Photographic Image Synthesis with Cascaded Refinement Networks, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1520–1529. doi:10.1109/ICCV.2017.168.

[10] S. Suzuki, K. Abe, Topological Structural Analysis of Digitized Binary Images by Border Following, Comput. Vis. Graph. Image Process. 30 (1985) 32–46.

[11] J. Cousty, G. Bertrand, L. Najman, M. Couprie, Watershed cuts: Minimum Spanning Forests and the Drop of Water Principle, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (8) (2009) 1362–1374. doi:10.1109/TPAMI.2008.173.

[12] C. G. Harris, M. J. Stephens, A Combined Corner and Edge Detector, in: Alvey Vision Conference, 1988.