Machine Intelligence Research
www.mi-research.net

22(2), April 2025, 368-385
DOI: 10.1007/s11633-024-1498-0

Auto-3D-house Design from Structured User Requirements

Minkui Tan[1†] Qi Chen[2†] Zixiong Huang[1] Qi Wu[2] Yuanqing Li[3] Jiaqiu Zhou[1]

[1] School of Software Engineering, South China University of Technology, Guangzhou 510006, China

[2] Faculty of Engineering, Computer and Mathematical Science, University of Adelaide, Adelaide SA 5005, Australia

[3] School of Automation Science and Engineering, South China University of Technology, Guangzhou 510006, China

Abstract: We study the task of automated house design, which aims to automatically generate 3D houses from user requirements. However, in the automatic system, it is non-trivial due to the intrinsic complexity of house designing: 1) the understanding of user requirements, where the users can hardly provide high-quality requirements without any professional knowledge; 2) the design of house plan, which mainly focuses on how to capture the effective information from user requirements. To address the above issues, we propose an automatic house design framework, called auto-3D-house design (A3HD). Unlike the previous works that consider the user requirements in an unstructured way (e.g., natural language), we carefully design a structured list that divides the requirements into three parts (i.e., layout, outline, and style), which focus on the attributes of rooms, the outline of the building, and the style of decoration, respectively. Following the processing of architects, we construct a bubble diagram (i.e., graph) that covers the rooms′ attributes and relations under the constraint of outline. In addition, we take each outline as a combination of points and orders, ensuring that it can represent the outlines with arbitrary shapes. Then, we propose a graph feature generation module (GFGM) to capture layout features from the bubble diagrams and an outline feature generation module (OFGM) for outline features. Finally, we render 3D houses according to the given style requirements in a rule-based method. Experiments on two benchmark datasets (i.e., RPLAN and T3HM) demonstrate the effectiveness of our A3HD in terms of both quantitative and qualitative evaluation metrics.

Keywords: Automated house design, user requirements understanding, outline processing, layout generation, graph feature generation.

Citation: M. Tan, Q. Chen, Z. Huang, Q. Wu, Y. Li, J. Zhou. Auto-3D-house design from structured user requirements. *Machine Intelligence Research*, vol.22, no.2, pp.368–385, 2025. http://doi.org/10.1007/s11633-024-1498-0

1 Introduction

Living in a safe, warm, and loving house is everyone′s requirement for their dream house. Generally, designing a house requires architects to complete the house with their professional knowledge, which is very time-consuming. Even certificated architects often take a couple of days or even several weeks to design a floor plan. Thus, it would be fantastic if people could design their own houses by themselves. To this end, as shown in Fig. 1, we propose a new automatic house design method, called auto-3D-house design (A3HD), aiming to allow users to design their own houses in a time-saving way. However, it poses two main challenges during the designing of such an automatic system.

First, understanding user requirements is non-trivial as users can hardly provide high-quality and accurate requirements. It is important to provide accurate requirements since automated house design is a user-oriented task. But many people may provide their requirements about the feeling of a dream house, which is difficult to understand, e.g., "I wish I had a warm and loving house". Besides, the outline constraint (i.e., a designed floor plan can not exceed the outline) needs to be considered since house layout design is often fixed-outline design. Although the automatic layout generation methods[1–4] produce good-looking layouts, they may not be able to analyze the user requirements well. For example, HouseGAN[1] and HouseGAN++[2] require the topology of rooms as input while the generated layouts have casual outlines. HPGM[3] produces layouts based on the linguistic requirements only, which, however, would introduce some irrelevant information. Para et al.[4] propose a generative model which requires no user input. As the generative model takes the user requirements as soft constraints, it can not guarantee that the requirements will be satisfied. Hu et al.[5] propose a learning framework to simplify the user requirements for house plan generation. However, they still require the users to provide the outline image of the layout dataset. It is significant if the problem can be simplified and would not increase the workload of humans.

Second, it is hard to make use of the outline due to its irregular shapes. The outline is important for the genera-

Research Article
Manuscript received on August 29, 2023; accepted on February 1, 2024; published online on January 7, 2025
Recommended by Associate Editor Hao Dong
Colored figures are available in the online version at https://link.springer.com/journal/11633
*These authors contributed equally to this work

tion of a floor plan since it directly limits the position and size of the rooms in the floor plan. However, many automated house design methods[1–3] ignore the outline and can not satisfy the outline constraint. Moreover, how to represent outlines and exploit the information of outlines with irregular shapes is a difficult problem. To this end, many existing methods[5, 6] process the outline as an image, which, however, may introduce much irrelevant information derived from zero-padding (i.e., an image filled with zeros except for the outline). Based on these outline images, they directly use convolutional neural network (CNN) models to extract the corresponding features, which results in a high computational cost (see the results in Table 1). Thus, how to process and exploit the outline requirements remains an open question.

In this paper, we propose an auto-3D-house design (A3HD) that generates 3D houses based on a series of structured user requirements automatically. To address the first challenge, we reconstruct and simplify the user requirements as structured requirements, which are divided into three parts: layout (i.e., the number of different rooms with their corresponding types), outline, and style, to reduce the difficulty of understanding requirement. Considering the users' feelings about their dream houses, we model them as style requirements and show the 3D houses for better understanding. In this way, our structured user requirements concentrate on useful information and ensure that the user requirements are clear and simple, which does not increase the workload on users. To address the second challenge, we explore a new outline representation method and propose an outline feature generation module (OFGM) to extract the outline features. Instead of processing it as an image[5, 6], we take the outline into sequential data/points, aiming to leverage the outline information more effectively. In addition, we convert the layout requirements as a bubble diagram and then design a graph feature generation module (GFGM) for the layout features. After that, we produce a floor plan (i.e., layout) based on the layout features and outline features. Last, we obtain a 3D house plan from the floor plan according to the style requirements. Extensive experiments on RPLAN and T3HM datasets demonstrate the effectiveness of our A3HD in terms of both quantitative and qualitative evaluation metrics.

We highlight our principal contributions as follows:

1) As it is hard for users to describe their requirements professionally, we simplify and reconstruct the user requirements as a structured list, w.r.t., layout, outline, and style, which alleviates the difficulties of requirement understanding for both the users and the automatic designing system.

2) To yield 3D houses from the structured user requirements automatically, we propose an automatic house designing framework, named auto-3D-house design (A3HD), which first generates a 2D-floor plan based on the layout and outline requirements and subsequently renders the generated floor plan according to the style requirements.

3) When generating a floor plan, unlike many existing works that consider the outline as an image, we take the outline into sequential data/points, which can represent any shape of outline and exclude the irrelevant information (e.g., useless pixels in the image format). Based on that, we propose an outline feature generation module (OFGM) to extract the outline feature along with the order of these points.

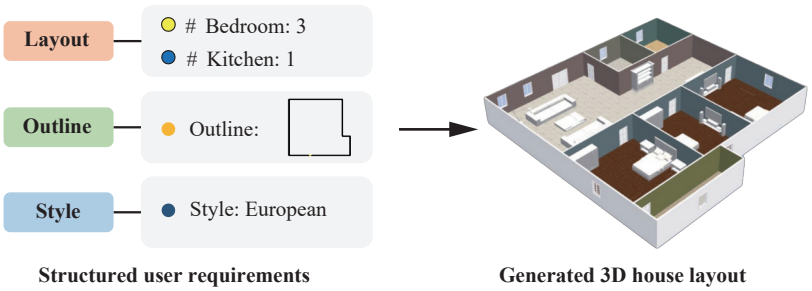


Fig. 1 An example of the 3D house generated using our A3HD. We divide the user requirements into a structured format, which consists of three parts: layout, outline, and style. We seek to design a 3D house automatically according to the user requirements

Table 1 Effectiveness of the sequential outline. "Images&CNN" means that we consider the outline as images and use the CNN to extract the outline features.

Datasets	Methods	IoU↑	BIoU↑	MAdds	Params
RPLAN[6]	Images&CNN	0.667	0.863	57.12 M	5.34 M
	A3HD (ours)	**0.731**	**0.902**	**5.45 M**	5.47 M
T3HM[3]	Images&CNN	0.627	0.816	57.12 M	5.34 M
	A3HD (ours)	**0.653**	**0.852**	**5.45 M**	5.47 M

2 Related work

2.1 Layout generation

Layout generation has been an active area of research in many domains, including house layout generation[3, 5, 7], scene image generation[8–12], visual-textual presentation layout generation[13, 14], layout reconstruction from image[15–18], 3D indoor scene design[19–24], 3D layout reconstruction from a single RGB image[25–30]. To generate a scene image, Ashual and Wolf[8] use a graph convolutional network to capture the relationship among different objects in a scene graph. For layout reconstruction from an image, Lv et al.[18] vectorize the floor plan by recognizing the structure, type, and size of the room in a floor plan image, and provide vectorized 3D reconstruction results of the residential floor plan. Focusing on synthesizing the indoor scene, Yang et al.[24] use Bayesian scene optimization to generate 3D scenes combined with prior distributions. To reconstruct a 3D layout from a single image, Yan et al.[31] propose a topology anchor point optimization to estimate the 3D layout of an indoor scene. In this paper, we focus on house layout generation from user requirements such as graphs and outlines.

2.2 Graph-based house layout generation

Many automatic house design methods[1–3, 32] focus on generating a house layout from the graph (the topology of the layout). Based on a set of high-level requirements such as the size range of each room in the layout, Merrell et al.[32] use a Bayesian network trained on real-world data to synthesize an architectural program and generate a 3D house model. More recently, Chen et al.[3] propose a house plan generative model to automatically generate a 3D house from a linguistic expression, which introduces some irrelevant information. In addition, Nauata et al.[1, 2] propose a graph-constrained generative adversarial network to generate a house layout from the input bubble diagram. However, these methods require inaccurate or complex user requirements as input and ignore the important outline constraint in real-world applications. In this paper, we simplify the user requirements by categorizing them into three components: layout, outline, and style. This approach allows us to offer more accurate inputs to our model, resulting in more deterministic outcomes. This improvement enhances the performance of automated house design methods without adding to the workload for users.

2.3 Outline-based house layout generation

There are many automatic house design methods[4–6, 33–35] that seek to generate a house layout under the outline constraint. Based on the given outline only, RPLAN[6] generates residential buildings without any high-level constraints. Besides, with sparse user requirements, Graph2Plan[5] combines generative modeling and user-in-loop designs for layout generation. However, these methods represent the outlines as outline images and use a convolutional neural network to extract the outline feature, which introduces much irrelevant information and increases the computational cost. Considering diverse user input, Para et al.[4] propose a transformer-based framework[36–38] to produce good-looking layouts. They use user requirements as a soft constraint, however, the condition is not guaranteed to be satisfied by their methods. In this paper, we seek to address the above challenges by introducing a novel outline representation method that interprets the outline requirement as a sequential outline, eliminating the need for extracting outline features through neural networks. This approach allows us to simplify our entire pipeline without introducing irrelevant information, as observed in previous works like [5, 6].

3 User requirements structuring & understanding

In this section, we design a structured list of user requirements to reduce the difficulty of requirement understanding. Moreover, to effectively exploit the outline information, we propose a novel outline representing method that treats the outline as a sequential outline. As for house layout generation, we parse the structured requirements into bubble diagrams (see an example in Fig. 2) based on house datasets.

Notation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a bubble diagram (i.e., graph), where $\mathcal{V}$ denotes the set of nodes (i.e., rooms) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ is a set of directed edges. Here, $\mathcal{R}$ refers to the relative positions between different rooms.

Problem definition. Given a set of user requirements, the automated house design task aims to automatically generate 3D houses that meet all the input requirements. To simplify the procedure of requirements understanding, our A3HD divides user requirements into three parts: layout requirements, outline requirements, and style requirements, which focus on the attributes of rooms, the outline of the building, and the style of decoration, respectively.

3.1 User requirements structuring

To process the complicated and tedious user requirements, we reconstruct the user requirements as structured requirements which are divided into layout, outline, and style. We seek to design simplified user requirements that focus on useful information for layout generation without increasing users′ workload.

Layout requirements. Inspired by the house layout design process, after consultation with professional architects, the users only need to focus on the functionality

and design rationality of the house layout. Therefore, to allow users to design a house layout without any architectural knowledge, we define the layout requirements as the number of different rooms and their corresponding types, which will not increase the workload of users while ensuring simplicity and clarity.

Outline constraint. It would not increase the workload on users to provide outline requirements since the automated house design task is often a fixed-outline task. To make sure that the generated layout satisfies the outline constraint and exploits the outline information, we require the users to provide the outline and process the outline as a sequential outline. Our structured requirements understanding and 2D layout generation are both based on the sequential outline, which will be introduced in detail below.

Style requirement. As we all know, people perceive the world in 3D. Thus, to provide a better understanding, we also generate the 3D house for users. To satisfy the users′ feelings about their dream houses, we process them as style requirements. Specifically, we design the decoration schemes for 3D house visualization. In the designing of the decoration schemes, we collect the floor texture and wall texture designed by the architects for different rooms such as the study and washroom (see an example in Fig. 3). In this way, we obtain twelve texture schemes which include “American”, “Chinese”, “European”, “Jane European”, “Japanese style”, “Mediterranean”, “Modern”, “Neoclassical”, “New Chinese”, “Northern European”, “Pastoral” and “Southeast Asia”. With a list of decoration schemes, we provide 3D house visualization for users to choose their favorite house.

3.2 Sequential outline

To effectively exploit the outline information, we propose a new outline representing method that treats the outline requirement as a sequential outline. Traditionally, it is difficult to exploit outline information since the outlines have irregular shapes. To this end, existing methods[5, 6] represent the outline requirement as images with a uniform format. As shown in Fig. 4(a), an outline image is filled with zero values except for the outline. However, the zero-padding image representation of an outline often introduces some irrelevant information that makes it difficult to effectively extract the outline feature. Moreover, existing methods use a convolutional neural network to process the outline image directly which can incur unnecessary computational costs (see results in Table 1).

Hence, we seek to explore a new outline representa-

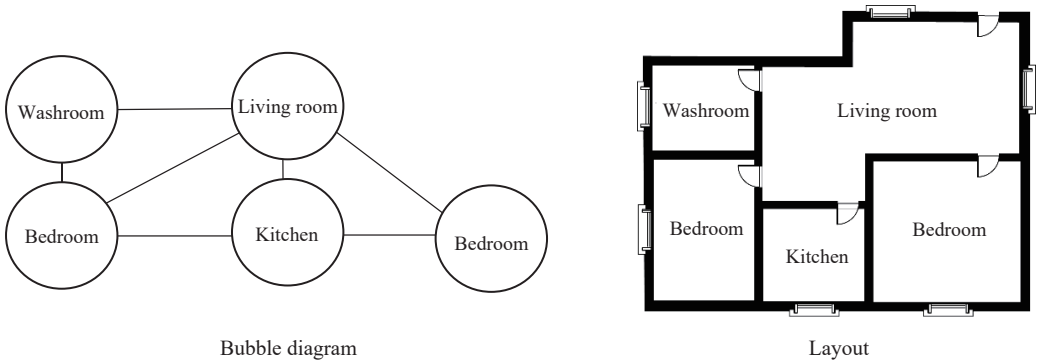


Fig. 2 An example of bubble diagram, i.e., graph (left) and corresponding house layout (right) on the house datasets

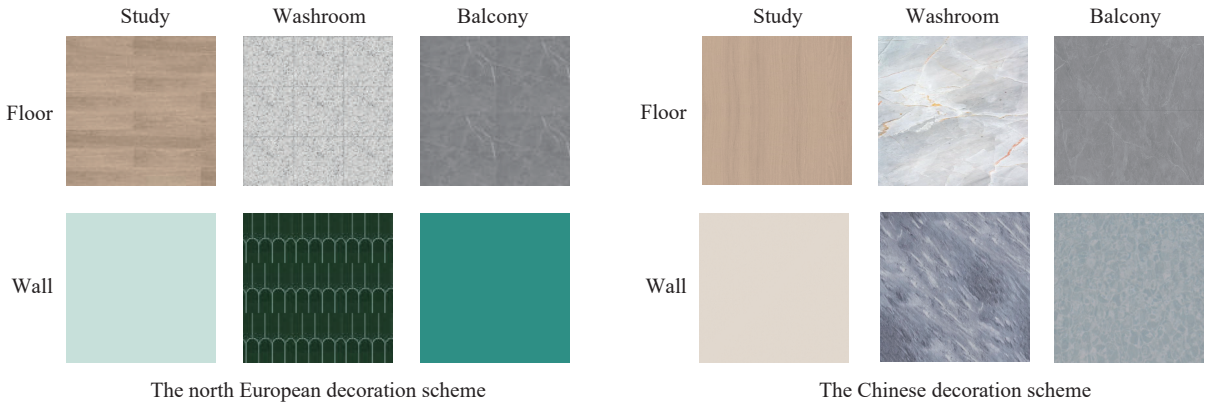


Fig. 3 Different decoration schemes used in our A3HD. We provide floor textures and wall textures designed by architects from different countries for different rooms. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

tion method. We observe that different orders between vertices of the outline refer to different polygonal shapes (see an example in Fig. 4(b)). The outline information may correlate with the order of the vertices in the outline. Motivated by this, we represent an outline as a sequential outline (i.e., a set of vertices in clockwise order). Mathematically, we denote the outline as $\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_N\}$, where N is the number of points in the outline and each point $\boldsymbol{p}_i = (x_i, y_i) \in \mathcal{P}$ denotes the position coordinates. Compared with an outline image, a sequential outline is a simple but effective way to represent the outline without introducing irrelevant information.

3.3 Structured requirements understanding

An overview of the structured requirement understanding is shown in Fig. 5. Given user requirements, we first select graph-outline pairs according to the requirements from the existing house datasets, e.g., T3HM[3] and RPLAN[6], which are designed by human architects. Then, we obtain a score by measuring the similarity between the outlines in the selected pairs and the input outline. According to the score, we choose the graphs with top-k scores and adjust the chosen graphs based on the input outline. We present more details as follows.

Graph selection from house datasets. To generate diverse floor plans, we select many different graphs from existing house datasets, such as RPLAN[6] and T3HM[3]. In house datasets, although different graphs are designed for different layouts, the adjacency of different graphs still satisfies the realism of house layouts (see results in Fig. 6). Thus, according to the layout requirements, we first select different graphs (with corresponding outlines) that satisfy the required number and types of rooms. Note that we only focus on whether the graphs cover the required rooms (i.e., type and the number).

Graph sorting by similarity ranking. To select the graphs that are suitable for the given outline, we design a ranking score and sort the above graph-outline pairs based on the score. To this end, we seek to meas-

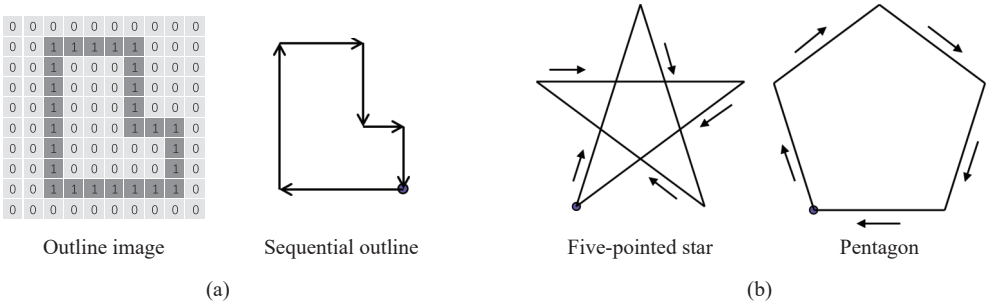


Fig. 4 Overview of sequential outline. (a) Comparison of outline image with sequential outline. The outline image is filled with zeros except for the outline while a sequential outline would not introduce irrelevant information. (b) An example of different polygonal shapes. The purple point represents the start point and the arrow denotes the build direction of different polygons. Following different build directions, five vertices of an outline can represent a five-pointed star or a pentagon.

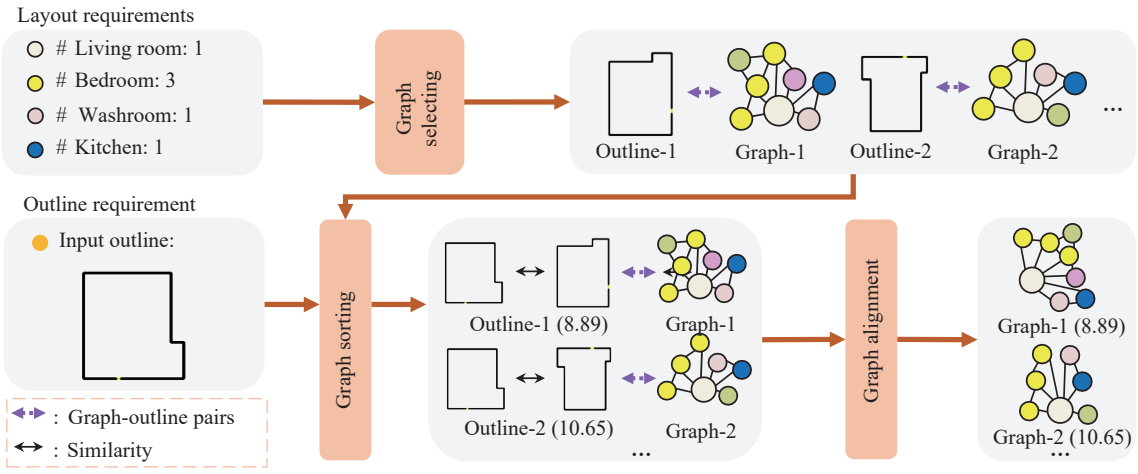


Fig. 5 Overview of structured requirements understanding. Based on the layout requirements, we first select the graph-outline pairs from the existing house datasets designed by architects. Then, we sort the graph based on a score, which compares the similarity between the given outline and the outline in selected graph-outline pairs. Last, we choose the graphs with the top-k scores and use graph alignment to align the graph with the outline. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

ure the similarity between the outlines in graph-outline pairs and the input outline. We use a turning function[5, 39] to extract polygonal features of each outline $\mathcal{P}$. In Fig. 7(a), we start the sequence on one side of the front door and record the angles in clockwise order. The polygonal feature records the accumulation of the angles corresponding to each point and the distances between points. For convenience, we denote the polygonal features of an outline as $\mathcal{A} = \{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \cdots, \boldsymbol{\omega}_N\}$, where N is the number of points in the outline. Each point $\boldsymbol{\omega}_i \in \mathcal{A}$ is represented as $\boldsymbol{\omega}_i = (\alpha_i, \phi_i)$, where $\alpha_i \in [0, 1]$ denotes an accumulated distance from the start (1st) point to the i-th point. $\phi_i \in [0, 2\pi]$ is an accumulation of angles from the 1st point to the i-th point. Mathematically, the value of α_i can be calculated as

$$\alpha_i = \begin{cases} \alpha_{i-1} + d_i/d, & \text{if } i > 1 \\ 0, & \text{if } i = 1 \end{cases} \tag{1}$$

where d_i is the distance between the points $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_{i-1}$ while d represents the perimeter of the outline.

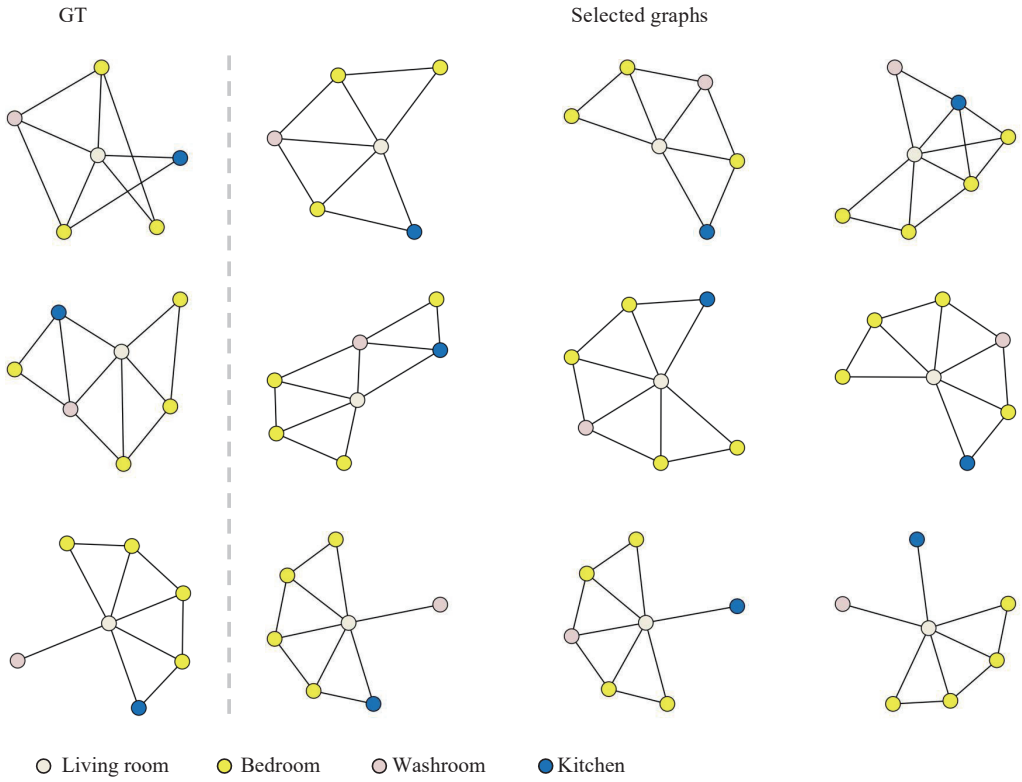


Fig. 6 Visualization of graphs selected from house datasets and corresponding ground-truth graphs. Different nodes represent different rooms. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

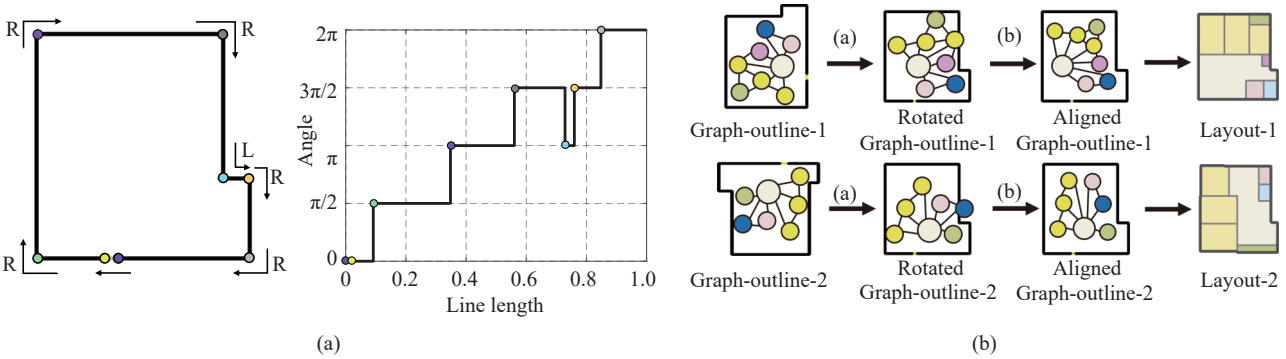


Fig. 7 Overview of graphs alignment. (a) An example of outline and its corresponding polygonal feature extracted by the turning function. The purple point represents the start point in the front door. "R" means turning in clockwise order while "L" refers to turning in anti-clockwise order. (b) Examples of aligning the selected graph-outline pairs with the given outline. Given the selected graph-outline pairs, we follow step (a) and step (b) to align the graphs with the outline. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

$$\phi_i = \begin{cases} \varphi_{i-1} + 1(\boldsymbol{\omega}_i, \boldsymbol{\omega}_{i-1}) \times \dfrac{\pi}{2}, & \text{if } i > 1 \\ 0, & \text{if } i = 1 \end{cases} \tag{2}$$

where $1(\boldsymbol{\omega}_i, \boldsymbol{\omega}_{i-1})$ is an indicator function, which is equal to 1 if the rotation from $\boldsymbol{\omega}_{i-1}$ to $\boldsymbol{\omega}_i$ turns right, and –1 if turning left. Note that its value would be 0 without rotation.

In this way, given an input outline and multiple graph-outline pairs, we first capture their polygonal features via the turning function. By calculating their distance using the L1-norm, we assess the similarity between the input outline and the outline in each graph-outline pair. We assess the similarity between the input outline and the outline in each graph-outline pair by calculating the distance of their polygonal features using the L1-norm as follows:

$$\mathcal{S} = ||\mathcal{A}_{in} - \mathcal{A}_{out}||_1 \tag{3}$$

where $\mathcal{A}_{in}$ denotes the polygonal features of the input outline and $\mathcal{A}_{out}$ denotes the polygonal features of the outline in each graph-outline pair. Then, we obtain a set of ranking scores $\mathcal{S}$ (i.e., similarities), which can be used to sort the graph-outline pairs. Based on the ranking scores, we select the graphs with top-k scores to generate the layouts. Lastly, following Graph2Plan[5], we align the graphs with the outline.

Graph alignment with outline. Given the selected top-k graph-outline pairs based on the ranking scores, we seek to align the selected graphs with the given input outline. To this end, following Graph2Plan[5], we align the graphs in several steps (in Fig. 7(b)). In step (a), we rotate the selected graphs according to the relative position between the gate of the outline in selected graph-outline pairs and the front door of the given outline. Note that we only consider 90, 180, 270, and 360 degrees to rotate the graphs, which, keeps the graphs still satisfying the realism of house layouts. In step (b), we move the position of graph nodes (i.e., moving nodes) that are outside of the outline into the given outline after rotation. Then, according to the original adjacent information of the nodes in the graph, we adjust each adjacent graph node of the moving node to keep the relative position between the two nodes.

4 Overall scheme of auto-3D-house design

The overall scheme of the proposed A3HD is shown in Fig. 8. Our A3HD can be divided into three parts: user requirements structuring and understanding, 2D layout generation, and 3D house rendering. Specifically, we take a list of user requirements as inputs, which are divided into three parts: layout, outline, and style. To generate various floor plans, we first select some bubble diagrams (i.e., graphs) from off-the-shelf house datasets based on the layout and outline requirements. Then, we construct a graph feature generation module (GFGM) to process the selected graphs. To effectively exploit the outline information, we represent the outline as a sequential outline. Moreover, we propose an outline feature generation module (OFGM) to extract the outline features.

Based on the graph features and outline features, we generate a set of bounding boxes (i.e., rooms) and produce 2D layouts after a box-outline alignment method.

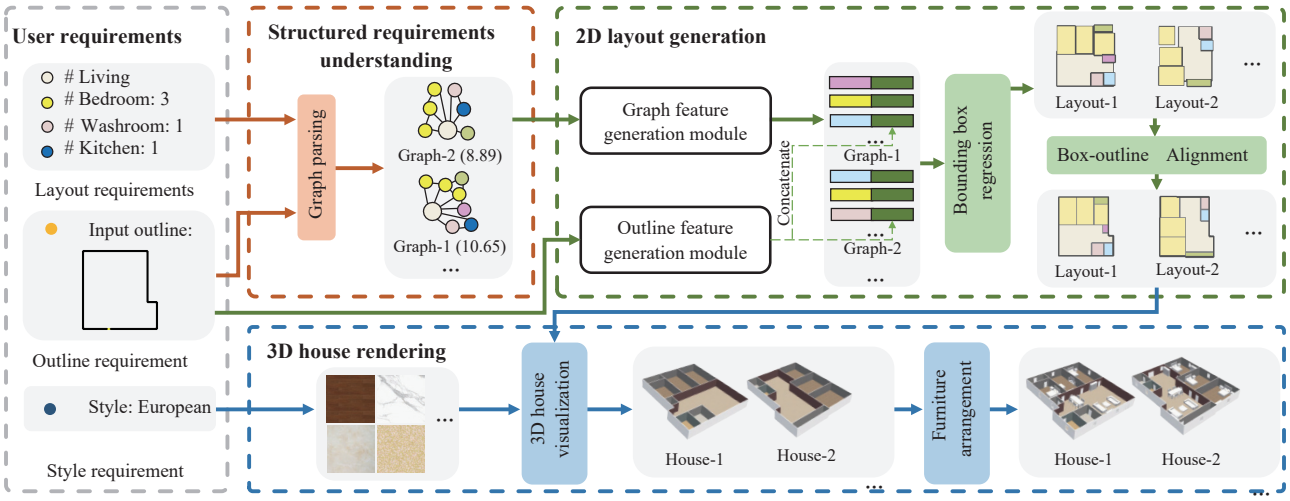


Fig. 8 Overview of A3HD. We divide the requirements into three parts: layout, outline, and style. Based on the requirements of layout and outline, we use a graph parsing method, which outputs a series of graphs that satisfy the requirements. Then, we design a 2D layout generation method, mainly consisting of a graph feature generation module (GFGM) and an outline feature generation module (OFGM), which captures the graph features and outline features, respectively. With these features, we generate the 2D layouts via a bounding box regression, followed by a box-outline alignment process. Last, we obtain 3D houses using a 3D house rendering method. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

To convert 2D layouts into 3D houses, we obtain a decoration scheme (i.e., texture images) based on the style requirements and then generate 3D houses via a rule-based decorating method for the users to choose from.

4.1 Graph feature generation module

For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ is a set of nodes representing rooms of the layout, and $\mathcal{E}$ is a set of edges denoting the relative position[1] between two rooms. Specifically, each edge $\boldsymbol{e}_{ij} \in \mathcal{E}$ can be defined as a triplet $\boldsymbol{e}_{ij} = (\boldsymbol{v}_i, \boldsymbol{r}_{ij}, \boldsymbol{v}_j)$, where $\boldsymbol{r}_{ij} \in \mathcal{R}$ denotes a relative position from $\boldsymbol{v}_i$ to $\boldsymbol{v}_j$. Here, $\boldsymbol{v}_i \in \mathcal{V}$ and $\boldsymbol{v}_j \in \mathcal{V}$ represent the i-th and the j-th rooms, respectively. Each node in $\mathcal{V}$ is assigned a set of attributes to store room information[2].

Inspired by [8], we process and update the nodes and their edges in the graph $\mathcal{G}$ with three functions f_s, f_r and f_e, where f_s and f_e are used to update the node features while f_r is used to update the edge features. These functions take the triple of vectors $(\boldsymbol{v}_i, \boldsymbol{r}_{ij}, \boldsymbol{v}_j)$ as input, and output new vectors $\widetilde{\boldsymbol{v}}_i$ and $\widetilde{\boldsymbol{r}}_{ij}$ for the nodes and edges, respectively. For the vector $\boldsymbol{r}_{ij}$, we can simply obtain the output vector $\widetilde{\boldsymbol{r}}_{ij} = f_r(\boldsymbol{e}_{ij}) = f_r(\boldsymbol{v}_i, \boldsymbol{r}_{ij}, \boldsymbol{v}_j)$. However, updating the node vector is more complex since a node (room) may have multiple edges (i.e., adjacent rooms). Hence, for each edge starting at $\boldsymbol{v}_i$, we use f_s to calculate a candidate vector and collect all these candidates in a set $\mathcal{V}_i^s$. Formally,

$$\mathcal{V}_i^s = \{f_s(\boldsymbol{e}_{ij}) \mid \boldsymbol{e}_{ij} = (\boldsymbol{v}_i, \widetilde{\boldsymbol{r}}_{ij}, \boldsymbol{v}_j) \in \mathcal{E}\}. \tag{4}$$

Similarly, we use f_e to compute a set of candidate vectors $\mathcal{V}_i^e$ for all the edges ending at $\boldsymbol{v}_i$. Mathematically,

$$\mathcal{V}_i^e = \{f_e(\boldsymbol{e}_{ji}) \mid \boldsymbol{e}_{ji} = (\boldsymbol{v}_j, \widetilde{\boldsymbol{r}}_{ji}, \boldsymbol{v}_i) \in \mathcal{E}\}. \tag{5}$$

We finally obtain an updated node vector $\widetilde{\boldsymbol{v}}_i = g(\mathcal{V}_i^s \cup \mathcal{V}_i^e)$, where $g(\cdot)$ denotes an elementwise average pooling function. For simplicity, we denote $\widetilde{\mathcal{V}}$ as a set of the updated vectors of rooms, where $\widetilde{\boldsymbol{v}}_i \in \widetilde{\mathcal{V}}$. For the functions f_s, f_r, and f_e, we use a network that concatenates three input vectors and feeds them to a multilayer perceptron (MLP).

4.2 Outline feature generation module

Based on the sequential outline $\mathcal{P}$, exploiting the outline information could be treated as extracting the order and location information of $\mathcal{P}$. Thus, to capture both location and order information from $\mathcal{P}$, we propose an outline feature generation module (OFGM). Specifically, we use a bidirectional LSTM (BiLSTM)[40] to extract the point features. Then, in order to extract the location information of the outline, we use an average pooling function $f_p(\cdot)$ followed by a multi-layer perceptron to fuse different point features. Formally, we capture the feature from the outline by using

$$\boldsymbol{x} = \mathrm{MLP}(f_p(\mathrm{BiLSTM}(\mathcal{P}))) \tag{6}$$

where $\boldsymbol{x} \in \mathbf{R}^D$ denotes the outline feature. Here, D is the dimension of the outline feature. Note that our sequential outline includes only a set of vertices which can reduce the computational cost mostly compared with outline images.

4.3 Process for 2D layout generation

For 2D house layout generation, based on the graph features and outline features, we obtain a set of bounding boxes which represent the rooms in the house layout. To satisfy the outline constraint, we leverage a box-outline alignment method to align the bounding boxes with the given outline to generate the final 2D house layout.

Bounding box regression. To generate a layout, we have to transform the 2D layout features from latent space to the image domain. To this end, we define each room as a coarse 2D bounding box, which can be defined as $\boldsymbol{o}_i = (x_0, y_0, x_1, y_1)$. In this way, we cast it as a problem of bounding box generation from given features. Specifically, based on the node (room) features $\widetilde{\mathcal{V}}$ (in Section 4.1) and outline feature $\boldsymbol{x}$ (in Section 4.2), we seek to generate the bounding box for each room. Thus, we use a two-layer perceptron network $h(\cdot)$ as a layout generator and predict the corresponding bounding box of each room $\hat{\boldsymbol{o}}_i = (\hat{x}_0, \hat{y}_0, \hat{x}_1, \hat{y}_1) = h([\boldsymbol{x}; \widetilde{\boldsymbol{v}}_i])$, where $\widetilde{\boldsymbol{v}}_i \in \widetilde{\mathcal{V}}$ and $[\cdot;\cdot]$ denotes the concatenation operation. Then, we integrate all the predicted boxes and obtain the target layout. For training our regression model, we minimize the objective function

$$\mathcal{L}_O = \frac{1}{M} \sum_{i=1}^{M} ||\hat{\boldsymbol{o}}_i - \boldsymbol{o}_i||_2^2 \tag{7}$$

where $\boldsymbol{o}_i$ denotes the ground-truth box of the i-th room and M is the number of the rooms in a layout.

Box-outline alignment. After obtaining the predicted bounding boxes layout, we align them to avoid overlap between boxes and make outline alignment to satisfy the outline constraint. Specifically, based on the post-processing method in HPGM[3], we align the bounding boxes of rooms with each other to obtain a refined layout. As depicted in Fig. 9, this process involves five steps. Step (a) involves extracting the boundary lines from all generated bounding boxes. In step (b), we merge adjacent line segments. Step (c) involves aligning these segments to form closed polygons. During step (d), we determine the affiliation of each polygon using a specific

[1] Relative position includes south, west, east, north, north-west, south-west, south-east, and north-east.

[2] Room information includes room type, room location, and relative size.

weight function:

$$W_{ij} = \iint \frac{1}{w_i h_i} \exp\left(-\left(\frac{x_j - c_{x_i}}{w_i}\right)^2 - \left(\frac{x_j - c_{x_i}}{w_i}\right)^2\right) \mathrm{d}x_j \mathrm{d}y_j \tag{8}$$

where $i = 1, 2, \cdots, n$ denotes the i-th predicted bounding box while $j = 1, 2, \cdots, m$ is the j-th aligned polygon. W_{ij} is the weight of the j-th polygon in relation to the i-th box. c_{x_i} and c_{y_i} represent the central point. w_i and h_i are the half width and height of the i-th bounding box. Coordinates within the aligned polygon are denoted by x_j and y_j. We assign the room type to the j-th polygon based on the corresponding bounding predicted box that has the maximum weight W.

Then, to make sure that all the refined rooms do not exceed the given outline (i.e., outline constraint), we calculate an intersection between rooms and the outline. We crop the parts that are outside of the outline. After removing the outline, we finally obtain a layout, where all the rooms satisfy the outline constraint.

4.4 3D house rendering from 2D layout

For better visualization of the generated floor plan, based on the style requirements, we convert the generated 2D layout into a 3D house. In addition, we place the 3D furniture into the 3D house based on some specific rules and a global rule.

3D house visualization based on style requirement. To process the style requirement, we use a series of decoration schemes which are designed by professional architects to visualize the 3D houses (as shown in Fig. 10). Based on the generated 2D layout and texture style (i.e., decoration scheme), we use a 3D visualization tool, called PyVista[41], to produce a 3D house model. In this way, we are able to provide different texture schemes for users and allow them to choose the most satisfactory scheme. Note that to draw the floor texture, we crop the floor into a series of small polygonal pieces, which makes the rendered textures clearer.

Furniture arrangement via rule-based decorating. To simulate real-world effects, we place doors, windows, and furniture in the 3D house model using a rule-based decorating method. We first add the windows and doors to our 3D house and then put the furniture in each room of the 3D house.

We first introduce the 3D mapping method we used to map 3D doors, windows, and furniture into our 3D house. For convenience, we take an example of mapping a window to a 3D house. Specifically, we denote the bounding box of the window in the 3D house as $\mathcal{B} = \{(x_i, y_i, z_i, 1)\}_{i=1}^{8}$. Based on the box $\mathcal{B}$, we estimate a transformation matrix $\boldsymbol{T} \in \mathbf{R}^{3\times4}$ by the RANSAC algorithm[42]. This matrix is able to project a 3D window $\hat{\mathcal{Q}} = \{(\hat{x}_i, \hat{y}_i, \hat{z}_i, 1)\}_{i=1}^{K}$ into the corresponding coordinates in our 3D house, where K is the number of points in the given 3D window. For the i-th point $\hat{\boldsymbol{q}}_i \in \hat{\mathcal{Q}}$, the process can be formulated as

$$\boldsymbol{q}_i = \boldsymbol{T} \times \hat{\boldsymbol{q}}_i \tag{9}$$

where $\boldsymbol{q}_i$ is the i-th point of the window in our 3D house.

To put the doors, windows, and furniture into our 3D house, we should determine the 3D position $\mathcal{B}$ in each of our 3D houses. To this end, we design some specific rules and a global rule. Taking a bed as an example, we should position the bed against a wall but without blocking the door. Moreover, to avoid overlaps among the furniture, we design a global rule that computes the IoU between

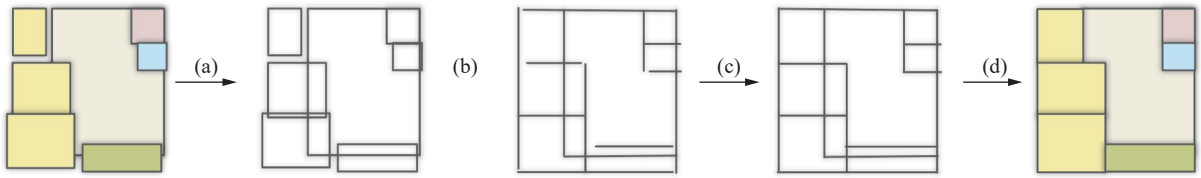


Fig. 9 The procedures of our post-processing method. We follow the post-processing method in HPGM[3] to align the predicted bounding boxes layout. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

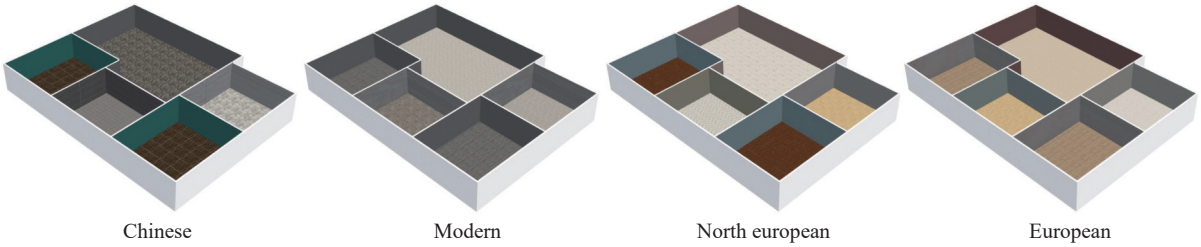


Fig. 10 Examples of different decoration schemes of 3D houses (Colored figures are available in the online version at https://link.springer.com/journal/11633)

different pieces of furniture. We depict the specific placing rule of windows, doors, and furniture as follows.

1) Window: For each room, we select the walls which are between the room and the given outline. Then, we set the window on the longest wall of the selected walls.

2) Door: We assume that the entrance to the house is through the living room. For the entrance of the house, we set it to the same position on one exterior wall of the given outline. For other doors, we set them on the walls between the living room and other rooms.

3) Bed: We position the bed against the middle of the longest wall in each room without blocking the door.

4) Nightstand: We choose the wall which the bed is against. If the length of the wall is enough, we add the nightstands on one/both sides of the bed.

5) Sofa: We process the sofa as a bounding box to judge the available space in the living room which avoids overlaps between the sofa and the wall. Then, we place the sofa in the middle of the available space.

6) TV: In the living room, we set the TV facing the middle of the sofa. As for the bedroom, we put the TV facing the middle of the bed.

7) Wardrobe: For the wardrobe, we put it on the corner between two walls in the bedroom without blocking the door, window, and other furniture.

After obtaining position $\mathcal{B}$ of the furniture, we use (9) to map the furniture into a 3D house. We find that these rules work well in most cases and are good enough to set reasonable positions, but a learning-based method may improve this process, and we leave it for future work.

5 Experiments

5.1 Datasets and evaluation metrics

Text-to-3D house model (T3HM) dataset[3]. This dataset contains 2 000 houses, 13 478 rooms, and 873 texture images. In addition, it provides the corresponding natural language descriptions of each floor plan (i.e., layout). For the T3HM dataset, following the settings in [3], we use 1 600 pairs (houses) for training and 400 for testing in the experiments.

RPLAN dataset[6]. This is a large-scale dataset with more than 80 000 floor plans (i.e., layouts), derived from real-world residential buildings in the Asian estate market. In the experiments based on the RPLAN dataset, we follow the settings in [5], which takes 56 000 floor plan data for training, 12 000 for validation, and 12 000 for testing.

Evaluation metrics. To evaluate the effectiveness of the layout generation method, following [3, 5], we use intersection over union (IoU) to assess the overlap between the predicted box and ground-truth box, where the value ranges from 0 to 1. For evaluation of the alignment between the generated layout and the input outline, we design a new metric, called bounding intersection over union (BIoU). The BIoU measures the overlap between the entire contour of the generated layout and the given outline. The value of BIoU also ranges from 0 to 1. The higher evaluation value of IoU and BIoU refers to better performance. Besides, following [1, 2], we use fréchet inception distance (FID)[43] to investigate the diversity of our A3HD based on the rasterized layout images. To evaluate the alignment between the generated layouts and user requirements, we use a graph editing distance[1, 2, 44] to measure the difference between the graphs retrieved from the generated layouts and input graphs.

Implementation details. When measuring the similarity of two outlines, we sample 1 000 points from each outline at equal intervals. Then, we use the turning function to transform each point into a feature and concatenate them together to obtain the polygonal features. In the experiment of 2D layout generation, our implementation uses PyTorch[45]. We do not use any data augmentation techniques to train models. A BiLSTM[40] with 256 hidden units is used in our outline feature generation module. We apply batch normalization and a ReLU activation after each linear layer. During the training of our 2D layout feature generation module, we set batch size as 20. In addition, we use Adam[46] with an initial learning rate of 1×10^{-4} and a weight decay of 5×10^{-4} as the optimizer. We train the 2D layout generation model for 100 epochs in total.

5.2 Quantitative evaluation

Structured requirements understanding. We use the graph editing distance to measure the similarity between graphs reconstructed from the house datasets and corresponding ground-truth graphs. The more similar the two graphs are, the selected graphs are more suitable for layout generation. Specifically, we randomly sample 5 000 structured requirements and use our graph parsing method to reconstruct different graphs without graph alignment. Then, to determine the best measurement method, we use l_1, l_2, l_∞ normalization methods to compare different outline features. As shown in Table 2, the measure way of l_1-norm obtains the best performance. Moreover, we compare the top-5 graphs and the top 5–10 graphs reconstructed from the house datasets. The top-5 experimental results outperform the top 5–10 results which demonstrates the effectiveness of our graph sorting method (in Section 3.3).

Layout generation. We compare our 2D layout generation method with two baselines, i.e., HPGM[3] and Graph2plan[5], on the T3HM and RPLAN datasets. For a fair comparison, we compare the generated bounding box layout without using the box-outline alignment method (in Section 4.3). Although HPGM takes the linguistic requirements as input, they parse the input into graphs[3]

[3] We change the room information of the node in the graph from the absolute location to the relative position.

Table 2 Similarity between selected different ranking graphs and ground-truth graphs. We compare the results of using different distance measurement methods to match the outlines.

Ranking graphs	l_1-norm	l_2-norm	l_∞-norm
Top-5 ↓	**2.26**	2.28	2.31
Top 5-10 ↓	3.01	3.05	3.03

that are similar to ours. Since HPGM does not make use of the outline, we use OFGM to extract the outline feature and combine it with the feature generated by HPGM to predict the bounding box layouts (i.e., HPGM + OFGM). As shown in Table 3, our method achieves the best performance on both the T3HM and RPLAN datasets. Specifically, the highest IoU and BIoU value demonstrate that our method is able to locate each room more precisely than other methods and is more satisfied with the outline constraint. Moreover, "HPGM+OFGM" brings a huge boost to the experimental result on BIoU compared with HPGM, which demonstrates the importance of the outline and the effectiveness of our OFGM.

Table 3 IoU and BIoU results of the generated layouts from our method and baselines. "HPGM+OFGM" means combining HPGM with our OFGM to generate house layout.

Methods	RPLAN[6]		T3HM[6]	
	IoU ↑	BIoU ↑	IoU ↑	BIoU ↑
HPGM[3]	0.514	0.756	0.589	0.484
HPGM + OFGM	0.609	0.832	0.591	0.815
Graph2plan[5]	0.660	0.790	0.627	0.810
A3HD (ours)	**0.731**	**0.902**	**0.653**	**0.852**

Human study. Since automated house design is a user-oriented task, the automatic metrics can not fully evaluate the performance of our method. Thus, inspired by [1, 47–49], we conduct a human study to compare our method with baseline methods. Since the baseline methods do not generate 3D houses, for a fair comparison, we apply our box-outline alignment method and 3D house rendering method on 2D bounding box layouts generated by baselines to generate 3D houses. Then, we invited 30 human evaluators (university students) to score the 3D houses. Specifically, we randomly sample 50 graphs and input outlines from datasets to generate bounding box layouts based on A3HD and baseline methods. Given the same user requirements, we show generated 3D houses besides ground-truth 3D houses corresponding to the user requirement for evaluation (see Fig. 11 for an example). We set the evaluation score range from 0 to 10 (a higher score means a better 3D house). As shown in Table 4, apart from the ground-truth 3D houses, our method achieves the best user scores compared with baseline methods. The results demonstrate that the 3D houses generated by A3HD are more reasonable than those generated by the baseline methods.

5.3 Qualitative evaluation

Structured requirements understanding. For the qualitative evaluation of the selected graphs, we visualize the selected graphs and corresponding ground-truth graphs in Fig. 6. The experimental results show that our selected graphs satisfy the layout requirements as ground-truth graphs. Moreover, the selected graphs are all effective for layout generation since they are all designed by architects. For example, the living room node has the maximum number of connections among all room nodes.

Layout generation. As shown in Fig. 12, we compare the layouts generated by our method and those generated by baselines with the same user requirements. We use the red circle remark to highlight the abnormal place in the generated house layout. The house layout generated by HPGM could not satisfy the outline constraint. As for the floor plan generated by Graph2plan, it may produce unreasonable rooms or miss some rooms. Our generated layouts are more natural than those of baseline methods (i.e., HPGM and Graph2plan). Moreover, our method is able to produce competitive visual results, even when compared with human-made layouts (i.e., GT in Fig. 12).

3D house rendering. To evaluate the effectiveness of our 3D house rendering method, we apply the 3D house rendering method on 2D layouts generated by our A3HD and the baseline methods, as well as the ground-truth counterparts (in Fig. 12). With the 2D layouts generated in the experiment of layout generation, we provide the 3D visualization of them. As shown in Fig. 13, compared with 2D layout visualization, it will be easier to appreciate different house layouts generated by different methods in 3D visualization. Moreover, the unreasonably designed room in the generated floor plan could be more

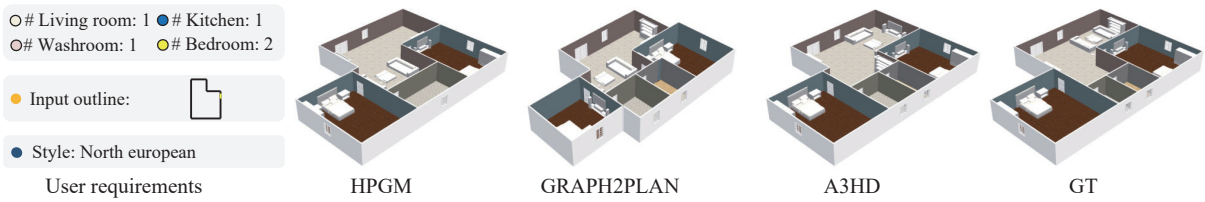


Fig. 11 Example of the houses generated by our A3HD and baseline methods besides the ground-truth (GT) 3D houses (Colored figures are available in the online version at https://link.springer.com/journal/11633)

Table 4 Results of human study based on RPLAN dataset

Methods	Humans	A3HD (ours)	Graph2plan	HPGM
Scores ↑	7.14	6.14	5.78	5.38

easily discovered. From the experimental result, the 3D houses generated by our A3HD achieve competitive visual quality even compared with the ground-truth houses.

5.4 Ablation study

Effectiveness of GFGM. To evaluate the effectiveness of our GFGM, we replace the GFGM with an LSTM model which takes the graph nodes as sequential data and uses the data order as the input order to extract the graph features. For a fair comparison, we remove the OFGM module to eliminate the influence of the outline. Table 5 shows that our GFGM is able to achieve higher values of both IoU and BIoU, which demonstrates the superiority of the GFGM.

Effectiveness of OFGM. To evaluate the impact of the proposed OFGM, we conduct two variants of our method. The first variant removes the OFGM directly (i.e., GFGM only) while the other replaces the OFGM with an MLP network (i.e., GFGM + MLP) which directly takes the outline points as input and generates outline features. As shown in Table 6, our A3HD (i.e., GFGM + OFGM) achieves the best performance, which demonstrates the effectiveness of our OFGM.

Effectiveness of sequential outline. To evaluate the effectiveness of our novel outline representation method, we compare the layouts generated from the sequential outline and outline images. We represent the outline as images and use CNNs to extract the outline features (i.e., Image&CNN). Additionally, we use GFGM to extract the graph features. To measure the computational cost of these two methods, we employ the metric multiply-add operations (MAdds). Each MAdds operation is a common step that computes the product of two numbers and adds that product to an accumulator. The results in Table 1 demonstrate the effectiveness of our sequential outline. With similar parameters, processing the outline as sequential outlines is less computationally cost than treating it as outline images.

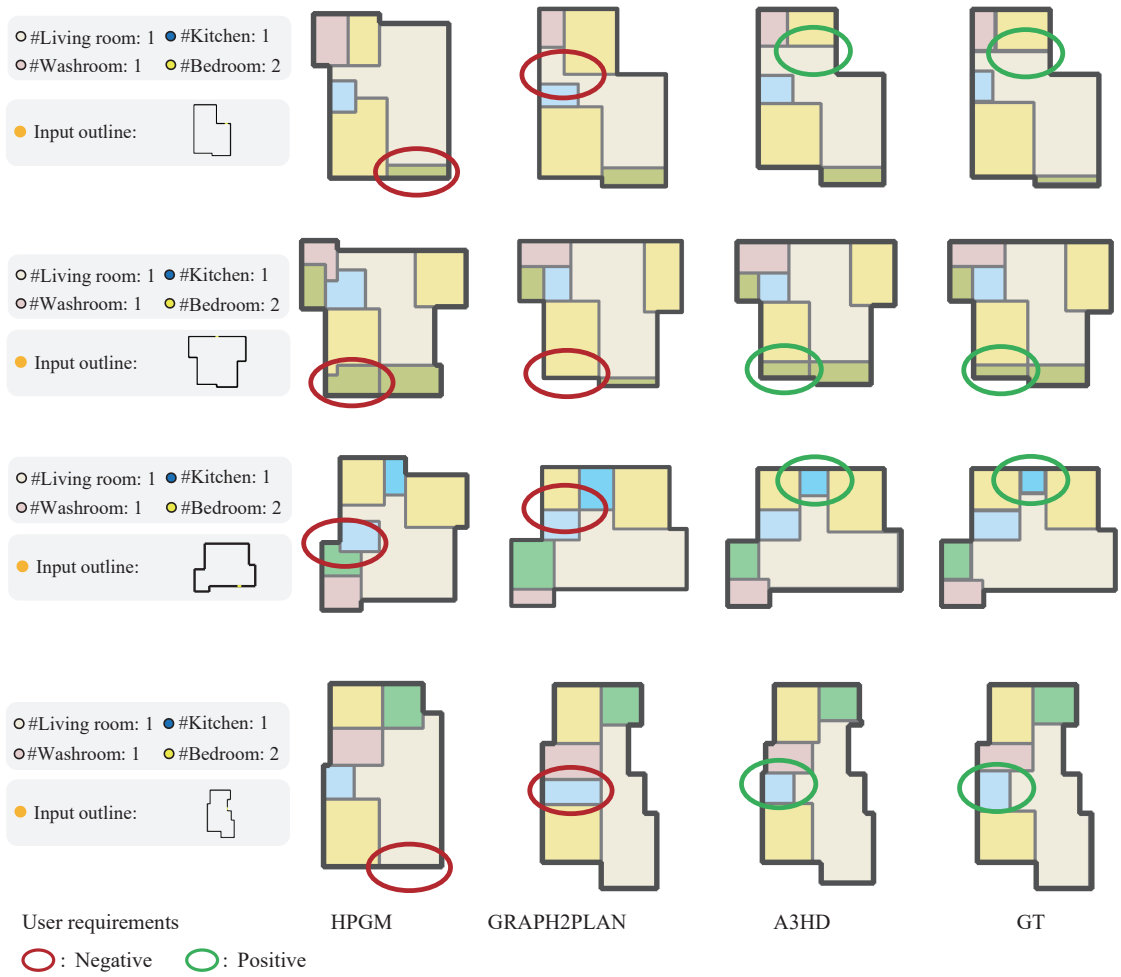


Fig. 12 Visual results of 2D layouts generated by our A3HD and baselines on RPLAN dataset based on the same input requirement (Colored figures are available in the online version at https://link.springer.com/journal/11633)

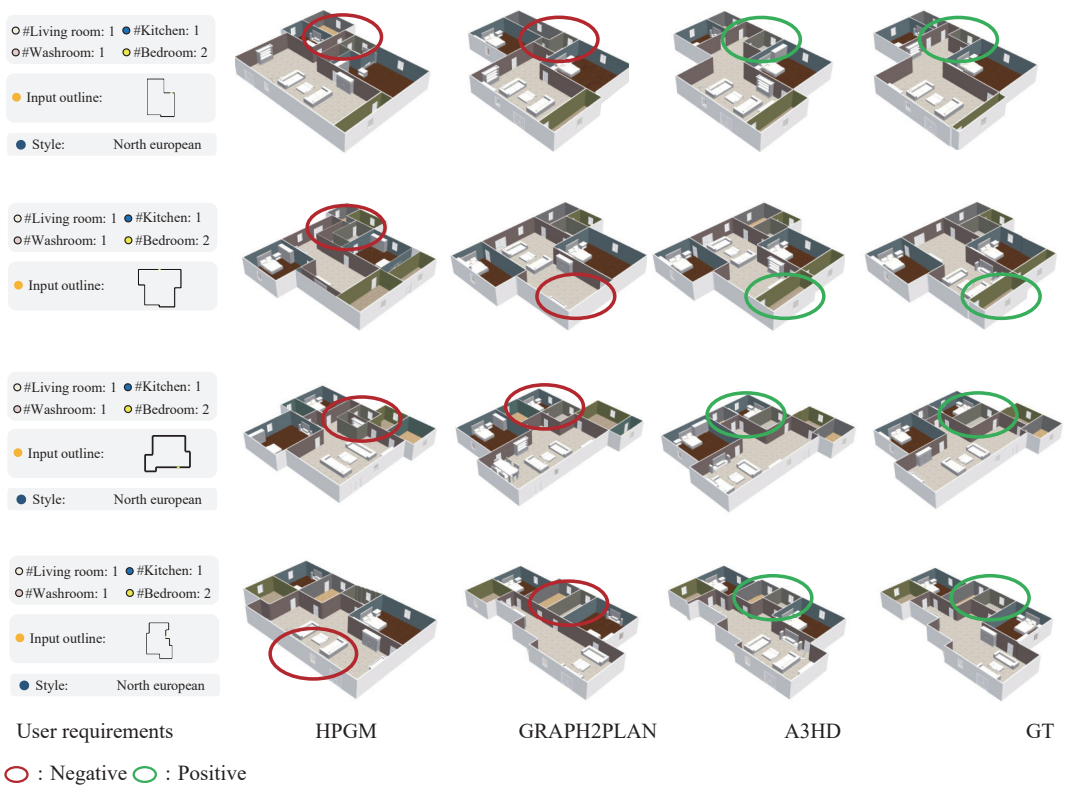


Fig. 13 Visual results of 3D houses generated by our A3HD and baselines on RPLAN dataset based on the same requirement (Colored figures are available in the online version at https://link.springer.com/journal/11633)

Table 5 Effectiveness of GFGM. We remove OFGM and use the graph data based only on the RPLAN dataset and T3HM dataset.

Methods	RPLAN[6]		T3HM[3]	
	IoU↑	BIoU↑	IoU↑	BIoU↑
LSTM	0.584	0.811	0.564	0.795
GFGM	**0.706**	**0.887**	**0.586**	**0.807**

Table 6 Effectiveness of OFGM. We compare our method with two variants, which remove OFGM (i.e., GFGM only) and replace OFGM with MLP (i.e., GFGM + MLP).

Methods	RPLAN[6]		T3HM[3]	
	IoU↑	BIoU↑	IoU↑	BIoU↑
GFGM only	0.706	0.887	0.586	0.807
GFGM + MLP	0.710	0.889	0.613	0.818
GFGM + OFGM	**0.731**	**0.902**	**0.653**	**0.852**

5.5 More discussions

Analysis of convergence. To verify the convergence of our method, we optimize A3HD, HPGM, and Graph2plan in 100 epochs on the RPLAN dataset. We record the loss and IoU in the validation set and draw the corresponding convergence curves. As shown in Fig. 14, our method can achieve better performance during training (i.e., lower loss and higher IoU).

Diverse results of our A3HD. As shown in Fig. 15, based on the same structured requirements, our A3HD is able to generate various 3D houses. For quantitative evaluation of diversity, following the setting of [1, 2], we randomly sample 5 000 user requirements and generate 10 house layout variations for each requirement set based on the A3HD and baseline methods. As for HPGM[3], we apply our user requirements understanding module to generate various house layouts. We transform the generated house layouts into layout images. Then, we use FID scores[43] to compute the distance between the distribution of generated layouts and ground-truth layouts corresponding to the user requirements. A lower FID score means a better diversity of generative methods. In Table 7, our A3HD achieves the best diversity results compared with the baseline methods.

Alignment evaluation of our A3HD. To measure the alignment between the generated layout and user requirements, we conduct an alignment evaluation. Following [1, 2], we randomly sample 5 000 user requirements and generate bounding box layouts based on our A3HD and baselines. Then, we retrieve graphs from the generated bounding box layouts. In this way, to measure the requirement alignment of A3HD is to measure the difference between the retrieved graphs and input graphs. As shown in Table 7, the performance of our A3HD is better than that of the baseline methods.

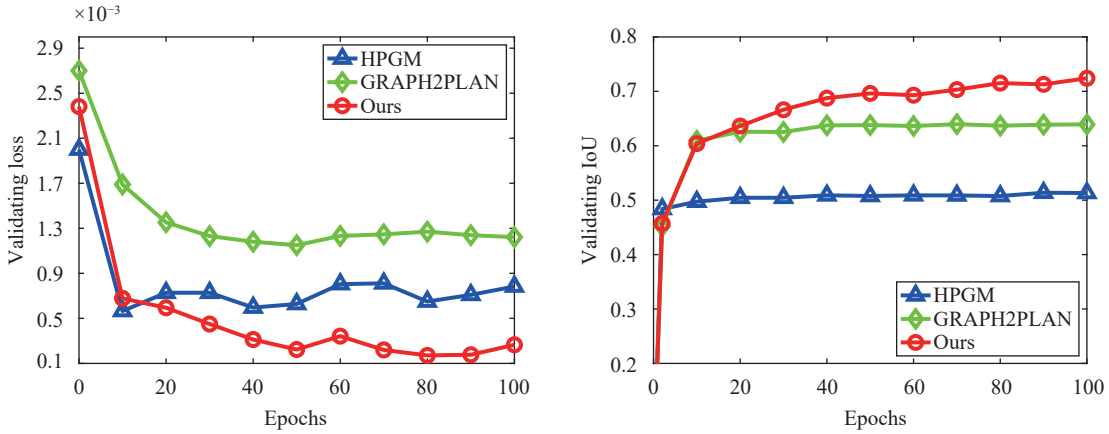


Fig. 14 Validating loss and IoU for A3HD and baselines (Colored figures are available in the online version at https://link.springer.com/journal/11633)

Fig. 15 Diverse 3D houses generated by the proposed A3HD based on the same user requirements (Colored figures are available in the online version at https://link.springer.com/journal/11633)

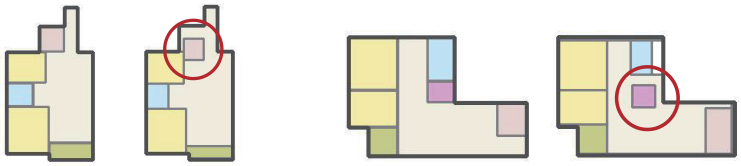


Fig. 16 Failure cases of our A3HD. Some unreasonable designed rooms (marked by red circles) of generated floor plans are not connected with the outline. (Colored figures are available in the online version at https://link.springer.com/journal/11633)

Table 7 Diversity and alignment evaluation of our A3HD compared with baselines based on the RPLAN dataset

Methods	Diversity↓	Alignment↓
HPGM	22.22	1.28
GRAPH2PLAN	17.78	0.61
A3HD (ours)	**13.19**	**0.35**

5.6 Limitations and future works

In this section, we discuss the limitations of our A3HD method and outline directions for future research. One primary limitation is our approach′s difficulty in accurately handling tiny, internal rooms. For instance, as shown in Fig. 16, some small rooms are positioned unreasonably, such as a room inside the house not aligning with other rooms or exterior walls, resulting in impractical white spaces. This design is generally unfavorable in real-world scenarios. Our analysis suggests that this issue stems from our box-outline alignment method, which lacks constraints to ensure that each room aligns with at least one other room or an exterior wall. To overcome this, we plan to develop a more sophisticated box-outline alignment method (e.g., incorporating a layout discriminator) in our future work.

Another limitation is that our A3HD is incapable of dealing with linguistic descriptions while people sometimes prefer generating layouts of their dream house from one sentence. Although we have conducted experiments using a stanford scene graph parser[50] to parse the linguistic descriptions into a structural graph layout, we find that the model may be confused about the ambiguous descriptions. In the future, we seek to address this challenge by harnessing the power of the emerging large language models (LLM).

6 Conclusions

In this paper, we propose an auto-3D-house design (A3HD) method, which seeks to generate various 3D houses automatically from structured user requirements. To focus on the useful information of user requirements, we simplify them as structured requirements which are divided into three parts: layout, outline, and style. We provide the 3D house rendering from the floor plan for users to better understand the 2D layout. Moreover, to improve the performance of the automated house design method, we explore a new outline representation method to make use of the outline more effectively and propose an outline feature generation module to extract the outline feature. The experimental results on RPLAN and T3HM datasets indicate that our method outperforms others in terms of structure requirements understanding and layout generation.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (NSFC) (No. 62072190), and TCL Science and Technology Innovation Fund, China.

Declarations of conflict of interest

The authors declared that they have no conflicts of interest to this work.

References

[1] 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 *Proceedings of the 16th European Conference on Computer Vision*, Glasgow, UK, pp. 162–177, 2020. DOI: 10.1007/978-3-030-58452-8_10.

[2] N. Nauata, S. Hosseini, K. H. Chang, H. Chu, C. Y. Cheng, Y. Furukawa. House-GAN++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, USA, pp. 13627–13636, 2021. DOI: 10.1109/CVPR46437.2021.01342.

[3] Q. Chen, Q. Wu, R. Tang, Y. H. Wang, S. Wang, M. K. Tan. Intelligent home 3D: Automatic 3D-house design from linguistic descriptions only. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, USA, pp. 12622–12631, 2020. DOI: 10.1109/CVPR42600.2020.01264.

[4] W. Para, P. Guerrero, T. Kelly, L. Guibas, P. Wonka. Generative layout modeling using constraint graphs. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, Montreal, Canada, pp. 6670–6680, 2021. DOI: 10.1109/ICCV48922.2021.00662.

[5] R. Z. Hu, Z. Y. Huang, Y. H. Tang, O. van Kaick, H. Zhang, H. Huang. Graph2Plan: Learning floorplan generation from layout graphs. *ACM Transactions on Graphics*,

vol. 39, no. 4, Article number 118, 2020. DOI: 10.1145/3386569.3392391.

[6] W. M. Wu, X. M. Fu, R. Tang, Y. H. Wang, Y. H. Qi, L. G. Liu. Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics*, vol. 38, no. 6, Article number 234, 2019. DOI: 10.1145/3355089.3356556.

[7] H. Tang, Z. Y. Zhang, H. Shi, B. Li, L. Shao, N. Sebe, R. Timofte, L. Van Gool. Graph transformer GANs for graph-constrained house generation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Vancouver, Canada, pp. 2173–2182, 2023. DOI: 10.1109/CVPR52729.2023.00216.

[8] O. Ashual, L. Wolf. Specifying object attributes and relations in interactive scene generation. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, Seoul, Republic of Korea, pp. 4560–4568, 2019. DOI: 10.1109/ICCV.2019.00466.

[9] H. Dhamo, A. Farshad, I. Laina, N. Navab, G. D. Hager, F. Tombari, C. Rupprecht. Semantic image manipulation using scene graphs. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, USA, pp. 5212–5221, 2020. DOI: 10.1109/CVPR42600.2020.00526.

[10] A. A. Jyothi, T. Durand, J. W. He, L. Sigal, G. Mori. LayoutVAE: Stochastic scene layout generation from a label set. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, Seoul, Republic of Korea, pp. 9894–9903, 2019. DOI: 10.1109/ICCV.2019.00999.

[11] J. N. Li, J. M. Yang, A. Hertzmann, J. M. Zhang, T. F. Xu. LayoutGAN: Generating graphic layouts with wireframe discriminators. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, USA, 2019.

[12] A. Luo, Z. T. Zhang, J. J. Wu, J. B. Tenenbaum. End-to-end optimization of scene layout. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, USA, pp. 3753–3762, 2020. DOI: 10.1109/CVPR42600.2020.00381.

[13] X. Yang, Z. L. Ma, L. T. Yu, Y. Cao, B. C. Yin, X. P. Wei, Q. Zhang, R. W. H. Lau. Automatic comic generation with stylistic multi-page layouts and emotion-driven text balloon generation. *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 17, no. 2, Article number 55, 2021. DOI: 10.1145/3440053.

[14] X. Y. Yang, T. Mei, Y. Q. Xu, Y. Rui, S. P. Li. Automatic generation of visual-textual presentation layout. *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 12, no. 2, Article number 33, 2016. DOI: 10.1145/2818709.

[15] C. Liu, J. J. Wu, P. Kohli, Y. Furukawa. Raster-to-vector: Revisiting floorplan transformation. In *Proceedings of IEEE International Conference on Computer Vision*, Venice, Italy, pp. 2214–2222, 2017. DOI: 10.1109/ICCV.2017.241.

[16] Z. L. Zeng, X. Z. Li, Y. K. Yu, C. W. Fu. Deep floor plan recognition using a multi-task network with room-boundary-guided attention. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, Seoul, Republic of Korea, pp. 9095–9103, 2019. DOI: 10.1109/ICCV.2019.00919.

[17] H. Zhao, M. Lu, A. B. Yao, Y. W. Guo, Y. R. Chen, L. Zhang. Physics inspired optimization on semantic transfer features: An alternative method for room layout estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, USA, pp. 870–878, 2017. DOI: 10.1109/CVPR.2017.99.

[18] X. L. Lv, S. C. Zhao, X. Y. Yu, B. Q. Zhao. Residential floor plan recognition and reconstruction. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, USA, pp. 16712–16721, 2021. DOI: 10.1109/CVPR46437.2021.01644.

[19] M. Y. Li, A. G. Patil, K. Xu, S. Chaudhuri, O. Khan, A. Shamir, C. H. Tu, B. Q. Chen, D. Cohen-Or, H. Zhang. GRAINS: Generative recursive autoencoders for INdoor scenes. *ACM Transactions on Graphics*, vol. 38, no. 2, Article number 12, 2019. DOI: 10.1145/3303766.

[20] D. Ritchie, K. Wang, Y. A. Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, USA, pp. 6175–6183, 2019. DOI: 10.1109/CVPR.2019.00634.

[21] K. Wang, Y. A. Lin, B. Weissmann, M. Savva, A. X. Chang, D. Ritchie. PlanIT: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics*, vol. 38, no. 4, Article number 132, 2019. DOI: 10.1145/3306346.3322941.

[22] K. Wang, M. Savva, A. X. Chang, D. Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics*, vol. 37, no. 4, Article number 70, 2018. DOI: 10.1145/3197517.3201362.

[23] T. Weiss, M. Nakada, D. Terzopoulos. Automated layout synthesis and visualization from images of interior or exterior spaces. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Honolulu, USA, pp. 41–47, 2017. DOI: 10.1109/CVPRW.2017.12.

[24] H. T. Yang, Z. W. Zhang, S. M. Yan, H. B. Huang, C. Y. Ma, Y. Zheng, C. Bajaj, Q. X. Huang. Scene synthesis via uncertainty-driven attribute synchronization. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, Montreal, Canada, pp. 5610–5620, 2021. DOI: 10.1109/ICCV48922.2021.00558.

[25] S. Y. Huang, S. Y. Qi, Y. X. Zhu, Y. X. Xiao, Y. L. Xu, S. C. Zhu. Holistic 3D scene parsing and reconstruction from a single RGB image. In *Proceedings of the 15th European Conference on Computer Vision*, Munich, Germany, pp. 194–211, 2018. DOI: 10.1007/978-3-030-01234-2_12.

[26] H. Izadinia, Q. Shan, S. M. Seitz. IM2CAD. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, USA, pp. 2422–2431, 2017. DOI: 10.1109/CVPR.2017.260.

[27] S. T. Yang, F. E. Wang, C. H. Peng, P. Wonka, M. Sun, H. K. Chu. DuLa-Net: A dual-projection network for estimating room layouts from a single RGB panorama. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, USA, pp. 3358–3367, 2019. DOI: 10.1109/CVPR.2019.00348.

[28] Y. D. Zhang, S. R. Song, P. Tan, J. X. Xiao. PanoContext: A whole-room 3D context model for panoramic scene understanding. In *Proceedings of the 13th European Conference on Computer Vision*, Zurich, Switzerland, pp. 668–686, 2014. DOI: 10.1007/978-3-319-10599-4_43.

[29] C. H. Zou, A. Colburn, Q. Shan, D. Hoiem. LayoutNet: Reconstructing the 3D room layout from a single RGB image. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, USA, pp. 2051–2059, 2018. DOI: 10.1109/CVPR.2018.00219.

[30] C. Sun, C. W. Hsiao, M. Sun, H. T. Chen. HorizonNet:

Learning room layout with 1D representation and pano stretch data augmentation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, USA, pp. 1047–1056, 2019. DOI: 10.1109/CVPR.2019.00114.

[31] C. G. Yan, B. Y. Shao, H. Zhao, R. X. Ning, Y. D. Zhang, F. Xu. 3D room layout estimation from a single RGB image. *IEEE Transactions on Multimedia*, vol. 22, no. 11, pp. 3014–3024, 2020. DOI: 10.1109/TMM.2020.2967645.

[32] P. Merrell, E. Schkufza, V. Koltun. Computer-generated residential building layouts. *ACM Transactions on Graphics*, vol. 29, no. 6, Article number 181, 2010. DOI: 10.1145/1882261.1866203.

[33] F. Bao, D. M. Yan, N. J. Mitra, P. Wonka. Generating and exploring good building layouts. *ACM Transactions on Graphics*, vol. 32, no. 4, Article number 122, 2013. DOI: 10.1145/2461912.2461977.

[34] S. Chaillou. ArchiGAN: Artificial intelligence x architecture. *Architectural Intelligence: Selected Papers from the 1st International Conference on Computational Design and Robotic Fabrication (CDRF 2019)*, P. F. Yuan, M. K. Xie, N. Leach, J. W. Yao, X. Wang, Eds., Singapore: Springer, pp. 117–127, 2020. DOI: 10.1007/978-981-15-6568-7_8.

[35] W. M. Wu, L. B. Fan, L. G. Liu, P. Wonka. MIQP-based layout design for building interiors. *Computer Graphics Forum*, vol. 37, no. 2, pp. 511–521, 2018. DOI: 10.1111/cgf.13380.

[36] Y. Zhao, J. J. Zhang, C. Q. Zong. Transformer: A general framework from machine translation to others. *Machine Intelligence Research*, vol. 20, no. 4, pp. 514–538, 2023. DOI: 10.1007/s11633-022-1393-5.

[37] X. Wang, G. Y. Chen, G. W. Qian, P. C. Gao, X. Y. Wei, Y. W. Wang, Y. H. Tian, W. Gao. Large-scale multi-modal pre-trained models: A comprehensive survey. *Machine Intelligence Research*, vol. 20, no. 4, pp. 447–482, 2023. DOI: 10.1007/s11633-022-1410-8.

[38] G. P. Ji, M. C. Zhuge, D. H. Gao, D. P. Fan, C. Sakaridis, L. V. Gool. Masked vision-language transformer in fashion. *Machine Intelligence Research*, vol. 20, no. 3, pp. 421–434, 2023. DOI: 10.1007/s11633-022-1394-4.

[39] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, J. S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 209–216, 1991. DOI: 10.1109/34.75509.

[40] I. Sutskever, O. Vinyals, Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, Montreal, Canada, pp. 3104–3112, 2014.

[41] C. B. Sullivan, A. A. Kaszynski. PyVista: 3D plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK). *Journal of Open Source Software*, vol. 4, no. 37, Article number 1450, 2019. DOI: 10.21105/joss.01450.

[42] M. A. Fischler, R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. DOI: 10.1145/358669.358692.

[43] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, S. Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, USA, pp. 6629–6640, 2017.

[44] Z. Abu-Aisheh, R. Raveaux, J. Y. Ramel, P. Martineau. An exact graph edit distance algorithm for solving pattern recognition problems. In *Proceedings of International Conference on Pattern Recognition Applications and Methods*, Lisbon, Portugal, pp. 271–278, 2015. DOI: 10.5220/0005209202710278.

[45] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. M. Lin, A. Desmaison, L. Antiga, A. Lerer. Automatic differentiation in PyTorch. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, Long Beach, USA, 2017.

[46] D. P. Kingma, J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA, 2014.

[47] S. G. Liu, H. X. Wang. Talking face generation via facial anatomy. *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 19, no. 3, Article number 125, 2023. DOI: 10.1145/3571746.

[48] A. Siarohin, G. Zen, C. Majtanovic, X. Alameda-Pineda, E. Ricci, N. Sebe. Increasing image memorability with neural style transfer. *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 15, no. 2, Article number 42, 2019. DOI: 10.1145/3311781.

[49] L. Zhang, C. J. Long, X. L. Zhang, C. X. Xiao. Exploiting residual and illumination with GANs for shadow detection and shadow removal. *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 19, no. 3, Article number 120, 2023. DOI: 10.1145/3571745.

[50] S. Schuster, R. Krishna, A. Chang, L. Fei-Fei, C. D. Manning. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proceedings of the 4th Workshop on Vision and Language*, Lisbon, Portugal, pp. 70–80, 2015. DOI: 10.18653/v1/W15-2812.

Mingkui Tan received the B. Sc. degree in environmental science and engineering and M. Sc. degree in control science and engineering, both from Hunan University, China in 2006 and 2009. He received the Ph.D. degree in computer science from Nanyang Technological University, Singapore in 2014. From 2014–2016, he worked as a senior research associate on computer vision at the School of Computer Science, University of Adelaide, Australia. He is currently a professor with the School of Software Engineering, South China University of Technology, China.

His research interests include machine learning, sparse analysis, deep learning, and large-scale optimization.

E-mail: mingkuitan@scut.edu.cn

ORCID iD: 0000-0001-8856-756X

Qi Chen received the B. Sc. degree in software engineering and the M. Sc. degree at the School of Software Engineering, South China University of Technology, China in 2017 and 2020. He is a Ph.D. degree candidate in the Faculty of Engineering, Computer and Mathematical Sciences (ECMS), the University of Adelaide, Australia.

His research interests include deep learning and computer vision.

E-mail: qi.chen04@adelaide.edu.au

Zixiong Huang received the B. Sc. degree in mechanical and electronic engineering from the School of Mechanical and Automotive Engineering, South China University of Technology, China in 2022. He is currently a master student in the School of Software Engineering, South China University of Technology, China.

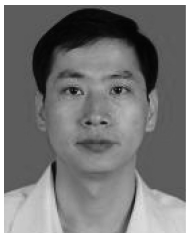
His research interests include neural radiance fields and virtual digital humans.

E-mail: sesmilhzx@mail.scut.edu.cn
ORCID iD: 0000-0001-7261-3509

Qi Wu received the M. Sc. degree and the Ph. D. degree in computer science from the University of Bath, UK in 2011 and 2015. He is a senior lecturer (assistant professor) at the University of Adelaide, Australia, and he is an associate investigator at the Australia Centre for Robotic Vision (ACRV), Australia. He is the ARC Discovery Early Career Researcher Award (DECRA) Fellow between 2019–2021. His educational background is primarily in computer science and mathematics. His work has been published in prestigious journals and conferences such as TPAMI, CVPR, ICCV, AAAI, and ECCV.

His research interests include vision and language problems, image captioning, visual question answering, visual dialog, etc.

E-mail: qi.wu01@adelaide.edu.au



Yuanqing Li received the Ph. D. degree in control theory and applications from South China University of Technology, China in 1997. He is the author or co-author of more than 60 scientific papers in journals and conference proceedings.

His research interests include blind signal processing, sparse representation, machine learning, brain–computer interface, EEG, and fMRI data analysis.

E-mail: auyqli@scut.edu.cn

Jiaqiu Zhou received the B. Sc. degree in mechanical and electronic engineering from the School of Mechanical and Automotive Engineering and the M. Sc. degree in software engineering from the School of Software Engineering, South China University of Technology, China in 2018 and 2022. He is currently a computer vision engineer in ShenZhen DJI Technology Company Limited, China.

His research interests include natural language processing and image processing.

E-mail: zhoujball@gmail.com (Corresponding author)
ORCID iD: 0009-0004-9676-3121

Machine Intelligence Research

Citation: M. Tan, Q. Chen, Z. Huang, Q. Wu, Y. Li, J. Zhou. Auto–3d–house design from structured user requirements. *Machine Intelligence Research*, vol.22, no.2, pp.368–385, 2025. https://doi.org/10.1007/s11633–024–1498–0