SceneExplorer: An Interactive System for Expanding, Scheduling, and Organizing Transformable Layouts

Shao-Kui Zhang, Jia-Hong Liu, Junkai Huang, Ziwei Chi, Hou Tam, Yong-Liang Yang, Song-Hai Zhang, *Member, IEEE*

Abstract—Nowadays, 3D scenes are not merely static arrangements of objects. With the development of transformable modules, furniture objects can be translated, rotated, and even reshaped to achieve scenes with different functions (e.g., from a bedroom to a living room). Transformable domestic space, therefore, studies how a layout can change its function by reshaping and rearranging transformable modules, resulting in various transformable layouts. In practice, a rearrangement is dynamically conducted by reshaping/translating/rotating furniture objects with proper schedules, which can consume more time for designers than static scene design. Due to changes in objects' functions, potential transformable layouts may also be extensive, making it hard to explore desired layouts. We present a system for exploring transformable layouts. Given a single input scene consisting of transformable modules, our system first attempts to derive more layouts by reshaping and rearranging the modules. The derived scenes are organized into a graph-like hierarchy according to their functions, where edges represent functional evolutions (e.g., a living room can be reshaped to a bedroom), and nodes represent layouts that are dynamically transformable through translating/rotating/reshaping modules. The resulting hierarchy lets scene designers interactively explore possible scene variants and preview the animated rearrangement process. Experiments show that our system is efficient for generating transformable layouts, sensible for organizing functional hierarchies, and inspiring for providing ideas during interactions.

Index Terms—Multi-Function Design, Interactive 3D Modeling, Scene Reconfiguration.

1 Introduction

With the continuous growth of urban populations, domestic scenes with functional transformability have become popular for better space utilization [1]. This motivates the emerging topic of transformable domestic space [2], which focuses on reshaping and rearranging furniture objects such that a room can present multiple functionalities [3]. Compared to traditional interior design, it further concerns accommodating different functions effectively within limited space [4].

In addition to the physical world, the importance of spatial experience within a virtual world is ever-increasing. High-priced NFT digital houses [5] and metaverse real estate [6] have brought forth another future for interior design. Online virtual conferences and art gallery exhibitions [7] utilize transformable modules and scenes. The Fortnite music festival [8] delivered a new immersive experience through reshaping layouts. With the development of novel applications, the usage of transformable scenes is highly promising. As a result, researchers have recently started to investigate flexible objects and scenes, ranging from reshaping objects [3], [9], [10] to reshaping scenes [11], [12], [13]. In this sense, objects involved in existing scene synthesis methods [14], [15], [16] can be treated as transformable modules with fixed states.

However, exploring potential transformable layouts is yet to be addressed. For example, Fu et al. [10] generate transformable modules given reference objects. How these objects are utilized in different scenes has yet to be explored. "Domestic Transformer" [12] and "Motion Planning" [11] have scenes evolving with time. However, the evolutions are pre-generated according to existing scenes, i.e., references are still needed.

This paper introduces SceneExplorer, which studies how to explore transformable layouts, as shown in Fig. 1. The only input we need is an initial scene of transformable modules. Our system automatically expands other possible transformable layouts. It calculates the schedules for reshaping/translating/rotating the modules, thus helping transformable layout designs. As the number of expanded layouts can be high, where thousands of layouts typically exist, our system also organizes a hierarchy of scenes to help designers efficiently explore transformable layouts. The expanded layouts are grouped as nodes according to functions. The edges represent functional evolutions from

- *Shao-Kui Zhang, Hou Tam and Song-Hai Zhang are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.*
  E-mail: shaokui@tsinghua.edu.cn, tanh24@mails.tsinghua.edu.cn, shz@tsinghua.edu.cn
- *Jia-Hong Liu and Ziwei Chi are with the Academy of Arts & Design, Tsinghua University, Beijing 100084, China.*
  E-mail: liujiaho23@mails.tsinghua.edu.cn, sineewave@gmail.com
- *Junkai Huang is with the Zhili College, Tsinghua University, Beijing 100084, China.*
  E-mail: huangjk21@mails.tsinghua.edu.cn
- *Yong-Liang Yang is with the Department of Computer Science, University of Bath, United Kindom.*
  E-mail: y.yang@cs.bath.ac.uk

Manuscript received - -, 2024; revised - -, -.
(Corresponding author: Song-Hai Zhang.)

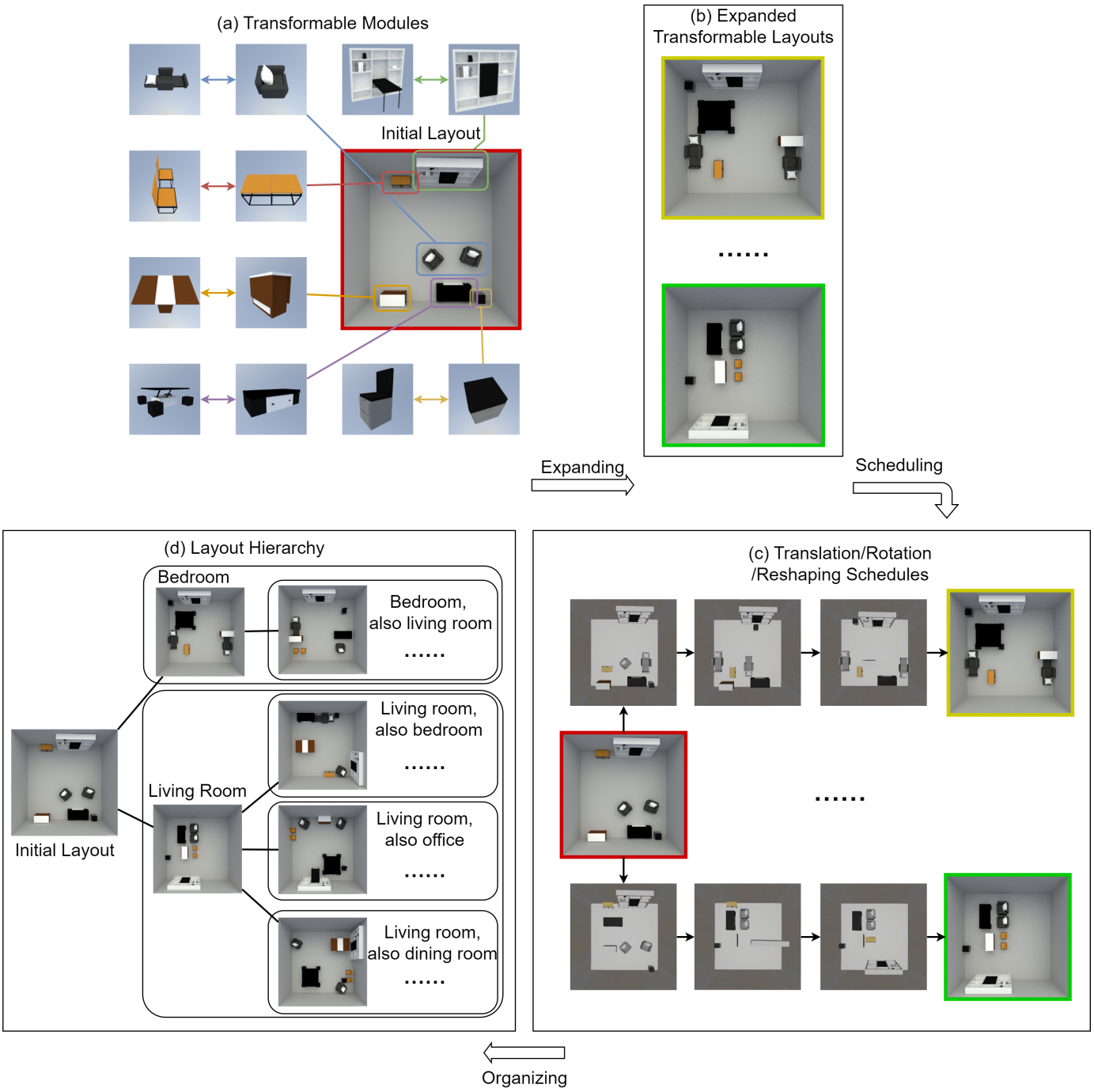
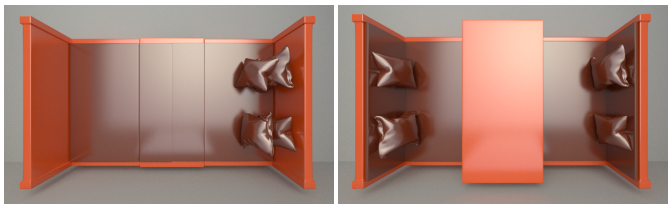


Fig. 1: Our work aims to explore transformable layouts given transformable modules. The only input is a layout of transformable modules (a), which refer to furniture objects that can change shapes to achieve other functions and appearances (see Fig. 2). Our system *expands* (b) the input layout to various transformable layouts with different functions by computing the *schedules* (c) of reshaping/translating/rotating the transformable modules (see Fig. 3). Additionally, to help users efficiently find a desired layout among complex and conceptual functions, our system *organizes* the transformable layouts into a hierarchy (d) representing the functional evolutions between them so that users can select the desired layout within a smaller range of numerous transformable layouts.



(a) A Double Bed (b) A Table-Chair Setting

Fig. 2: A transformable module that switches from a double bed to a table-chair setting. See the supplementary video for the reshaping animations.

the initial scene. Note that existing works only address translating and rotating objects [11], [13], while reshaping transformable modules and correlating scene functionalities are not considered as in our work. Our system's pipeline follows 1) expanding, 2) scheduling, and 3) organizing transformable layouts.

Our system first expands potential transformable layouts and computes the schedules from an initial layout to these (Sec. 3). Given an initial layout with several transformable modules, new layouts with different functions can be derived by reshaping these modules. For example, reshaping a bed into a table-chair setting (Fig. 2) makes the room one step closer to a living room while one step away from a bedroom. Each layout with one or multiple reshaped modules can be arranged leveraging spatial relation priors [17], [18].

After generating these diverse layouts, we compute the translations/rotations/reshapings of the modules required for scheduling the initial layout for each (Sec. 4). A rendezvous priority-based search algorithm is applied to determine the transfer path for each module. Fig. 3 shows an example of the schedule from one layout to another. Our system enables practical and efficient transformations between layouts.

Then, we formulate a hierarchy that organizes the transformable layouts (Sec. 5). In the hierarchy, each node represents a layout derived by reshaping several modules. Some branches are towards specific functions, e.g., a very "bedroom-like" bedroom. Some branches are towards mixed functions, e.g., a room with a bed, a coffee table and an office desk. Deriving branches depends on the initial layout and the expansions/schedules above. The hierarchy is interactive. Designers can explore it and click a layout (node), resulting in an animation of scheduling existing modules to form the layout.

We conducted three experiments to evaluate our system. First, a usability study shows that our system helps transformable/traditional scene designs for professional interior designers and non-expert users by providing ideas to them. Second, we compare the functional hierarchies with the manually crafted hierarchies by the designers, indicating a competitive practicality. Finally, we quantitatively demonstrate the competitive rationality of translating/rotating/reshaping schedules compared with the transformable layouts by interior design experts.

The primary contribution of this paper is an interactive system that explores the scene layouts of transformable object modules. Technically, we also have the following points:

- We expand transformable layouts given various transformable modules.
- We schedule how transformable modules are reshaped/translated/rotated across transformable layouts.
- We organize the transformable layouts as a hierarchy expressing their functional evolutions.

2 Related Works

2.1 Scene Synthesis

Synthesizing 3D scenes has been explored considerably in the past decade. Research starts arranging objects by extracting spatial relation priors between objects [14], [15], [17], [19], [20], [21]. The optimization methods [22], [23] vary, but the overall target is to make the translations/orientations/scales of objects as coordinated with priors as possible [24]. With the rapid development of neural networks, researchers also use generative methods based on deep learning [25], [26], [27], [28] to synthesize indoor scenes. Literature [29], [30], [31] applies Variational Auto-Encoder, which encodes the indoor scene and its hierarchy into latent codes and reconstructs scenes from sampled latent code. Different from VAE, Transformer [26] fulfils the indoor scene autoregressively [32], [33]. Diffusion Model (DM) [27], [34] and the DM-based image/3D synthesizing algorithms [35], [36], [37] improve the results of image-based scene synthesis [38], [39], [40], [41], leveraging aesthetic image generations. Scene graphs are also constructed leveraging DM [42]. Interactive scene synthesis is also investigated to facilitate user preferences, which typically pop up objects based on user-given positions/categories [16], [43], [44], [45], [46]. Our work differs from the above as it targets transformable domestic space [2]. While our system can automatically generate a hierarchy with all transformable layouts, we also allow users to explore the transformable layouts interactively.

2.2 Transformable Layout Design

Interior designer Gary Chang's "Domestic Transformer" [12] allows a 32-square-meter apartment to be reshaped into 24 scenarios. Depending on the time of day, the apartment can dynamically shift along the tracks of functional walls, reshaping flexibly into a kitchen, library, laundry room, dressing room, living room, bedroom, dining area, and bar. Similarly, the "Infinite Living" design by Crossboundaries [3] utilizes deformable modular furniture walls to enclose the area. The modular wall autonomously contracts and transforms as time progresses, allowing the space to evolve from a small office area into an expanded living room suitable for gatherings. These works require manual adjustments to achieve the layout reshapings, thus being complex for users without a design background.

MIT's "CityHome" project [13] explores intelligent indoor scene reshaping, enabling a furniture box to automatically deform and alter its layout through gesture and voice control. This work emphasizes the interaction of individual transformable modules, while our system emphasizes the interaction of multiple transformable layouts. The design

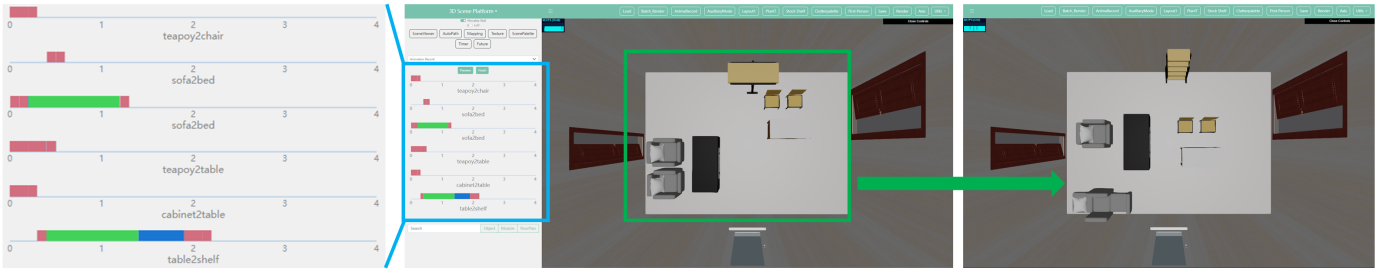


Fig. 3: Our system schedules transformable modules concerning their translations (Red Bars), rotations (Blue Bars), and reshapings (Green Bars). A bar can be a "reshaping", "translation" or "rotation". The length of a bar is the duration in seconds, e.g., the reshaping of the second "sofa-to-bed" last for a second. Each line refers to a transformable module, which can be translated/rotated/reshaped multiple times, resulting in multiple discrete bars. The schedule in this figure shows a living-dining room (The Green Box) is dynamically changed to a bedroom through translations/rotations/reshapings of transformable modules. Please refer to a supplementary video for the animated demo.

project "House ATO" [3] and "LiftTiles" [47] created a series of transformable boxes (modules) to connect indoor and outdoor spaces, achieving a transition from domestic to social functions through their adaptable design. However, because the furniture boxes can only unfold and are not movable, they cannot be freely leveraged for expanding transformable layouts, resulting in a lack of flexibility. In contrast, this paper recognizes transformable modules as dispersed, lightweight and standalone objects contributing to the layouts. Our system can be integrated with mobile swarm robots [48], [49] or lifting robots [50] to achieve a synergy between virtual environments and actual indoor spaces.

2.3 Transformable Module and Layout Generation

Fu et al. [10] introduce a paradigm for assembling transformable modules by programming vertical and horizontal movements of components. Their paradigm can generate a transformable module given reference objects, e.g., generating a desk-bed module given an existing desk and bed. It focuses on arranging parts within transformable modules, and one primary constraint is how each part follows the reference objects. Compared with their method, we focus on arranging transformable modules in scenes. One of our goals is appropriate schedules for reshaping transformable modules. Xiong et al. [11] introduce an algorithm for programming an arrangement between two input scenes. Their algorithm focuses on the floorplan migration, which ensures that objects in the initial layout match the reference layout as much as possible. In contrast, our system explores unknown scenes. Also, Xiong et al. [11] combine objects, thus forming more oversized objects of reference layouts, while our system calculates when and where modules should transfer.

Garg et al. [9] introduce a novel tool for interactively creating reconfigurables, i.e., transformable modules and layouts. During designers' editing of a reconfigurable concerning space and time, their tool interactively notifies whether collision(s) occurs at a different time than the designer's current timepiece. It also supports automatic view guidance for collision, collision resolution hints, etc, but it focuses on elaborately and interactively creating design-based states. In contrast, we leverage existing reconfigurables and automatically explore potential layouts that the reconfigurables can generate.

Additionally, reinforcement learning is utilized to plan and augment scene arrangement, e.g., the Scene Mover [51] and PEARL [52] by Wang et al. Sun et al. [53] refine objects' arrangements using neural networks. However, the above methods still focus on reshaping a layout to another instead of exploring potential layouts to be reshaped.

3 Expanding Transformable Layouts

The input of this section is an initial layout in room R that comprises multiple transformable modules. Users can either manually design the initial layout or generate it using existing scene synthesis methods [24]. A transformable module is denoted as M, and the collection of transformable modules in room R is $\{M_1, M_2, ..., M_n\}$, where n denotes the number of transformable modules. Each module M_i possesses m_i ($m_i \geq 2$) individual states, denoted as $\{O^i_1, O^i_2, ..., O^i_{m_i}\}$. For any module, every two different states can be reshaped into each other, changing its structure, function, and appearance during reshaping. In a given state, a module is a static furniture object, such as a bed, a table, or a chair. In the initial scene, each transformable module M_i is in its u_i-th state, having a specific position $\mathbf{p}_i$ and orientation θ_i. Then, the initial configuration for M_i is denoted $Q_i = (O^i_{u_i}, \mathbf{p}_i, \theta_i)$. Consequently, the initial layout can be expressed as $L_{init} = (R, \{Q_i\}_n)$, where $\{Q_i\}_n = \{Q_1, Q_2, ..., Q_n\}$.

More layouts with diverse functionalities can be expanded by altering the states of transformable modules and rearranging the modules. Any module with a different state may lead to significantly different room functionalities. For instance, if a transformable module switches from a bed to a desk, the room will likely be considered a study room rather than a bedroom. Therefore, we explore the layouts according to different combinations of states. A "function node" $N_{j_1,j_2,...,j_n}$ is used to represent the expanded layouts comprising states $\{O^1_{j_1},, O^2_{j_2}, ..., O^n_{j_n}\}$, as shown in Fig. 4. The total number of nodes $c(N)$ can be computed by a straightforward combination: $c(N) = \prod_{1 \leq i \leq n} m_i$. Consequently, numerous potential transformable layouts will exist with the increase in the number of transformable modules in the initial layout and the number of states.

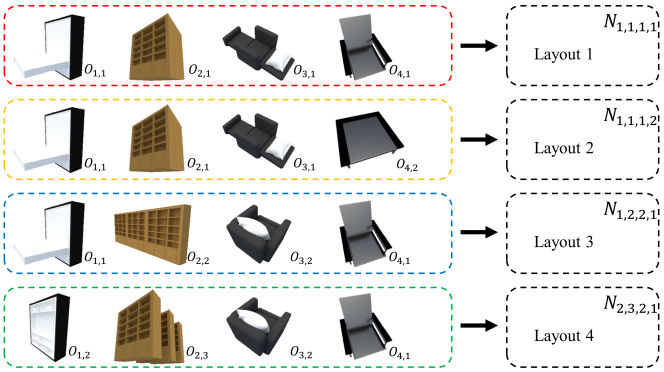


Fig. 4: An example of different function nodes with the same set of transformable modules. When one or multiple modules change their states, we must rearrange them since the overall function may change significantly.

Determining the layout for any N is a well-studied scene synthesis problem with static furniture objects [24]. We utilize a rule-based approach using the spatial relation priors assigned to the modules [17], [18]. These priors specify the relative positions and orientations among various objects or between an object and a wall. Given the modules and their corresponding priors, we sequentially place them according to a priority scheme, where larger objects (i.e., having a big scale) are prioritized as they are more likely to cause collisions. Objects with among-object priors are arranged as groups. An among-object prior indicates how objects are arranged locally, e.g., several sofas surround a coffee table, so objects satisfy such a prior can be arranged together.

However, when generating these layouts, we must consider not only the quality of the layouts but also how easily the original layout can be modified, which is summarized into three principles: (1) Each module should undergo minimal translation and rotation. (2) Larger modules require more effort to move than smaller ones. (3) The target position for any module should be unoccupied by other modules to avoid the additional effort needed to free up space. Optimal results can only be achieved when all three principles work together. Fig. 5 showcases the ablation layouts expanded from an initial layout, with three layouts among them demonstrating the impact of removing one principle at a time. We employ two approaches to address the above principles: generation enhancement and loss-based filtering.

First, we enhance our rule-based layout generation method by considering the positions and angles relative to the initial layout. Specifically, when placing a module relative to the wall and/or other modules, we prioritize positions and orientations closest to the original configuration, i.e., the given initial layout. Additionally, we observe that certain module states are inherently unsuitable for moving or rotating in a transformable design. For example, heavy objects like bookshelves or bunk beds are better left stationary. To address this, we invite a professional designer to annotate all module states, identifying which states should be considered "fixed". When generating new layouts, we ensure all module states marked as "fixed" remain stationary without any movement or rotation. This enhancement

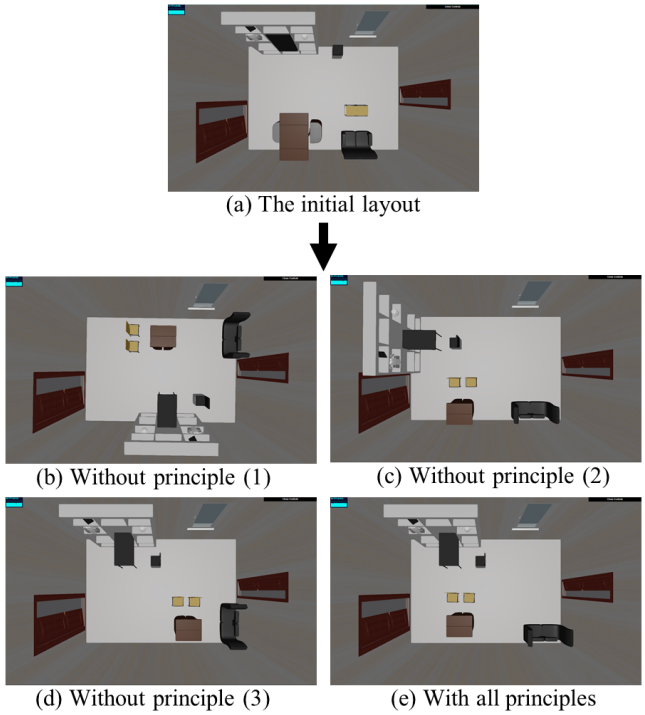


Fig. 5: An example of expanding offices (b-e) from a living-dining room (a), with different principles guiding the arrangement. (b): Without minimizing translations and rotations, significant layout changes can occur. (c): Not emphasizing larger modules, such as the bookshelf-desk set in the figure, can result in high effort for moving them. (d): Neglecting current occupation may cause unnecessary effort to free up space. Specifically, the table is moved to where the sofa occupies, forcing the sofa to move and rotate. (e): Our generation enhancement and loss-based filtering approaches account for all three principles.

leads to more practical and user-friendly layouts for homeowners.

Second, we generate many candidate layouts for each function node and filter them to obtain better-quality results. The rule-based method can efficiently generate dozens of candidate layouts with different priors or placement schemes. Given the initial layout $L_{init} = (R, \{(O^i_{u_i}, \mathbf{p}_i, \theta_i)\}_n)$, each candidate $L' = (R, \{(O^i_{u'_i}, \mathbf{p}'_i, \theta'_i)\}_n)$ is computed a heuristic loss score $\eta = \eta_1 + \eta_2$, following Equation 1 and 2:

$$\eta_1 = \sum_i \min\{w_i, w'_i\} * \left(\frac{|\mathbf{p}'_i - \mathbf{p}_i|_M}{R_l + R_w} + \frac{|\theta'_i - \theta_i|}{2\pi}\right) \quad (1)$$

$$\eta_2 = \sum_i \min\{w_i, w'_i\} * I(P(O^i_{u'_i}) \cap \bigcup_{j \neq i} P(O^j_{u_j}) \neq \emptyset) \quad (2)$$

In the above equations, $P(O^i_j)$ denotes the projection polygon for O^i_j on a 2D plane, with w_i being the area of this polygon. $\mathbf{p}_i/\theta_i/w_i$ is a module's position/rotation/area in the initial layout, and $\mathbf{p}'_i/\theta'_i/w'_i$ is the target position/rotation/area. Equation 1 estimates the minimum effort required for moving and rotating the modules. Specifically, $|\mathbf{p}'_i - \mathbf{p}_i|_M$ is the Manhattan distance between the

original and the target positions, normalized by the room's length R_l and width R_w. The term $|\theta_i' - \theta_i|$ represents the angle required for rotation. The area of the projection polygon is the weight applied for balancing modules of different sizes. A smaller value of η_1 generally indicates less effort and is therefore preferable.

Equation 2 determines whether other modules block the target configuration of any module. If a blockage occurs, additional effort may be required to successfully reposition the obstructing modules to implement the changes. The indicator function I assigns a value of 1 as a penalty when such a blockage occurs (and 0 otherwise). Therefore, η_2 serves as a supplementary term that addresses possible collisions.

After calculating the loss scores η for all candidate layouts, we retain only a few with the lowest scores for each function node. These selected layouts will be evaluated in the next section upon scheduling computation.

4 Scheduling Transformable Layouts

In the previous section, we expand new layouts from the initial layout, where valid schedules that convert the initial layout to the target ones are yet to be computed. Since most layouts have significant differences, scheduling the transition from one layout to another involves changing all transformable modules' positions, orientations, and states. During this process, it is crucial to ensure that the furniture objects do not collide with each other or with the room's boundaries. We address this scheduling as a 2D motion planning problem. The objects in the room are represented by their projections on the ground plane. Given the positions $\mathbf{p}$ and orientations θ, the initial layout is represented as $L = (R, \{(O^i_{u_i}, \mathbf{p}_i, \theta_i)\}_n) = (R, \{Q_i\}_n)$, and the target layout is represented as $L' = (R, \{(O^i_{u_i'}, \mathbf{p}_i', \theta_i')\}_n) = (R, \{Q_i'\}_n)$. Then, the schedule can be denoted as $T : L \rightarrow L'$.

We propose an algorithm to compute schedule T, detailed in Sec. A of the supplementary document. The core idea of the algorithm is to continuously attempt to transfer each transformable module from its starting configuration Q_i to the target configuration Q_i'. If the transfer of a module fails, we try transferring other modules until all options are exhausted. If these attempts fail, the algorithm will try to clear the path by moving a module and then resume the transfers.

Transferring a transformable module M_i requires securing a collision-free path from position $\mathbf{p}_i$ to $\mathbf{p}_i'$ and providing sufficient space for reshaping from state S_{u_i} to $S_{u_i'}$ and rotation from θ_i to θ_i'. Therefore, we design a four-step transfer process for any module: translate, rotate, reshape, and translate again. We translate the module to an appropriate position, rotate and reshape it (the order can be swapped), and finally move it to the target position. We regard it as a rendezvous problem [54] and utilize a heuristic priority-based search approach to optimize the path for the movements, as shown in Fig. 6. Note that not every step is necessary in all situations. For example, reshaping or rotation can be skipped if the state or orientation remains unchanged. Similarly, if the room is spacious, the module may only need to be moved once.

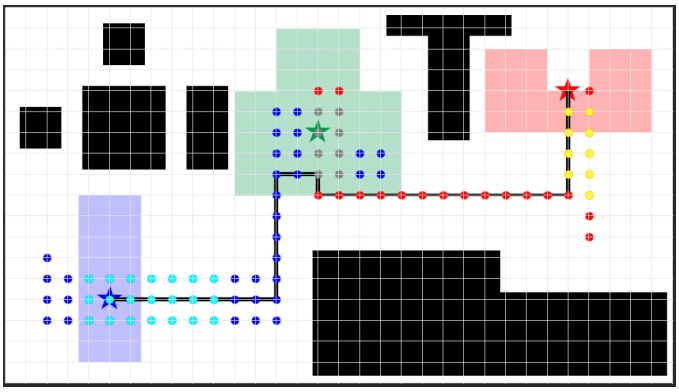


Fig. 6: We build a discrete grid to determine the path for transferring a module. Static obstacles are marked in black. The red, blue, and green polygons with stars represent the module's starting, target, and intermediate configurations. The dots in different colors stand for reachable positions for the three polygons. Yellow (red+green), cyan (blue+green), and grey (red+blue+green) dots indicate positions fit for multiple polygons of the corresponding colors. In order to complete a transfer, a module must move (along a red/yellow path) to an intermediate position (marked in grey), perform in-place rotation and reshaping, and move (along a blue/cyan path) to the destination. We utilize a rendezvous priority-based search to find a valid transfer path (marked in black).

The proposed algorithm is applied to compute the schedules from the initial layout L_{init} to all other layouts. As described in Section 3, each function node retains several filtered layouts. After scheduling each layout, the actual effort value (i.e., loss/cost) required for translation and rotation is recalculated in a term similar to Equation 1. Unlike the heuristic approach used earlier, this calculation reflects the final effort needed to complete the schedule with reference to the transfer path and actions. The layout with minimal actual effort is retained as the optimal choice for each node. Although jamming modules could potentially cause scheduling failures, the heuristic methods and multiple candidate layouts in Section 3 ensure that a well-scheduled layout can almost always be achieved for each function node.

The resulting function nodes form a radial structure, with the initial layout L_{init} acting as a bridge connecting all layouts, guaranteeing a transformation sequence between any pair of layouts. In this structure, any two transformable layouts L_1 and L_2 can be scheduled into each other in at most two steps. For instance, if there exist schedules $T_1 : L_{init} \rightarrow L_1$ and $T_2 : L_{init} \rightarrow L_2$, then $T : L_1 \rightarrow L_2$ can be represented as $T = \overline{T}_1 T_2$, where $\overline{T}_1$ denotes the inverse schedule of T_1. In comparison, achieving a one-step change between any pair of layouts requires $\frac{k(k-1)}{2}$ schedules for k layouts, which is inefficient. Therefore, in our implementation, we only compute these direct schedules between any two layouts when there are a small number of layouts.

Sec. 3 and 4 can be applied to traditional non-transformable objects, i.e., $m_i = 1$, where the objects are only translated and rotated. No reshapings change their

states. Since our system is devised for transformable modules and layouts, we use transformable modules with two or more states as examples.

5 Organizing Transformable Layouts

After introducing transformable layouts, the layouts generated in the previous step can exceed 2^n, where n represents the number of transformable modules. Our pipeline allows users to discard a few unwanted layouts, which is unnecessary. Subsequently, our system generates a hierarchy to filter and organize the layouts based on functionalities to let users find favorable layouts easily, as shown in Fig. 7. For example, if two layouts share very similar functions, they can be categorized as the same node on the hierarchy; if two layouts have the same major function but have different minor functions, they can be placed on the same branch in the hierarchy. Consequently, there may be only a few nodes in the hierarchy even if many layouts are generated since some may be clustered into a single node.

We define the representation of a layout's functionality through a vector, where each entry represents a function type, such as a living room or bedroom. The total number of function types is denoted as r. A larger entry indicates a stronger function of the room and vice versa. A room can have multiple functions, e.g., being a living room and an office. The functionalities of a layout depend on the functionalities of the objects included. We achieve functionality vectors $\vec{\lambda}_{O^i_j} = (\lambda_{O^i_j,1}, \lambda_{O^i_j,2}, \ldots, \lambda_{O^i_j,r})$ for the specific state O^i_j (i indices an object and j indices a state), using a dataset as illustrated in Sec. B of the supplementary document.

Given a generated layout $L = (R, \{Q_i\}_n)$ from Sec. 4, where $Q_i = (O^i_{u_i}, \mathbf{p}_i, \theta_i)$, an importance factor w_i is assigned to each module state Q_i which has existed in the layout. w_i equals the projected area of the i-th module state's geometric convex hull on the ground, i.e. the area of $P(O^i_{u_i})$. This is based on the observation that larger objects are more likely to serve as the focal points [55] or primary functional components of a room [56]. We then assess the functionality of a layout from two perspectives: the presence of objects and the relationships between these objects.

If an object (a state) exists in a room, it will possess the corresponding functionalities of the object, e.g., the inclusion of a dining table allows a bedroom to serve as a dining room. We evaluate the object-presence functionality of the layout using the vector $(v^{presence}_{L,1}, v^{presence}_{L,2}, \ldots, v^{presence}_{L,r})$, which is calculated by summing up the product of weight w_k and functionality value $\lambda_{O^k_{u_k}}$ of all the objects in the layout, as shown in Equation 3.

$$v^{presence}_{L,i} = \sum_{1 \le k \le n} w_k \times \lambda_{O^k_{u_k}, i} \tag{3}$$

Besides, more than using the area to weigh the importance of a module is required. We also weigh the importance of using the prior relations between/among objects. We refer to the spatial relation priors [17] and assume that if a set of objects is associated with a specific prior, the objects can collaboratively perform coherent functionalities [57]. More specifically, suppose the indices of the modules' states included in the prior are represented as $\{\{\pi_{k,o}\}_{l_k}\}_l$, where l_k is the number of modules involved in the k-th prior, l is the number of all the prior used in the layout, and $\{\pi_{k,o}\}_{l_k}$ are all the indices of modules involved in a single prior. Note that we do not use the spatial information, i.e., geometric features of priors, between the modules here, but only use the information that these modules appeared in the same prior set. We then use $\Pi_{k,o} = O^{\pi_{k,o}}_{u_{\pi_{k,o}}}$ to represent the transformable module states that appeared in the prior, which is the $\pi_{k,o}$-th module state in the layout. In particular, we assume $\Pi_{k,1} = O^{\pi_{k,1}}_{u_{\pi_{k,1}}}$ is the dominant transformable module state in the prior. Finally, we evaluate the collaborative functionalities by adding up the products of two values for all the prior involved: the dominant module's functionality value $\lambda_{\Pi_{k,l},i}$, and the geometric mean of the weight of the dominant object and the sum of the weight of other objects ($\sqrt{\sum_{2 \le x \le l_k} w_{\pi_{k,x}}} \times \sqrt{w_{\pi_{k,1}}}$). Formally, it can be represented by Equation 4.

$$v^{prior}_{L,i} = \sum_{1 \le k \le l} \lambda_{\Pi_{k,l},i} \times \sqrt{\sum_{2 \le x \le l_k} w_{\pi_{k,x}}} \times \sqrt{w_{\pi_{k,1}}} \tag{4}$$

However, only applying the two metrics above may still lead to wrong assessment in certain circumstances. When extracting functionality values from the vector, we use the whole layout's functionality to represent the actual functionality it has performed in the layout. Consequently, a layout may include modules without their corresponding functions. For instance, a layout identified as a bedroom would attach a high functionality to all objects involved, including objects that do not serve the primary functionality of sleeping, such as an ordinary table. Consequently, a layout can be identified as a bedroom even if it lacks a bed, nightstand, wardrobe, etc. Other irrelevant objects can still contribute to the "bedroom".

To address this, we established a specific category of modules essential for the layout. For instance, a bedroom requires a bed module, while a dining room necessitates a dining table module. If a layout does not include its necessary modules, we multiply a penalty factor of less than 1 by the relevant attribute. The factor will be set as 1 if necessary modules exist. In general, the functionality vector $\vec{V}_L = (V_{L,1}, V_{L,2}, \ldots, V_{L,r})$ of layout L is calculated by Equation 5. It applies the penalty factor to the weighted sum of the object-presence functionality $v_{L,1,i}$ and the prior functionality $v_{L,2,i}$. $C^{presence}$ and C^{prior} are the weight coefficients controlling the significance of the two functionalities, and C_p is the penalty factor. We choose these constants based on our observation, though users can manually adjust them based on their preference of different factors determining the room type. An intuition of Equation 5 is shown in Fig. 7.

$$V_{L,i} = C_p \times (v^{presence}_{L,i} \times C^{presence} + v^{prior}_{L,i} \times C^{prior}) \tag{5}$$

After calculating the functionalities of layouts, we generate the hierarchy using a breadth-first-search method, as described in Sec. C of the supplementary document. Each node in the hierarchy contains layouts with similar functionalities, and each edge leads to different significant functionalities. We define the similarity of two layouts based on

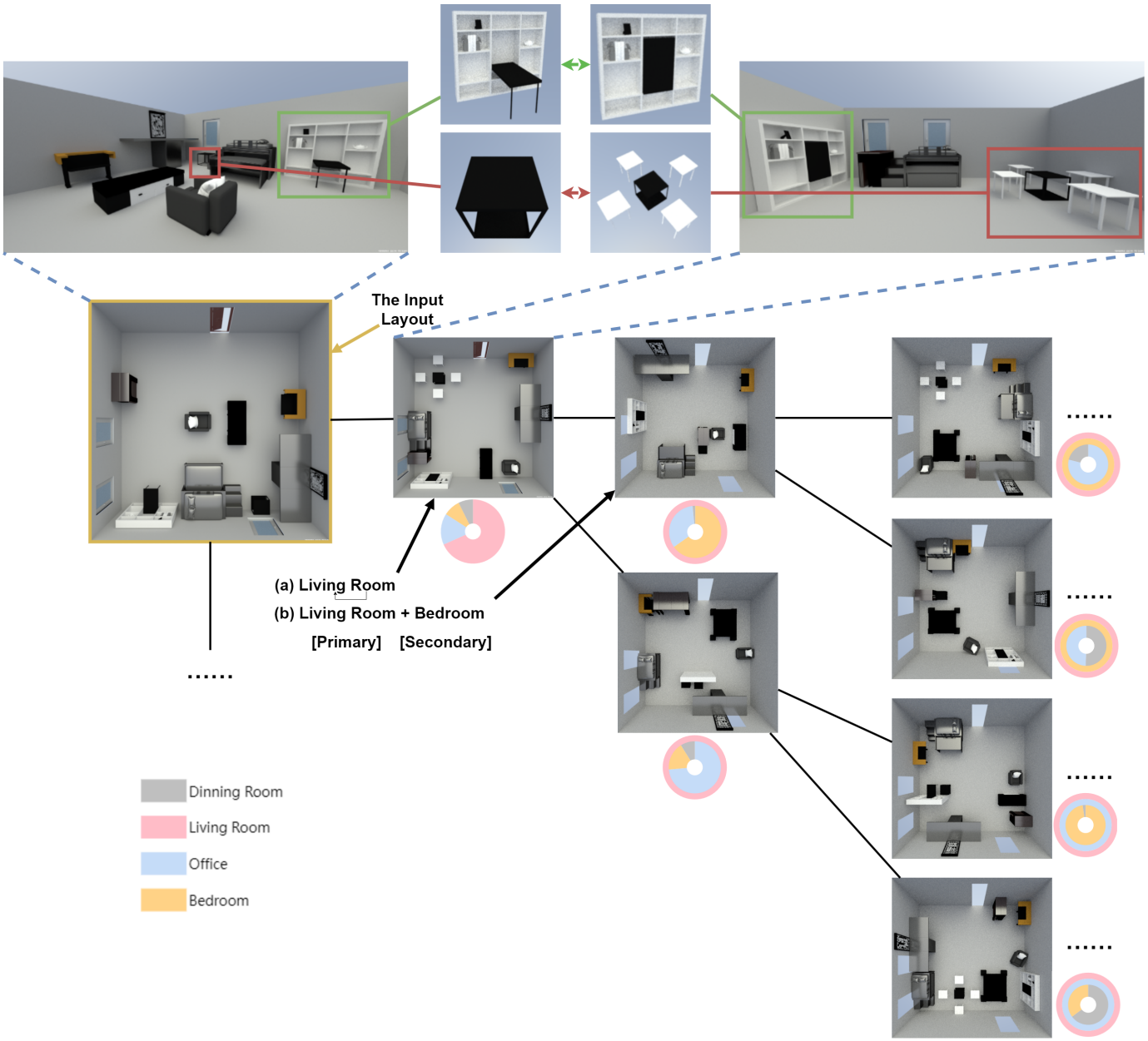


Fig. 7: Illustrating the proposed hierarchy. Given an input scene of furniture objects (Yellow), our system organizes a hierarchy with each node referring to a transformable layout and each edge referring to the functional evolution between two nodes. Layouts are transformable to each other by schedules of objects' translations, rotations and reshapings (Top Red and Green Boxes). (a): A layout is assigned "Living Room" by calculating its functionalities ($V_{L,i}$). (b): The layout is still a "Living Room" (Pink Outer Ring) with the secondary "Bedroom" functionality. The cabinet module (Green Box) is foldable, where a small table can be folded. The coffee table set (Red Box) is assemblable, where four white chairs can be spread out.

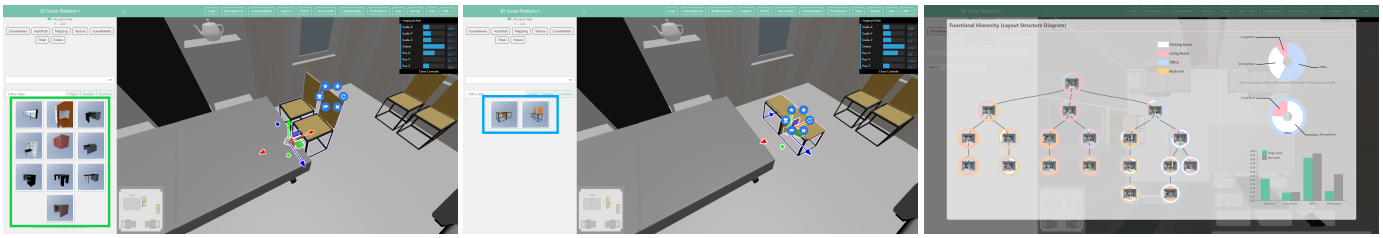
the L^∞ norm of the difference between their functionality vectors. Formally, if two layouts L_1 and L_2 are considered similar, then $\|\vec{V}_{L_1} - \vec{V}_{L_2}\|_\infty \leq \epsilon$, where ϵ is a predefined constant. Therefore, if a user desires a specific functionality, they can choose from layouts within a single function node rather than constantly switching between different nodes and branches.

Theoretically, the functionality vectors can cause two layouts with totally different module states to have the same vector in Equation 5. If such a situation happens, these layouts will still be grouped as a single node, as illustrated above, and the users can still access these layouts from the nodes.

6 Experiments

6.1 System Setup

Our interactive system runs on a Windows 10 personal computer with 10 cores, 3.7 GHz processors, and 32 GB RAM. The generation time for arrangements depends on the number of explored layouts, the number of modules in the room, the available space for scheduling layout, the



(a) Searching and Manipulating Modules (b) Reshaping Modules (c) Reshaping Layouts

Fig. 8: (a): Our interactive system supports searching objects (Green Box) and translating/rotating them. The system supports local coordination, raycasting-based control, and numerical modifications for translations and rotations. (b): The system supports transformable modules. After clicking a module in the 3D scene, its corresponding states are shown in a panel (Blue Box). Clicking a state triggers a module reshaping. (c): The proposed hierarchy with function nodes.

complexity of the objects, etc. For a simple room with four modules, it takes approximately 0.1 seconds to generate a new layout along with the reshaping process. When determining the functional hierarchies, the computation time is as short as several milliseconds. The crucial notations used in Sec. 3, 4 and 5 are summarized in Tab. 1. Due to our limited budget, our transformable modules only support four function types: bedroom, living room, dining room, and office.

We conducted several user studies to evaluate our system and the results generated.

1) We evaluate the usability of our system through tasks of arranging objects. Participants are instructed to utilize our system to craft scenes based on transformable modules. We record the operation time with and without our system and invite them to rate our system (Sec. 6.2).
2) The proposed functional hierarchy is compared with the manually crafted hierarchy by the designers. Practicality and relevance are the criteria in this study (Sec. 6.3).
3) We compare the translations/rotations/reshapings schedules with the experts' designs. Participants rate both results regarding the rationality of the transformable layout designs (Sec. 6.4).

We design an interactive system enabling controls over generated hierarchies and transformable modules to support user studies. First, it supports searching/ translating/rotating modules, as shown in Fig. 8a. Searching also supports "state-based" searches, e.g., searching a table or a nightstand. Both find the same module but will be added with different states. To reshape a module, clicking a module will pop up its states. Users can select a state, and the module will reshape accordingly, as shown in Fig. 8b. The user can search for transformable modules, add them to the room, or remove them. The above basic features support the usability study (Sec. 6.2).

Second, an interactive panel shows the resulting functional hierarchy that connects transformable layouts, as shown in Fig. 8c. Each node is demonstrated as a bird's-eye-viewed layout, which is the target layout after reshaping layouts if clicking the node. When a user points to any of these nodes, a pie chart and a bar chart will appear, describing the functionality of the corresponding layout. Upon clicking a node, the system automatically plays the animation of reshaping the layout from the current layout and eventually stops at the target layout. This feature supports evaluating transformable layout organization (Sec. 6.3).

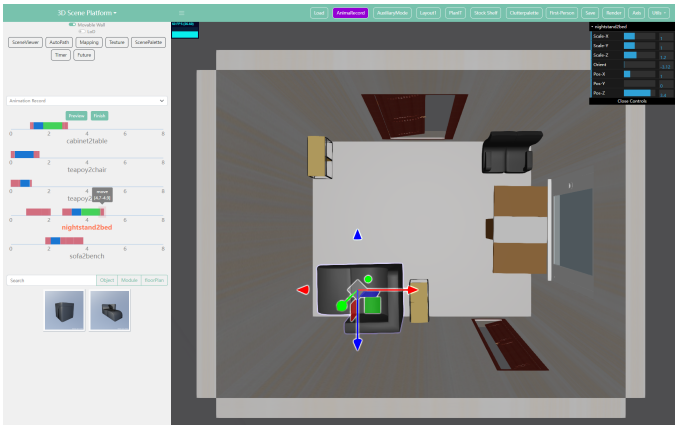


Fig. 9: The panel on the left consists of several timelines (transfers), each corresponding to a module. Each action refers to a time step, and all time steps of a module form its timeline. Selecting a module will highlight its timeline (Red Texts). Designers can easily edit or replay the actions that have taken place at each time step by hovering over the colored bars (red for translations, blue for rotations and green for reshapings). Designers can add an action by operating (translation/rotation/reshaping) on a module.

Third, our system enables designers to schedule the animation sequence for the transformable modules, as shown in Fig. 9. The panel records the timing of each action designers perform, such as translations, rotations or reshapings of specific transformable modules. Designers can preview the entire animation sequence they have created, edit it or reschedule (by dragging the bars) it until they are satisfied. Designers can also replay each animation segment inside a sequence by hovering over the corresponding bar, e.g., replaying a translation during the middle of the entire animation. This feature supports evaluating transformable layout schedules (Sec. 6.4).

Please refer to the accompanying video and Sec. D in the supplementary document for more engineering details/illustrations of the interactive system[1]. Sec. E in the supplementary also presents how we tune and decide the

1. The video and document are provided in the supplementary materials.

TABLE 1: A summary of key notations for describing our implementation in Sec. 3, 4 and 5.

Notation	Brief Description
M_i	The i-th **transformable module** in the room
O^i_j	The j-th **state** of the i-th module
$Q_i = (O^i_{u_i}, \mathbf{p}_i, \theta_i)$	The **initial configuration** of the i-th module
$L = (R, \{Q_i\}_n)$	A **layout** in room R with n modules
$N_{j_1,j_2,...,j_n}$	The **function node** with states $\{O^1_{j_1}, O^2_{j_2}, ..., O^n_{j_n}\}$
$P(\cdot)$	The **polygon** representing a ground projection
$\vec{\lambda}_{O^i_j} = (\lambda_{O^i_j,1}, \lambda_{O^i_j,2}, \ldots, \lambda_{O^i_j,r})$	The functionality **vector** for the state O^i_j
w_i	The area of $P(O^i_{u_i})$
$(v^{presence}_{L,1}, v^{presence}_{L,2}, \ldots, v^{presence}_{L,r})$	The object-presence functionality **vector** of a layout
$\{\{\pi_{k,o}\}_{l_k}\}_l$	The **indices** of the module states included in the priors
$\Pi_{k,o}$	A **transformable module** appeared in the prior, equals to $O^{\pi_{k,o}}_{u_{\pi_{k,o}}}$
$(v^{prior}_{L,1}, v^{prior}_{L,2}, \ldots, v^{prior}_{L,r})$	The prior functionality **vector** of a layout
$\vec{V}_L = (V_{L,1}, V_{L,2}, \ldots, V_{L,r})$	The overall functionality **vector** of a layout

parameters of our system. We understand that a good work would have transparency and enable reproducibility. Hence, our source code and dataset is publicly available[2].

To improve our practicality, we further design a taxonomy of transformable modules, as shown in Fig. 10. Five templates classify how the transformable modules are reshaped, including Move-Lift, Move-Slide, Move-Stretch, Rotate-Fold and Rotate-Spin. Each template has a template module expressing the template's general idea. Users can select a template. Then, several predefined transformable modules that are faithful to the template are shown on the UI. An upload button is added, allowing users to annotate new modules according to the templates. Users can craft and upload new transformable modules using existing novel systems [9], [10]. The transformable modules' taxonomy is beyond the scope of this paper. Please refer to Sec. F in the supplementary document for our current taxonomy details.

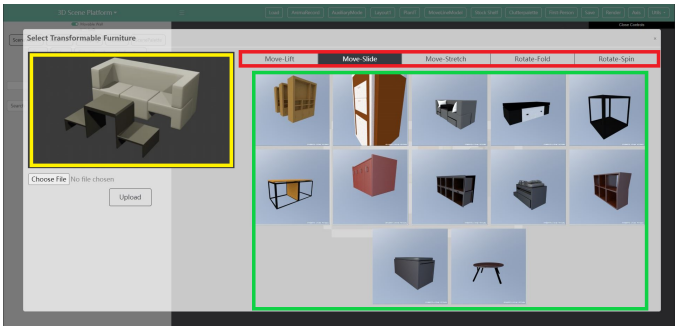


Fig. 10: Our system allows finding transformable modules according to templates. Five template are designed (Red Box). Each template has a "template module" to the left (Yellow Box), as the example show how the transformable modules are reshaped in that template. Clicking a template shows several predefined transformable modules (Green Box) belonging to the template. Users can also upload new annotated modules. Please refer to Sec. F in the supplementary document and supplementary video for the five template.

6.2 Evaluating SceneExplorer

Given an initial scene (room/layout), our system generates a series of potential layouts with diverse functionalities and builds a functional hierarchy supporting the dynamic arrangements. We aim to examine whether it helps users craft scenes with transformable modules.

We invite thirty-one participants. All participants have experience in interior design but are not necessarily experts. Every participant is tasked with reshaping an existing layout to another one based on a specified description. The description includes the functionality and some constraints, e.g. "a spacious bedroom with two beds and a nightstand between them". Provided with an initial layout, a participant is instructed to complete the same task through two approaches: with and without the help of the proposed system. When not using our system, participants only have access to the basic operations of the modules. Our technical staff gave in-person instructions on how to use our interactive system. The participants can freely use the system until they are familiar with it.

In contrast, with our system, participants can use the functional hierarchy, check the functionalities of transformable layouts, and edit modules based on generated content. For each approach, the duration of completion is recorded. Every user is allocated 3 initial layouts, so there are 6 transformable scene designs to do in sum. We randomize the order of the tasks and ensure that the same tasks are separated.

After completing all the tasks, participants are asked to rate our system and the manual approach based on several criteria, as outlined in Tab. 2. Since the last four criteria are specific to our system's features and do not apply to the manual approach, participants only rate the first three criteria for the manual approach. Each criterion was evaluated using a 5-point Likert scale.

We first compare the time required to complete the reshaping task. Fig. 11a illustrates the distribution of com-

2. https://github.com/Shao-Kui/3DScenePlatform/#sceneexplorer

TABLE 2: The criteria for evaluating our system in Sec. 6.2. Users are asked several questions before rating them.

Name	Corresponding Question
Usefulness	Is the system useful?
Learnability	Is the system easy to get started?
Convenience	Is the system easy to operate?
Smoothness	Are the reshaping animations smooth?
Aesthetics	Is the system aesthetically pleasing?
Hierarchy	Is the functional hierarchy clear?
Functionality	Is the functionalities of the layouts clear?

pletion time. Since the tasks are simple, most users complete them in a short time. However, users tend to operate longer without the help of the proposed system (manual). We calculate that the mean completion times with and without our system are 133.3 seconds and 154.7 seconds, respectively. During the experiment, we observed that some users spent a lot of time searching for an appropriate layout that matched the descriptions, significantly increasing the average completion time. On the other hand, the required operation time after a layout selection in our system is significantly decreased. Our system generally helps users with more efficient indoor designs concerning transformable modules.

The scores for the criteria are illustrated in Fig. 11b and 11c. Our system receives excellent scores on the first four criteria (usefulness, learnability, convenience, and smoothness), each with an average score higher than 4. The learnability is exceptionally good, indicating that our system enables an easy and user-friendly start-up. Additionally, our system receives competitive usefulness, learnability, and convenience scores compared with the manual approach. However, our system's scores for the last three criteria are slightly lower. Some users reflect that the presentation of the hierarchies and the functionalities can take some time to comprehend and require improvements. Still, our system is recognized by both inexperienced and experienced designers for its effectiveness in aiding the design and exploring potential layouts. We interviewed the participants about their emotional/intellectual experiences on our system, e.g., how the system helped them, reduced their mental effort or saved their brainpower. They can also express the confused/frustrated parts of the system.

In general, we summarize three representative positive experiences. 29% of the participants found the system convenient, with comments like "I can visually see the proportion of functional zones" or "The system is convenient to arrange objects". 42% appreciated the system's efficiency, noting that "The system can be applied in the field of interior design to reduce costs and increase efficiency" or "The system can quickly conduct preliminary planning. Only minor adjustments are needed in the future". 84% of them felt the system helped generate ideas and assist space design. They mentioned, "The system can be used as a preliminary spatial design reference for interior designers" and "Designers can use the system to showcase solutions to non-experts". Note that we did not limit benchmarks to participants. They could freely state what impressed them. From the interview, we conclude that our system is novel at providing ideas to practical users.

Nevertheless, 32% of the participants suggest a better UI or art design, e.g., "System interface needs to be optimized" or "The system's aesthetics can be further improved". This indicates that the front-end engineering should be improved in the future.

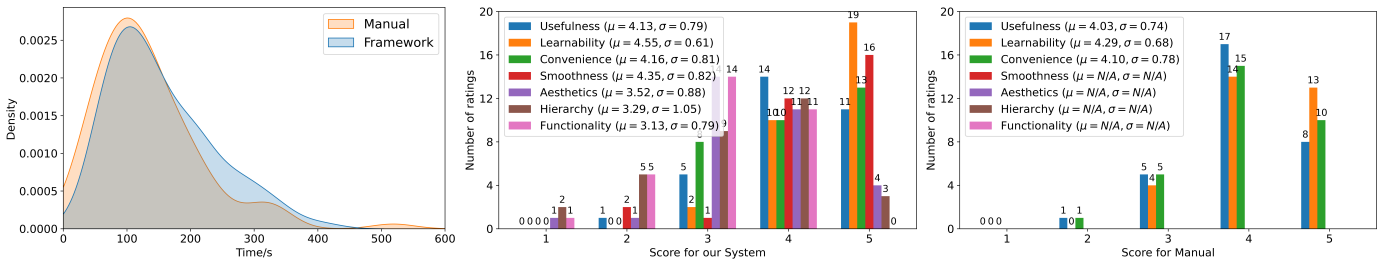
Another 27 participants were invited to compare the layouts generated with and without the help of the proposed system. The newly recruited participants are presented with two layouts in random order. They are asked to choose a favored one or choose "similar". They only compare them according to the static results, i.e., no schedules are involved. Sec. 6.4 further evaluates the schedules. The statistics show that newly recruited participants favour 42.6% of the reshaping results. 42.8% of the participants favor the human products, and 14.6% choose 'similar'. The statistics indicate that our system generates competitive layouts compared with human products.

6.3 Evaluating Organizations

We next evaluate the proposed functional hierarchy. Given a set of transformable layouts, our system automatically organizes a hierarchy based on the functionalities of the layouts. We randomly select generated hierarchies and invite expert designers to manually organize the hierarchies, given the same transformable layouts. The designers connect the transformable layouts to form a tree-like structure. Every layout is allocated a unique serial number (Identifier) and a brief explanation of the involved module states so designers can better understand it and do the task. The designers are free to infer the layouts' functionalities. The designers are also informed that a well-designed structure can help users understand the relationships and functionalities of layouts.

In total, fifteen independent hierarchies are designed by five experts paired with our corresponding results, with each hierarchy taking 1124 seconds on average to craft. We recruited another thirty-three participants. Each allocated five pairs of results. The newly recruited participants are university students with different majors. They are asked to compare the presented pairs of hierarchies. For every pair, two 5-point Likert scale ratings are required. First, participants rate the structures' practicality, indicating their ability to help locate layouts with specific functionalities. Second, participants rate the relevance, which judges whether relationships between adjacent layouts are reasonable. The participants were also presented with serial numbers and functionality explanations of all layouts.

We count the scores for practicality and relevance, respectively (Tab. 3). The practicality score measures how the hierarchy, along with the explanations, can help the users get a desired layout, and the relevance score measures the rationality of the hierarchy. For both criteria, our hierarchies receive better ratings than those manually crafted. The average scores for our results are 3.43 and 3.46 for practicality and relevance, respectively, higher than those for the designed ones (3.24 and 3.09, respectively). Overall, the proposed structure makes it easier to understand the groupings of different functionalities in the layouts. Therefore, our hierarchy can help users follow the functionalities and locate expected layouts better. During the experiment, most users preferred our results when they comprehended the flow of functionalities. We therefore conclude that the



(a) (b) (c)

Fig. 11: The results of experimenting with the system. (a): The time required to complete the design task is illustrated using a distribution graph. Manual (without the aid of our system) operations tend to be longer. (b) and (c): The scores for our system and manual operations, respectively. The last four criteria are exclusive to the proposed system. Our system receives excellent scores for usefulness, learnability, and convenience, being competitive to manual operations, highlighting that our system is easy to use. Users are also satisfied with our system for the remaining criteria since there are only a few low ratings (not higher than 2).

TABLE 3: The number of ratings for different scores in evaluating the proposed hierarchy. By comparing our results with those designed by experts in terms of practicality and relevance, we demonstrate that our framework scores higher in both metrics, especially for relevance.

Score	1	2	3	4	5
Ours-Practicality	1	22	62	65	15
Ours-Relevance	2	31	43	66	23
Crafted-Practicality	5	22	76	52	10
Crafted-Relevance	9	42	51	51	12

proposed functional hierarchy in our system works well in terms of both practicality and relevance.

6.4 Evaluating Schedules

Given the initial and target layouts, our system automatically computes the schedule. Specifically, every module experiences one or more translations/rotations/reshapings from the initial layout to the target layout. Given the same initial and target layouts, we invite expert designers to manually design transformable layouts using our interactive system.

All designers have a significant background in interior design and rich experience in manipulating industrial 3D scene applications. They are informed of the basic actions that can be applied to the modules and the usage of the schedule panel (Sec. 6.1). Every design starts with a given initial layout. Designers translate, rotate, or reshape every module so the layout can eventually be reshaped. The timelines and the action sequences can also be modified. Manual reshapings are saved in the same format as our generated results upon completion. During scheduling, one technical staff is accompanied for any questions. We record the time duration of each reshaping. In total, thirty independent transformable layouts are designed by six experts, with each reshaping taking 655.7 seconds on average to craft. The designs are then paired with our corresponding results.

We next recruit another thirty-four participants to perform the evaluations. The participants are university students with different majors, e.g., art, computer science, architecture, etc. Every participant is randomly allocated fifteen pairs of layout reshapings. The order of assigning reshapings is shuffled to minimize biases, and whether each process is generated or crafted is concealed. The participant is then instructed to watch the layout reshapings (schedules) as often as they want. A 5-point Likert scale concerning rationality is used. The layout reshapings of each pair are presented simultaneously. The pairs are assigned (distributed) evenly to the participants. The complete experiment takes about 30 minutes for each participant.

We count each score's total number of ratings, as shown in Tab. 4. For our system, 356 scores out of 510 (69.8%) are not lower than 3, which indicates that most users accept the generated transformable layouts in terms of rationality. For the manually crafted results, the ratio reaches 94.5%. Additionally, the expert designs received higher scores (4 or 5) than ours. Scores for our results have a mean of 3.03 and a standard deviation of 0.96, and the scores for the crafted ones have a mean of 4.05 and a standard deviation of 0.87. In general, the manually designed reshapings are more welcomed than those generated by our system. The main reason is that our system generates more redundant reshapings than the designed ones. Firstly, the algorithm can introduce additional actions due to some fail cases, which may be avoided when designed manually. Secondly, the process may take two steps (using the initial layout as the intermediate), leading to unnecessary positionings and moves. For the explanations, see Sec. 4. The above setbacks are acceptable since our system aims at collision-free and successful layout reshapings. We conclude that our system makes plausible dynamic arrangements in most cases.

On the other hand, our automatic system generates transformable layouts much faster. Our system takes less than 1 second to generate a transformable layout, while a designer takes more than 10 minutes to craft one manually.

6.5 Evaluating Effectiveness

We conduct analyses to demonstrate the effectiveness of the two refinement approaches (generation enhancement and loss-based filtering) in Section 3. We evaluate three methods: (1) a baseline method without the two refinement approaches, (2) a method with generation enhancement but without filtering, and (3) the current method with both approaches applied. One hundred initial layouts are gen-

TABLE 4: The number of ratings for different scores in evaluating the rationality of the reshaping processes. Although the crafted reshapings are rated better, our results are rational (scoring not lower than 3) in most cases (about 70%).

Score	1	2	3	4	5
Ours-Rationality	24	130	194	133	29
Crafted-Rationality	2	26	92	214	176

erated, each within a $7.5m \times 4.5m$ room containing five random modules. In total, approximately 3,500 function nodes and 98,000 layouts are expanded. These layouts then go through the scheduling process in Section 4, where an effort value is calculated if scheduling succeeds.

We use the average success rate and effort value as metrics to evaluate the three methods. The results show average success rates of 81.6%, 85.4%, and 90.5% and average effort values of 2.89, 2.45, and 2.19, respectively. These suggest that both refinement approaches significantly improve the scheduling success rate and reduce the effort required for translation/rotation/reshaping actions. Second, we examine whether the optimal layout is mistakenly filtered out. Specifically, if a candidate layout that could be successfully scheduled with the minimum effort value is removed by the loss-based filtering process, it is considered a mistake. Fortunately, the results indicate that such cases occur in only 16.7% of the function nodes, demonstrating that the filtering approach effectively retains layouts more practical to scheduling.

7 Conclusion

This paper presents an interactive system to explore transformable layouts based on transformable modules. Our system is summarized in three phases: expanding, scheduling and organizing. Our system enables automatic transformable layout expansion and interactive transformable layout editing. However, there are still limitations, and further improvements are required.

First, the algorithm of scheduling layouts in Sec. 4 may be refined to improve the success rate, remove redundant paths, and boost efficiency, as suggested in Sec. 6.4. Scheduling can sometimes fail. Some steps may introduce unnecessary actions and lead to redundancy. The computation time is also unstable due to heuristic approaches and randomness. A better algorithm could be more effective, efficient, and stable.

Second, although the hierarchy provides an organized and concise way to show the generated layouts, it is based on the functionality inferred by the algorithm. Different users may have different opinions on the functionality, which may lead to confusion if users' opinions and ours differ. Besides, it is hard to give detailed descriptions of the hierarchy, so it would take some time for the users to comprehend how each function node "functions". Future work may consider adding automatically generated natural language descriptions on nodes.

Third, more factors are yet to be considered in the computation of the functionalities of layouts described in Sec. 5. In our implementation, we ignore the positions and orientations when assigning importance factors to the modules. Additionally, the configurations of the room (e.g., windows and doors) do not affect the functionalities. Therefore, the computed result cannot accurately represent the actual function of a layout. In the future, we will take these factors into the computation process for more comprehensive evaluations.

Fourth, our system needs improvements for continuous state space, e.g., a table plate (the part that can be extended on the top) can stretched (extended) from 0 to 2 meters continuously. The transformable layouts are also continuous, given that the modules involved have continuous states. We still need future ideas to carefully select the representative transformable layouts or visualize the hierarchy yielded by the continuous transformable layout space.

Acknowledgments

We thank the reviewers for their thoughtful comments to help us improve the paper. We also thank Prof. Hongbo Fu for his valuable comments of improving this paper.

This work was supported by the National Key Research and Development Program of China (No. 2023YFF0905104), the National Natural Science Foundation of China (No. 62132012, 62402262) and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. Yong-Liang Yang is funded by UKRI grant CAMERA (No. EP/T022523/1). Shao-Kui Zhang is funded by Shuimu Tsinghua Scholar Program (No. 2023SM061), China Postdoctoral Science Foundation (No. 2024M751696), Postdoctoral Fellowship Program of CPSF (No. GZB20230353).

References

[1] R. Reuter, A. Snijders, and M. Bilow, "Space-saving techniques by the use of transformable architecture," *Delft University: Delft, The Netherlands*, 2006.

[2] J. L. Cabanes Ginés, "Graphic expression of the transformability of domestic space. from le corbusier to andrés jaque: A real and symbolic evolution," in *Graphical Heritage: Volume 2-Representation, Analysis, Concept and Creation*. Springer, 2020, pp. 367–377.

[3] K. Hara, *House Vision 3:2018 Beijing Exhibition*. CITIC Press Corporation, 2018.

[4] H. Husein, "Multifunctional furniture as a smart solution for small spaces for the case of zaniary towers apartments in erbil city," *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies*, vol. 12, pp. 1–11, 10 2020.

[5] K. T. Kit, "Sustainable engineering paradigm shift in digital architecture, engineering and construction ecology within metaverse," *International Journal of Computer and Information Engineering*, vol. 16, no. 4, pp. 112–115, 2022.

[6] A. Kemec, "From reality to virtuality: Re-discussing cities with the concept of the metaverse," *International Journal of Management and Accounting*, vol. 4, no. 1, pp. 12–20, 2022.

[7] L. Tomarchio, P. Herthogs, and B. Tunçer, "Hybrid art space typologies in singapore from social media data," *City, Culture and Society*, vol. 34, p. 100519, 2023.

[8] K. Moritzen, "Opening up virtual mosh pits: Music scenes and in-game concerts in fortnite and minecraft," *Journal of Sound and Music in Games*, vol. 3, no. 2-3, pp. 115–140, 2022.

[9] A. Garg, A. Jacobson, and E. Grinspun, "Computational design of reconfigurables," *ACM Trans. Graph.*, vol. 35, no. 4, jul 2016. [Online]. Available: https://doi.org/10.1145/2897824.2925900

[10] Q. Fu, F. Zhang, X. Li, and H. Fu, "Magic furniture: Design paradigm of multi-function assembly," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–12, 2023.

[11] G. Xiong, Q. Fu, H. Fu, B. Zhou, G. Luo, and Z. Deng, "Motion planning for convertible indoor scene layout design," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 12, pp. 4413–4424, 2020.

[12] S. Canepa, "Mechanization takes command, again," *Journal of Civil Engineering and Architecture*, vol. 14, pp. 565–571, 2020.

[13] H. Larrea-Tamayo, "Arkits: architectural robotics kits," Ph.D. dissertation, Massachusetts Institute of Technology, 2015.

[14] L.-F. Yu, S. K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. Osher, "Make it home: automatic optimization of furniture arrangement." *ACM Transactions on Graphics*, vol. 30, no. 4, p. 86, 2011.

[15] Q. Fu, X. Chen, X. Wang, S. Wen, B. Zhou, and H. Fu, "Adaptive synthesis of indoor scenes via activity-associated object relation graphs," *ACM Transactions on Graphics*, vol. 36, no. 6, pp. 1–13, 2017.

[16] S.-K. Zhang, Y.-X. Li, Y. He, Y.-L. Yang, and S.-H. Zhang, "Mageadd: Real-time interaction simulation for scene synthesis," in *Proceedings of the 29th ACM International Conference on Multimedia*, ser. MM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 965–973. [Online]. Available: https://doi.org/10.1145/3474085.3475194

[17] S.-H. Zhang, S.-K. Zhang, W.-Y. Xie, C.-Y. Luo, Y.-L. Yang, and H. Fu, "Fast 3d indoor scene synthesis by learning spatial relation priors of objects," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 9, pp. 3082–3092, 2021.

[18] S.-K. Zhang, W.-Y. Xie, and S.-H. Zhang, "Geometry-based layout generation with hyper-relations among objects," *Graphical Models*, vol. 116, p. 101104, 2021.

[19] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan, "Example-based synthesis of 3d object arrangements," *ACM Transactions on Graphics*, vol. 31, no. 6, p. 135, 2012.

[20] S. Qi, Y. Zhu, S. Huang, C. Jiang, and S.-C. Zhu, "Human-centric indoor scene synthesis using stochastic grammar," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. USA: IEEE, 2018, pp. 5899–5908.

[21] T. Weiss, A. Litteneker, N. Duncan, M. Nakada, C. Jiang, L.-F. Yu, and D. Terzopoulos, "Fast and scalable position-based layout synthesis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 12, pp. 3231–3243, 2018.

[22] Y.-T. Yeh, L. Yang, M. Watson, N. D. Goodman, and P. Hanrahan, "Synthesizing open worlds with constraints using locally annealed reversible jump mcmc," *ACM Transactions on Graphics*, vol. 31, no. 4, p. 56, 2012.

[23] K. Wang, M. Savva, A. X. Chang, and D. Ritchie, "Deep convolutional priors for indoor scene synthesis," *ACM Transactions on Graphics*, vol. 37, no. 4, p. 70, 2018.

[24] S.-H. Zhang, S.-K. Zhang, Y. Liang, and P. Hall, "A survey of 3d indoor scene synthesis," *Journal of Computer Science and Technology*, vol. 34, no. 3, p. 594, 2019.

[25] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[27] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851.

[28] Q. Fu, S. He, X. Li, and H. Fu, "Plannet: A generative model for component-based plan synthesis," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[29] P. Purkait, C. Zach, and I. Reid, "Sg-vae: Scene grammar variational autoencoder to generate new indoor scenes," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 155–171.

[30] M. Li, A. G. Patil, K. Xu, S. Chaudhuri, O. Khan, A. Shamir, C. Tu, B. Chen, D. Cohen-Or, and H. Zhang, "Grains: Generative recursive autoencoders for indoor scenes," *ACM Transactions on Graphics*, vol. 38, no. 2, pp. 1–16, 2019.

[31] H. Yang, Z. Zhang, S. Yan, H. Huang, C. Ma, Y. Zheng, C. Bajaj, and Q. Huang, "Scene synthesis via uncertainty-driven attribute synchronization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 5630–5640.

[32] D. Paschalidou, A. Kar, M. Shugrina, K. Kreis, A. Geiger, and S. Fidler, "Atiss: Autoregressive transformers for indoor scene synthesis," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 12 013–12 026.

[33] X. Wang, C. Yeshwanth, and M. Nießner, "Sceneformer: Indoor scene generation with transformers," in *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 106–115.

[34] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[35] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10 684–10 695.

[36] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8821–8831.

[37] R. Liu, R. Wu, B. Van Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick, "Zero-1-to-3: Zero-shot one image to 3d object," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 9298–9309.

[38] J. Tang, Y. Nie, L. Markhasin, A. Dai, J. Thies, and M. Nießner, "Diffuscene: Scene graph denoising diffusion probabilistic model for generative indoor scene synthesis," 2023.

[39] C. Fang, X. Hu, K. Luo, and P. Tan, "Ctrl-room: Controllable text-to-3d room meshes generation with layout constraints," 2023.

[40] L. Höllein, A. Cao, A. Owens, J. Johnson, and M. Nießner, "Text2room: Extracting textured 3d meshes from 2d text-to-image models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 7909–7920.

[41] J. Lei, J. Tang, and K. Jia, "Rgbd2: Generative scene synthesis via incremental view inpainting using rgbd diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. USA: IEEE, June 2023, pp. 8422–8434.

[42] C. Lin and Y. Mu, "Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior," in *International Conference on Learning Representations (ICLR)*, 2024.

[43] L.-F. Yu, S.-K. Yeung, and D. Terzopoulos, "The clutterpalette: An interactive tool for detailing indoor scenes," *IEEE transactions on visualization and computer graphics*, vol. 22, no. 2, pp. 1138–1148, 2015.

[44] S. Zhang, Z. Han, and H. Zhang, "User guided 3d scene enrichment," in *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*, ser. VRCAI '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 353–362. [Online]. Available: https://doi.org/10.1145/3013971.3014002

[45] S. Zhang, Z. Han, Y.-K. Lai, M. Zwicker, and H. Zhang, "Active arrangement of small objects in 3d indoor scenes," *IEEE transactions on visualization and computer graphics*, vol. 27, no. 4, pp. 2250–2264, 2019.

[46] S.-K. Zhang, H. Tam, Y. Li, K.-X. Ren, H. Fu, and S.-H. Zhang, "Scenedirector: Interactive scene synthesis by simultaneously editing multiple objects in real-time," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–13, 2023.

[47] R. Suzuki, R. Nakayama, D. Liu, Y. Kakehi, M. D. Gross, and D. Leithinger, "Lifttiles: constructive building blocks for prototyping room-scale shape-changing interfaces," in *Proceedings of the fourteenth international conference on tangible, embedded, and embodied interaction*, 2020, pp. 143–151.

[48] J. Lengiewicz and P. Hołobut, "Efficient collective shape shifting and locomotion of massively-modular robotic structures," *Autonomous Robots*, vol. 43, pp. 97–122, 2019.

[49] O. Medina, S. Hacohen, and N. Shvalb, "Robotic swarm motion planning for load carrying and manipulating," *IEEE Access*, vol. 8, pp. 53 141–53 150, 2020.

[50] R. Suzuki, H. Hedayati, C. Zheng, J. L. Bohn, D. Szafir, E. Y.-L. Do, M. D. Gross, and D. Leithinger, "Roomshift: Room-scale dynamic haptics for vr with furniture-moving swarm robots," in *Proceedings of the 2020 CHI conference on human factors in computing systems*, 2020, pp. 1–11.

[51] H. Wang, W. Liang, and L.-F. Yu, "Scene mover: Automatic move planning for scene arrangement by deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.
[52] H. Wang, Z. Wang, W. Liang, and L.-F. Yu, "Pearl: Parallelized expert-assisted reinforcement learning for scene rearrangement planning," *arXiv preprint arXiv:2105.04088*, 2021.
[53] J.-M. Sun, J. Yang, K. Mo, Y.-K. Lai, L. Guibas, and L. Gao, "Haisor: Human-aware indoor scene optimization via deep reinforcement learning," *ACM Transactions on Graphics*, vol. 43, no. 2, pp. 1–17, 2024.
[54] S. Alpern, "The rendezvous search problem," *SIAM Journal on Control and Optimization*, vol. 33, no. 3, pp. 673–683, 1995.
[55] K. Xu, R. Ma, H. Zhang, C. Zhu, A. Shamir, D. Cohen-Or, and H. Huang, "Organizing heterogeneous scene collections through contextual focal points," *ACM Transactions on Graphics*, vol. 33, no. 4, pp. 1–12, 2014.
[56] M. Saruwono, N. F. Zulkiflin, and N. M. N. Mohammad, "Living in living rooms: furniture arrangement in apartment-type family housing," *Procedia-Social and Behavioral Sciences*, vol. 50, pp. 909–919, 2012.
[57] S. Masran, Z. Nur Farhana, and N. M. Nik Mastura, "Living in living rooms: Furniture arrangement in apartment-type family housing," *Procedia - Social and Behavioral Sciences*, vol. 50, pp. 909–919, 2012, acE-Bs 2012 Bangkok, Sukosol Hotel, Bangkok, Thailand, 16-18 July, 2012.



Shao-Kui Zhang received the PhD degree of Computer Science and Technology from Tsinghua University, Beijing, in 2023. He is currently a postdoctoral researcher in the Department of Computer Science and Technology at Tsinghua University. His research interests include Computer Graphics, 3D AIGC and Multimedia Applications.



Jia-Hong Liu received the bachelor's degree of Computer Science and Technology at Tsinghua University. His research interests include virtual reality and 3D scene synthesis.



Junkai Huang is an undergraduate from Zhili College, Tsinghua University. His research interests include computer graphics and 3D scene synthesis.



Ziwei Chi received her Bachelor's degree from Tsinghua University's Academy of Art and Design in 2021 and completed her Master's in Design in 2024. Her research interests include intelligent interior design, generative models, and human-computer interaction.



Hou Tam received both his Bachelor's and Master's degree in Computer Science and Technology from Tsinghua University. He is currently pursuing his doctoral degree in the Department of Computer Science and Technology, Tsinghua University. His research interests include computer graphics and 3D scene synthesis.



Yong-Liang Yang is a Senior Lecturer in the Department of Computer Science, University of Bath. He received the B.S. degree and the Ph.D. degree of Computer Science from Tsinghua University. His research interests are broadly in visual computing and interactive techniques.



Song-Hai Zhang received the PhD degree of Computer Science and Technology from Tsinghua University, Beijing, in 2007. He is currently an associate professor in the Department of Computer Science and Technology at Tsinghua University. His research interests include computer graphics and virtual reality.