SceneDirector: Interactive Scene Synthesis by Simultaneously Editing Multiple Objects in Real-Time

Shao-Kui Zhang, Hou Tam, Yike Li, Ke-Xin Ren, Hongbo Fu, and Song-Hai Zhang, *Member, IEEE*

***Abstract*—Intelligent tools for creating synthetic scenes have been developed significantly in recent years. Existing techniques on interactive scene synthesis only incorporate a single object at every interaction, i.e., crafting a scene through a sequence of single-object insertions with user preferences. These techniques suggest objects by considering existent objects in the scene instead of fully picturing the eventual result, which is inherently problematic since the sets of objects to be inserted are seldom fixed during interactive processes. In this article, we introduce SceneDirector, a novel interactive scene synthesis tool to help users quickly picture various potential synthesis results by simultaneously editing groups of objects. Specifically, groups of objects are rearranged in real-time with respect to a position of an object specified by a mouse cursor or gesture, i.e., a movement of a single object would trigger the rearrangement of the existing object group, the insertions of potentially appropriate objects, and the removal of redundant objects. To achieve this, we first propose an idea of coherent group set which expresses various concepts of layout strategies. Subsequently, we present layout attributes, where users can adjust how objects are arranged by tuning the weights of the attributes. Thus, our method gives users intuitive control of both how to arrange groups of objects and where to place them. Through extensive experiments and two applications, we demonstrate the potentiality of our framework and how it enables concurrently effective and efficient interactions of editing groups of objects.**

***Index Terms*—3D scene synthesis, 3D scene editing, interactive 3D modeling.**

I. Introduction

3D scene synthesis benefits various applications, including metaverse [1], virtual reality [2], [3], computer vision [4], interior designs [5], etc. Many attempts (e.g., [6], [7], [8], [9]) have been made to synthesize 3D scenes automatically. However, as verified in [10] and [11], automatic layout generations often do not guarantee users' preferences. Typical interior designers usually have to listen to their customers and manually craft scenes according to the needs of customers while following interior rules [12], [13], [14]. Thus, an intelligent interactive tool is more practical and has also been investigated in recent years.

To allow interactive control of scene synthesis, existing literature considers "objects" as the targets of manipulation [10], [11], [15], [16], [17], i.e., as the control units, objects are successively inserted into scenes. While the users take full control of the synthesis process, these techniques do not offer quick overviews of synthesized results during the synthesis process. For example, a user might add several objects in the beginning. However, halfway through the synthesis, she/he finds that some objects are functionally/aesthetically/stylistically unsuitable, thus consuming more time due to a longer interactive session. Thus, in practice, existing interactive solutions still leave blanks in foreseeing variations on desired scenes and have gaps in simultaneously manipulating multiple objects.

To address these issues, we present SceneDirector, which provides a novel tool for interactive scene synthesis by simultaneously editing multiple objects, which we call "controlled objects", including a selected object and its related objects in the scene (e.g., the objects in the green boxes in Fig. 1). With our tool, when a user changes a position of a single object, our method automatically rearranges the rest of the related objects, as shown in Fig. 1. This is achieved by considering the concepts of layouts, which consist of style-compatible sets of objects, express strategies of arranging objects, and respond to human affordance [12], [13], [14], [18].

However, layout concepts are sophisticated [12], [13], [14], [18], especially when we want to acquire a specific one and explicitly apply it under various contexts. Thus, we try to express layout concepts through a data-driven process. Instead of preparing a large amount of data and training unified models such as [9], [19] and [6], we treat groups of objects with a consistent style and functions as a unique set, which we name *Coherent Group Set* (CGS). A CGS is created by a designer considering a specified concept of a style and functions, as shown in Fig. 2, e.g., a luxury style with a double bed set. A layout concept can thus be expressed by a CGS. Fig. 4 shows an example of a CGS, which contains 14 object groups. One of our contributions is to

Manuscript received 4 July 2022; revised 2 April 2023; accepted 14 April 2023. Date of publication 21 April 2023; date of current version 1 July 2024. This work was supported in part by the Natural Science Foundation of China under Grant 62132012 and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. Recommended for acceptance by Daniel G. Aliaga. *(Corresponding author: Song-Hai Zhang.)*

Shao-Kui Zhang and Hou Tam are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: zhangsk18@mails.tsinghua.edu.cn; th21@mails.tsinghua.edu.cn).

Yike Li and Ke-Xin Ren are with the Academy of Arts & Design, Tsinghua University, Beijing 100084, China (e-mail: lyk20@mails.tsinghua.edu.cn; rkx20@mails.tsinghua.edu.cn).

Hongbo Fu is with the School of Creative Media, City University of Hong Kong, Hong Kong (e-mail: hongbofu@cityu.edu.hk).

Song-Hai Zhang is with the Department of Computer Science and Technology, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China (e-mail: shz@tsinghua.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TVCG.2023.3268115, provided by the authors.

Digital Object Identifier 10.1109/TVCG.2023.3268115

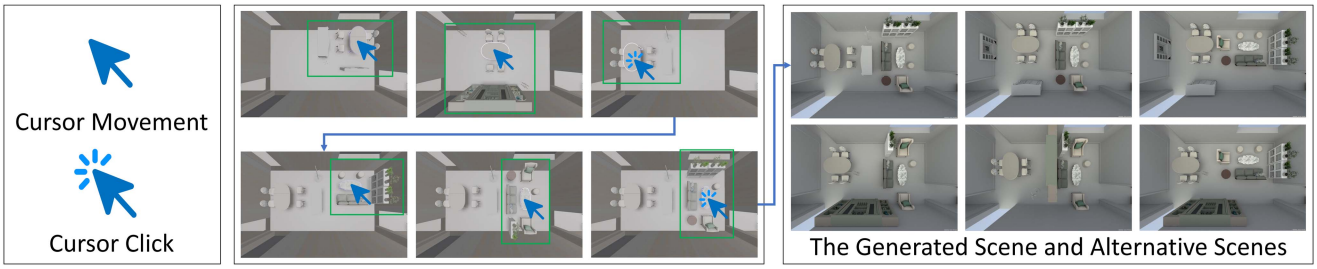


Fig. 1. We present SceneDirector for interactively synthesizing 3D indoor scenes by simultaneously editing multiple objects based on cursor movements and clicks (Left). Given a cursor movement at any moment, our framework automatically inserts, removes, translates, and rotates a group of objects (in green boxes) plausibly into a new scene in real-time, w.r.t. the cursor's current position (Middle). Mouse click would end an iteration, thus directly achieving a layout of a group of objects. After a few iterations, a plausible 3D indoor scene is synthesized while alternative results are available depending on user preferences (Right).

Fig. 2. Three examples of CGSs with different styles and functions. Each image shows a coherent group of objects. Each row refers to a CGS. All the coherent groups in a CGS share the same pool of objects (e.g., the same double bed and wardrobe).

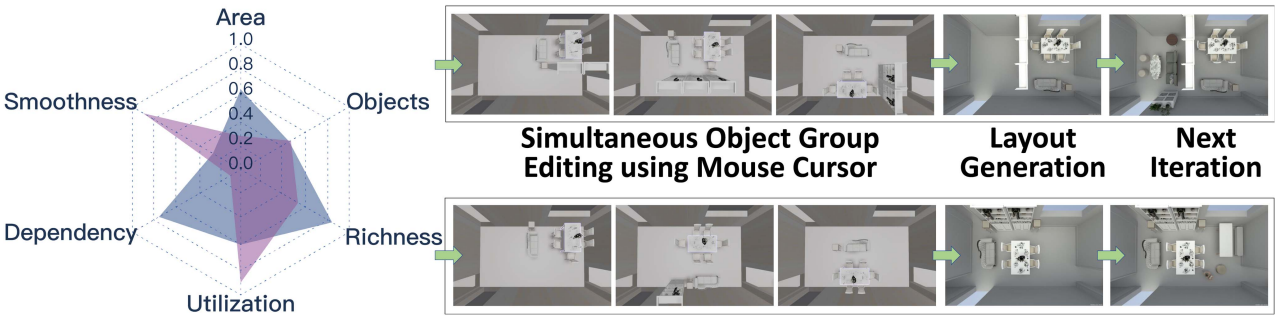


Fig. 3. The pipeline of our method. After a user changes the weights of the attributes, the strategy for object rearrangement changes accordingly. For example, the top row tries arranging objects as privately as possible (Blue in the attribute weight diagram). In contrast, the bottom row tries to arrange them as transparently as possible (Purple in the diagram). This entire process is considered a single interaction and is executed twice or more for exploration towards a final scene.

mathematically formulate a layout concept given a CGS design. The more groups in a CGS, the clearer the description of the corresponding concept. To support the CGS, existing datasets are insufficient due to their low densities of objects relative to layouts, so we also contribute a 3D scene dataset. Each CGS has more than 10 coherent groups to express its concept.

Additionally, we present layout attributes adjustable to the learned concepts. For example, a concept could derive a 3D scene requiring one or two walls surrounding it, while it could also derive a 3D scene as compact as possible, given the same location and room shape. User-specified weights of the layout attributes resolve such ambiguities. For example, a user can strongly demand a scene to be synthesized without a dependent wall by reducing the attribute "dependency" (e.g., the top row in Fig. 3). He/she could also increase the attribute "space utilization" to compact the scene (e.g., the bottom row in Fig. 3). Subsequently, our method gives convenient control over multiple objects while it enables the exploration of various layouts through tuning attributes.

The evaluation shows that our work significantly reduces the interaction time in various aspects, e.g., searching or transforming objects. We also conduct a usability study to evaluate how users feel more comfortable interacting with our system compared to the existing solutions [10], [20], [21]. We also conduct a user study to quantitatively demonstrate a higher plausibility and aesthetics of the resulting scenes of our method

than the scenes by the existing solutions. Finally, we present two fully implemented applications based on cursor movements and gestures in VR to show the potentiality of the proposed technique.

This paper makes the following contributions:[1]

- We present a new technique for interactively editing multiple objects concurrently in 3D scenes, using minimum inputs such as cursor movements on a desktop environment or hand gestures in VR.
- We quantify the attributes of layouts. Users can easily change how layouts are generated under different contexts by continuously adjusting the attributes.
- We propose the idea of "coherent group set", which helps restore layout concepts as much as possible. To achieve this, a dataset and an annotation platform are also presented.

II. Related Works

Automatic 3D scene synthesis focuses on the generation of entire layouts. Data-driven techniques range from mathematical formulations of interior rules [6], [23] to neural networks [9], [24]. They fit a series of models to examples in scene datasets and use the fitted or learned models to derive new layouts. For example, [6], [8], [25], [26], [27] and [28] formulate a set of models and optionally fit them with datasets of scenes if being data-driven and then synthesize scenes based on MCMC. [9], [19], [24], [29] and [30] directly yield 3D scenes by feeding random numbers or top views into neural networks. [23] and [31] propose synthesizing scenes based on objects and rooms' geometries. Although our method could be automated, we focus on developing an interactive system since desired scenes are not unique and often need user inputs to guide the synthesis process. Additionally, compared with data-driven approaches, our method expresses a concept with a very coherent set of object groups with functional and aesthetic compatibility instead of learning unified models.

Various scene synthesis techniques have been proposed to generate layouts by considering additional inputs. For example, He et al. [32] propose to generate 3D scenes by considering real-world blocks. Hand-drawn sketches have been utilized for deriving scenes [33]. The techniques in [34] and [35] interpret texts into scene graphs for generating scenes. RGB-D scans [36], [37], [38] or RGB images [39] have also been used for the physical and visual guidance of scene synthesis. Xiong et al. [40] propose to transfer reference layouts to 3D scenes with motion plans of objects, and the technique by Fisher et al. [7] synthesizes scenes with learned models and given example scenes. We propose using attribute weights as an additional input, which allows users to tune weights to adjust 3D scenes with respect to the concepts.

Interactive 3D scene synthesis is our paper's primary focus, and this topic has not been explored extensively. Savva et al. [16] present a tool for enabling object selections given mouse clicks. Some frameworks suggest detailed and small objects to enrich existing scenes [10], [11], [15]. [41] generates a series of scenes and gives users choices on them, thus conversely optimizing scenes presented to users. [42] generates scenes given user-specified constraints and examples. Some works investigate passive interactions [43], [44], where layouts are optimized to make workspaces more efficient given subject behaviours. Nevertheless, existing literature focuses on interactions incorporating only a single object each time instead of a group of objects. To our knowledge, we are the first to investigate the simultaneous editing of multiple objects.

[1]Code and dataset are publicly available at: https://github.com/Shao-Kui/3DScenePlatform#scenedirector.

III. Methodology

As discussed in Section I, a CGS-based dataset consists of various CGSs. Each CGS consists of a set of "groups", and each "group" has several objects. Each object has its index to a CAD model and its transformations in a 3D world. So a CGS is decided by both the object instances and the layout of these objects. In the same CGS, different groups share the same objects. For example, a nightstand can be used in a group with a dressing table or another group with a wardrobe. We will incorporate a CGS expressing the style and functions for synthesizing each scene. In other words, after fixing a CGS, our method synthesizes scenes concerning the CGS's groups and objects (with their transformations). This section discusses the methodology of expressing how objects are arranged leveraging CGSs. In Section IV, we will introduce how we create a CGS-based dataset.

We first introduce the online phase of our method. Fig. 1 shows an example of interactive scene synthesis with SceneDirector. A user first selects an existing object or inserts an object into the scene. When she/he suggests a single movement of the selected object using the mouse cursor, multiple objects related to the selected one are simultaneously rearranged or inserted into the scene (Section III-A). She/he can also specify different weights of the layout attributes so that objects are arranged further according to the user preference, as shown in Fig. 3. Our current implementation considers the following attributes: area, number of objects, richness, utilization, dependency, and smoothness (Section III-B). Overall, the process above is considered a single interactive session. Subsequently, in the offline phase, we multiplex each CGS and extract its attributes to restore the designer's concept of creating this CGS.

A. Coherent Group Set

A coherent group set is a set of groups $\mathcal{S} = \{G_i | i \in [1, N]\}$. Each group G_i has a finite number of objects, where groups of the same set $\mathcal{S}$ all hold the same and unique dominant object (e.g., the double bed in Fig. 4), typically the object to interact with w.r.t. mouse cursor movements or hand gestures. Subsequently, other objects would be adjusted according to the CGS. The unique dominant object in $\mathcal{S}$ is leading others, and each G_i is constructed according to its transformations and styles. A dominant object is selected by a user when interactively synthesizing scenes. Our method does not restrict selections of dominant objects, i.e., any object can be selected as a dominant object. For example, a user can select a coffee table or a sofa as a dominant object,

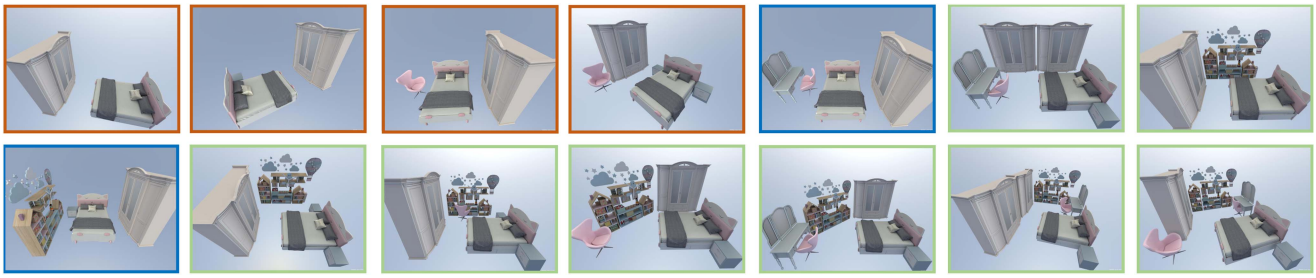


Fig. 4. A typical example of coherent group set with the gentle light style. All groups share the same object pool, which contains all alternative objects to express the concepts of layouts. Depending on contexts such as positions suggested by users and constraints such as attributes and room shapes, a concept could derive an open-plan layout (in orange boxes), a parallel layout (in blue boxes), or a semi-enclosed layout (in green boxes) [22]. The set is crafted by using very few objects within a pool of objects. In this paper, all the CGSs are crafted by professional interior designers to fully exploit our method.

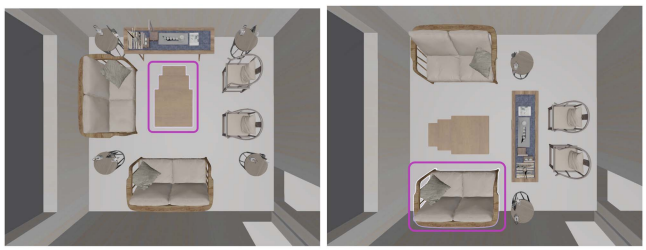


Fig. 5. In the same CGS, users can select different dominant objects (in the purple box) to control. (a) A dominant object can be a coffee table, and a user places it near the center of the room, thus generating objects surrounding it. (b) The user can also select a sofa as a dominant object, and the user places it near a wall.

Fig. 6. Illustration of derived spaces. A derived space (blue dotted lines) of a coherent group G_i is defined by the anchor, left, right, and depth (purple arrows) w.r.t the dominant object (blue circle). The distance between the nearest boundary to the dominant object creates the anchor. A derived space (green solid lines) of $\vec{x}$ and R is conducted by further expanding $D(G_i)$ so that a larger square is fitted to the room shape.

as shown in Fig. 5. When a user selects a dominant object such as a double bed, other objects, such as wardrobes, ottomans, nightstands, etc., are transformed w.r.t. the dominant object. For each interactive session, a user can only select one dominant object.

Each object in G_i has an index to a particular object instance, such as a CAD model. It also has a relative transformation w.r.t. the dominant object. Fig. 4 shows an original CGS designed by a professional interior designer, in which the dominant object is the double bed.

As illustrated in Fig. 1, our system calculates optimal coherent groups G^t in real-time to the movements of casted locations $\vec{x}$ and the room shape R. Specifically, this problem is formulated in (1), such that two more conditions are satisfied as shown in (2)

$$G^t = \text{argmax}\vec{\eta}^T g_i(\vec{x}, R, G^{t-1}), \tag{1}$$

$$\ni \begin{cases} G_i \in \mathcal{S} \cup \mathcal{S}' \\ d(\vec{x}, R) \geq D(G_i), \end{cases} \tag{2}$$

where G^t is a potential group of controlled objects to be synthesized, G^{t-1} is the currently calculated group before the cursor movement, and the position $\vec{x}$ could be a location casted by the mouse cursor or a hand gesture (Section IV). $g_i(\vec{x}, R, G^{t-1})$ returns the attributes of a coherent group G_i, and $\vec{\eta}$ contains the user-adjustable weights of the attributes. Some attributes, such as area and space utilization are proprietary values of G_i, while others require a context, e.g., R. Each $g_i(\cdot)$ corresponds to a $G_i \in \mathcal{S} \cup \mathcal{S}'$, where $\mathcal{S}'$ is obtained by expanding $\mathcal{S}$ in size, i.e., the number of object groups. So $\mathcal{S}$ and $\mathcal{S}'$ are both sets of object groups. Since too few instances in $\mathcal{S}$ are insufficient to express a layout concept, we multiplex $\mathcal{S}$ as far as possible by creating more variations of object groups in $\mathcal{S}$. First, we take the power set $\{G' \cup \{o_i^{dom}\} | G' \subseteq (G_i \setminus \{o_i^{dom}\})\} \setminus \{\emptyset\}$ for each G_i, where o_i^{dom} denotes the user-specified dominant object. Since the empty set is meaningless, it is removed from the power set. An interactive session should always contain the dominant object, so the "$\setminus\{o_i^{dom}\}$" operation guarantees that o_i^{dom} is not included in the power set operation and o_i^{dom} is added to all generated subsets. An example is to remove a sofa in a coffee table set layout of $\mathcal{S}$ and add this new layout to $\mathcal{S}'$. Second, for each G' in the power set, we further derive three groups by flipping it vertically, horizontally, and diagonally, e.g., swapping the left and right objects. Since for a large G_i, taking its power set could have excessive calculation pressure, we sample M ($M = 75$ in our implementation) G' to prevent $\mathcal{S}'$ from being too large. We will further discuss the impact of M in Section VI.

The $d(\vec{x}, R) \geq D(G_i)$ is a rigid requirement since we never want furniture to be outside or embedded into walls. $d(\cdot)$ and $D(\cdot)$ respectively return the derived space w.r.t. $\vec{x}$ and G_i. The illustration of derived spaces is given in Fig. 6, where four additional values are calculated for G_i: "anchor" μ_a, "left" μ_l, "right" μ_r, and "depth" μ_d. These four values are computed

as the shortest distance values from the dominant object to the corresponding four edges of the boundary. Specifically, μ_a is the distance from the dominant object to the nearest boundary. μ_d refers to the distance from the boundary opposite the nearest one to the dominant object. μ_l and μ_r are, respectively, the left-side and right-side expanded distance w.r.t. μ_a. For G_i, we aim to find a minimal rectangle tightly wrapping the objects in G_i. For $\vec{x}$, we aim to find a maximal rectangle tightly fitting the room shape R. Therefore, this condition requires that $(\mu_a, \mu_l, \mu_r, \mu_d)$ are all smaller than $d(\vec{x}, R)$.

A tiny collision may destructively influence results because the results of 3D scene synthesis are subjective to humans. To avoid collisions, we consider object-object detection and object-door/window detection. Each object is decomposed into a set of constituent components, e.g., desktop and desk legs. Any component of an object intersecting with a component of another object is considered a collision. Windows and doors are considered as single components that are cuboids elevated in the normal direction of their depending walls. If a collision occurs between $o' \in G_i$ and an existing entity in the scene, G_i will remove o as $G_i \setminus \{o'\}$.

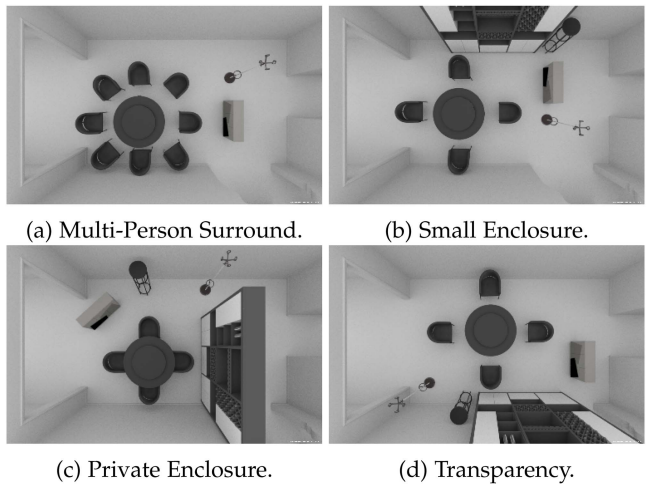
After calculating G^t, the layout of objects at $t-1$ time is re-organized accordingly, which could be messy since objects might drastically move together, thus reducing the visual experience. We then formulate how we re-organize objects from G^{t-1} to G^t. Assuming a set containing all objects is $\tilde{O} = \{o_i = (x_i, y_i, z_i, u_i, \gamma_i)\}$, the set containing the actual objects being placed to 3D scenes.

The tuple (x_i, y_i, z_i) stands for the current position of o_i relative to the dominant object in 3D space, and the variable u_i stands for whether o_i is already used in the scene. γ_i refers to a specific instance, e.g., a pink chair demanded by G^t. Assuming the target coherent group generated by a layout concept is $G^t = \{o'_j = (x'_j, y'_j, z'_j, \gamma_j)\}$, where each target object o'_j will be selected from $\tilde{O}$. The tuple (x'_j, y'_j, z'_j) stands for the target position of o'_j in the next transient movement. Thus, the problem is selecting a subset $\hat{O} \subset \tilde{O}$, so that the re-organization from G^{t-1} to G^t is tidy. The re-organization process is formulated in Algorithm 1. This algorithm is executed once for each transient time, so the positions of objects in $\tilde{O}$ constantly change.

Algorithm 1 follows a greedy approach. The outer loop iterates through the entire G^t to find a best-matched $o_i \in \tilde{O}$. Each o_i can be used up to once. If o_i is already occupied by a $o'_j \in G^t$, the u_i of o_i is marked as "true". The inner loop iterates through the $\tilde{O}$. The basic idea is to greedily find a nearest $o_i \in \tilde{O}$ based on the euclidean distance. After that, the position (x_i, y_i, z_i) of o_i will be set to the target position (x'_j, y'_j, z'_j) suggested by G^t. Subsequently, G^{t+1} would run Algorithm 1 based on the positions set by G^t. Note that the proposed coherent group set guarantee that all target objects $o'_j \in G^t$ can always find an object from the object set $\tilde{O}$.

B. Layout Attributes

3D scenes are complicated [45]. In some cases, attributes may affect how objects are arranged. They may also contradict each other. Thus, we present a typical usage of the attributes based on our observations and give users the balance control of the trade-offs. All the attributes are pre-computed before the pipeline described by (1) operates. Note that all attributes are eventually normalized in $[0, 1]$ to have the same magnitude.



(a) Multi-Person Surround. (b) Small Enclosure. (c) Private Enclosure. (d) Transparency.

Fig. 7. Four example scenes with the grey modern style for illustrating layout attributes.

Algorithm 1: Finding the Appropriate Objects Set for the Next Generated Layout G^t.

Input: Entire object set $\tilde{O}$ and target layout G^t.
Output: An optimal subset $\hat{O} \subset \tilde{O}$ with the overall minimal moving distances of objects.

```
for Each object o_i ∈ Õ do
    u_i = False;
Ô = ∅ ;
for Each o'_j ∈ G^t do
    D = ∞ ;
    for Each o_i ∈ Õ do
        if not u_i and γ_i == γ_j then
            d = ||(x_i, y_i, z_i) − (x'_j, y'_j, z'_j)||_2;
            if d ≤ D then
                I = o_i;
                D = d;
    Ô = Ô ∪ {I} ;
    o_i = (x'_j, y'_j, z'_j, True) ;
```

Area. In most cases, we want a coherent group G_i to be as large as possible to fill a given room. The area of G_i refers to the area of the derived space of G_i (i.e., dashed blue rectangles in Fig. 6), where the gap spaces among objects are considered due to aesthetic and affordance. In practice, this attribute frequently contradicts space utilization $A_u(G_i)$ (to be introduced soon). For example, the layout in Fig. 7(a) is more compact, but the layout in Fig. 7(b) is larger.

Space Utilization. This attribute $A_u(G_i)$ describes how efficiently G_i exploits the given space measured by $A_a(G_i)$. The space utilization is calculated by the following equation:

$$A_u(G_i) = A_p(\mathcal{P}(o_1) \cup \ldots \cup \mathcal{P}(o_N))/A_a(G_i), o_i \in G_i, \quad (3)$$

where $\mathcal{P}(\cdot)$ projects each object $o_i \in G_i$ into the ground using the mesh of o_i. By taking the union of each projected mesh, a polygon is unified, and its area is calculated by $A_p(\cdot)$. Therefore, $A_u(G_i)$ equals the ratio of the unified polygon to the area expanded by $D(G_i)$. Note that area and space utilization measure the layout differently. A higher value of area results in a larger derived space in Fig. 6, while a higher value of space utilization results in smaller gaps among objects. Both attributes can be considered measuring in a floorplan view, where no perspective views are involved.

Number of Objects. This attribute $A_n(G_i) = |G_i| - 1$ counts the number of objects in group G_i, excluding the dominant object. This attribute is used with the other attributes. For example, to acquire a compact scene with as many objects as possible, a user may tune both $A_n(G_i)$ and $A_u(G_i)$ up. Otherwise, being used individually, this attribute may not yield the expected results for users.

Richness. This attribute $A_r(G_i)$ measures the potential functionality of G_i, by counting the number of sub-groups inside G_i. For example, a double bed with two nightstands in a bedroom is considered a sub-group, and a dressing table with an ottoman is another sub-group. Thus, dividing G_i into sub-groups is equivalent to finding potential relations in G_i, where the relations can involve three or more objects. We employ an existing method [31] to determine such relations.

Dependency. For home decorations, many objects are designed to be strictly placed against walls, e.g., the cabinet in Fig. 7(b). In contrast, objects such as cabinets and wardrobes could also be assembled for zoning spaces at the early stage of the residential design [12], [14], e.g., the scene in Fig. 7(c). Thus, formulated in (4), dependency $A_d(G_i)$ measures how G_i is inclined to require walls to support it. A higher value of A_d results in "wall supporting" while a lower value results in "space zoning". Entries in $\vec{\xi}$ are either 0 or 1, denoting the dependencies w.r.t. the anchor, left, right, and depth. Calculating the entry follows the same rule of calculating $D(G_i)$ for coherent groups. For example, if a wardrobe is against the left side of a coherent group, the "anchor" of the wardrobe results in the "left" of the group. The function $\phi(\cdot)$ truncates the distances to 0 or 1, i.e., if the anchor is sufficiently close to the wall, the dependency is considered satisfied, and vice versa. $\circ$ denotes the element-wise multiplication, where the L1 norm finally sums the entries of the result vector together. Therefore, we first calculate the differences between the two derived spaces and truncate them. The corresponding entries are valid as long as $\vec{\xi}$ denotes those dependencies in some (or all) boundaries are required.

$$A_d(G_i, R) = ||\phi((d(\vec{x}, R) - D(\mathcal{G}_i(\vec{x}))) \circ \vec{\xi})||_1. \quad (4)$$

Smoothness. To interact with coherent groups, we realize that the transition from one G^t to another G^{t+1} could be overdramatic, e.g., objects in G^t are swapped instead of modifying the transformations of merely one or two objects. As a result, smoothness is calculated between groups to measure their differences. $A_s(G_i, G_j)$ takes in two groups. We follow the method of Fisher et al. [46], which uses graph kernels [47] to measure how two scenes are different. For node kernels, we follow their solutions,

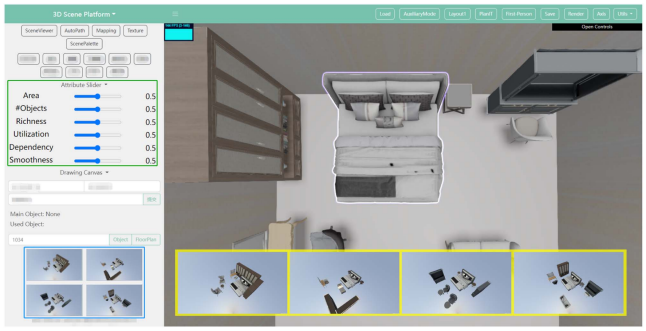


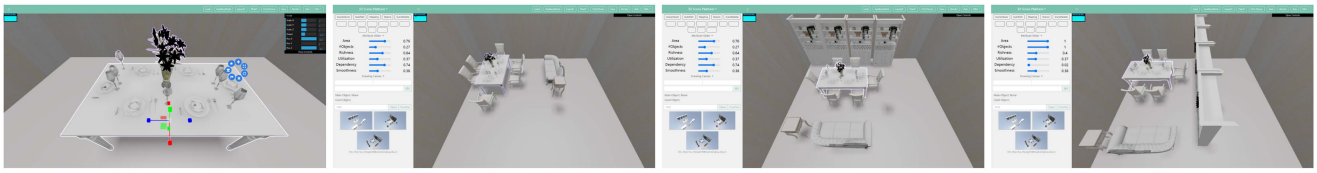
Fig. 8. The prototype system with the proposed CGS-based method. On the left, the attribute panel (in the green box) allows users to tune the weights of the attributes, and the search panel (in the blue box) allows users to swap alternative CGSs (highlighted in the yellow bottom box) w.r.t. the dominant object (highlighted in purple). On the right, users can select a dominant object by clicking it. The double bed moves accordingly with the cursor's movement, and the layout is adjusted in real-time.

but for edge kernels, groups in CGS may have identical relationships (Enclosure, Horizontal Support, Vertical Contact, Oblique Contact in [46]) between objects. The Kronecker delta kernel used in [46] returns whether the two edges being compared are tagged with the same contact type. It does not apply in our cases where objects are distanced, and the distance may vary due to the size of the room area, e.g., the scenes in Fig. 7(b) and (d). Therefore, measurements based on transformations in the 3D space are required.

We thus modify the method in [46] by exploiting the relative direction between the dominant and subordinate objects. Edges are defined between each subordinate object o_i and the dominant object as unit vectors of o_i's relative direction concerning the dominant object. The kernel between two edges is the inner product of their unit vectors and is clamped to 0 if less than 0. This metric shows how consistent the directions of two subordinate objects are with the dominant object. The distance is excluded since it might devalue the similarity metric as the room area goes larger or smaller. Section C of the supplementary document, available online, further shows an example of calculating smoothness between four scenes.

IV. System and Dataset

In this section, we design a prototype system to show how we utilize CGS for interactive scene synthesis/editing. As shown in Fig. 8, the UI of our system consists of three parts. First, in a 3D scene, a user could select a dominant object and enter the CGS mode. Theoretically, a dominant object could be any entity. For simplicity, this paper recognizes entities such as coffee tables or double beds as dominant objects, typically representing the functionality of coherent groups. Second, it has a panel for users to tune the weights on layout attributes. Since the attributes are normalized, the weights are in the range of $[0, 1]$, where 0 denotes ignoring an attribute, and 1 denotes fully engaging an attribute. Third, users could select various CGSs in a search panel, where each CGS stands for a particular concept and is represented by its final form, a layout derived from the concept to the greatest



(a) (b) (c) (d)

Fig. 9. An interactive editing process. The interacted object is highlighted in purple. (a) in the beginning, users could rescale the dominant object and thus rescale the entire CGS. (b) when the cursor is close to the corner, the space constraint (see (1)) is hard to satisfy, so a small scene is compromised. (c) given sufficient space, the CGS generates a scene as large as possible. (d) a more private scene is presented by changing the attribute weights, e.g., reducing the dependency.

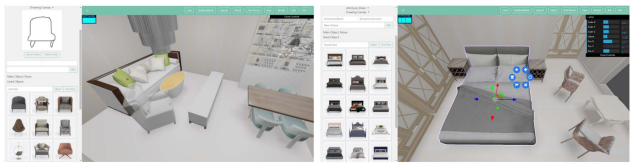
extent without any constraint. In Fig. 8, the four presented scenes stand for four different CGSs with the dominant double bed in the 3D scene. When a user clicks a CGS in that panel, the layout concept is swapped accordingly, and our method operates on the newly selected concept.

Fig. 9 shows a typical interactive process of our system, where a user sets a proper scale of the dominant object w.r.t. the room and starts to edit multiple objects. Different layouts are established, given different layout attributes and casted cursor positions. Clicking the cursor would add the suggested layout to the scene and end an interaction. A supplementary video, available online, shows more demonstrations of our system in real-time.

To implement our ideas, we need a CGS dataset. Initially, we tried 3D-Front [48], which, however, has the following problems: (1) The usage of dominant objects w.r.t. subordinate objects is over sparse, i.e., the size $|\mathcal{S}|$ of each CGS could be very small for each dominant object, thus being insufficient for deriving layout concepts.[2] (2) Layout styles of 3D-Front are limited, e.g., the four variations in Fig. 7 are hard to be derived from 3D-Front. (3) It is also hard to decompose different object groups from rooms, e.g., a large bay may contain both a bed set and a coffee table set, but they need to be closer to separate.

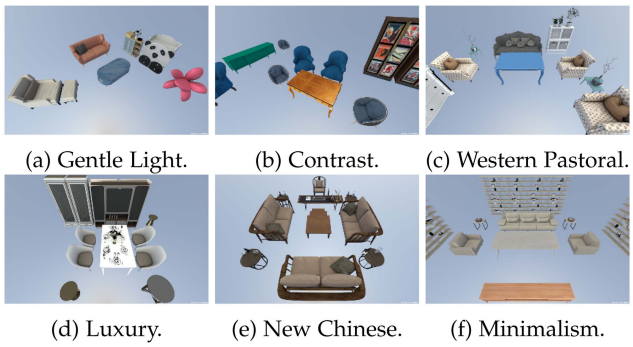
Therefore, we invited 25 professional interior designers to create more layouts as object groups in CGSs to address the above issues, leading to a new dataset focusing on density and layout variations. We use the same system in Fig. 8 for annotating layouts: designers could search, insert, translate, and rotate objects, as shown in Fig. 10. Designers could also edit room shapes if they are too small to hold a coherent group. To ensure style compatibility, we partition CAD models into different categories and require objects in the same CGS to be from a unique style category. Fig. 11 shows representative samples of the dataset. Due to our limited budget, the dataset contains 1,719 layouts. It will be released to the public and expanded in the future. To fully show the generalization and more variations of our interactive framework, we also develop a VR-oriented client to exploit more potentialities of CGS. In immersive virtual environments, no more mouse clicks or cursor is available. Instead, we typically use controllers with hand gestures for interacting with 3D scenes. Existing literature researching 3D scene interactions in VR based on hand gestures has explored one-to-one gesture-command mapping with little

[2]See Section B in the supplementary document, available online, for the statistics.



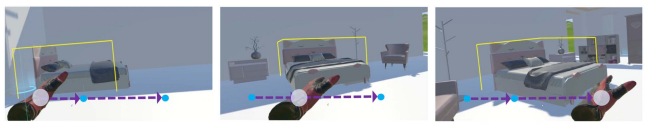
(a) Search and Insert. (b) Transform and Submit.

Fig. 10. The system for annotations and user studies. It supports industrial operations on 3D scenes. (a) designers could search for a style-compatible object and insert it into the scene. (b) the object can be translated, rotated, and re-scaled through three panels: the blue panel using cursors, the local coordinate system, and the value setting panel in the upper left corner. After finishing a coherent group, she/he could submit it with a CGS name (red box).



(a) Gentle Light. (b) Contrast. (c) Western Pastoral.
(d) Luxury. (e) New Chinese. (f) Minimalism.

Fig. 11. Examples in our dataset. Each example has an evolutionary chain similar to Fig. 4. More details are shown in Section B of the supplementary document, available online.



(a) (b) (c)

Fig. 12. Interacting with objects using "Force". Our method allows mouse-free interactions in VR using hand gestures. The time axis shows the movement of the hand. More demonstrations are also shown in the video.

intelligent response by environments, where each interaction involves up to a single object [49], [50], [51], [52]. Using the proposed method, we present a natural hand gesture interaction with multiple objects. In response to users' gestures, our system suggests different layouts for a VR scene in real-time. Fig. 12 illustrates how users create new 3D scenes in VR with our system, where eye-tracking instead of the cursor detects the

dominant object, and the 3D scene is synthesized according to hand gestures instead of the cursor hovering. More technical details can be found in Section D of the supplementary document, available online.

The SceneDirector was implemented in Three.js (Web), Unity (VR) and Flask. Our implementation follows a client-server environment. Our server runs on a Ryzen 2700x machine with 16 GB of RAM. Our method runs in real-time, thus being suitable for nearly all modern devices, including those with integrated graphics cards. For the computing time, extracting layout attributes and boundaries requires less than 10 seconds for a CGS with 15 groups and an object pool capable of deriving up to 10 objects, i.e., a layout with as many objects as possible.

V. Evaluation

To evaluate the utility of SceneDirector, we conduct a user study to measure the system's efficiency, usability, and result quality compared with existing industrial approaches. We recruited 36 subjects to craft indoor scenes using our system interactively. The invited subjects were composed of university students from different majors. All subjects frequently used computers and reported experiences in using tools for 3D modeling or editing office documents. Subjects were paid to participate in our user study. More statistics about the subjects are attached in Section E of the supplementary document, available online, including the age, gender, major, etc. Given an empty room, subjects were asked to generate three satisfied layouts as quickly as possible until they felt satisfied with the crafted layouts. The three satisfied layouts by each participant were crafted using the following three settings.

In Setting Tradition, similar to Kujiale[3] and Planner5D,[4] the system provides typical industrial solutions for manipulating objects, as shown in Fig. 10. Users could search objects with keywords, sketches [53], or styles. Styles of objects are manually classified by several professional interior designers, e.g., new Chinese style or grey modern style. Users could also directly click a frequent keyword in a recommended list. Users could manipulate (translate, rotate, and/or scale) objects by a transform controller, a cursor-based approach that transforms a selected object with the movement of the cursor, or a form that directly accepts values (coordinates), where the object alignment is enabled, e.g., for object surrounding layout. For adding objects, users could click a searched result, and the object would follow the movement of the cursor until object insertion, or they could duplicate existing objects with a single click. We generally implement as many ways to manipulate objects as possible to our best efforts. This is because we cannot guarantee that all the subjects are familiar with a default setup or a usual way. Our practice is to let users choose.

In Setting Clutterpalette, the system provides Clutterpalette [10]. The subjects could alternatively adjust the positions and orientations of objects. They could also optionally search and add more objects if they want. Section F of the supplementary document, available online, shows our platform's implementation details of Clutterpalette.

In Setting SceneDirector, the system provides the proposed method. The subjects could also adjust, search, and add objects.

These three settings were randomly assigned to the participants, with the possible permutations (orders) of the settings evenly distributed. We randomly assigned each subject to a scene type, including bedroom, living room, dining room or living-dining room. We also made sure that room types are evenly distributed across subjects. Note that each subject is assigned with the same room type, shape and size, so the numbers of each room type are the same for all settings. The subjects were shown several well-designed layouts crafted by interior designers as a standard in advance to avoid generating low-quality layouts.

We prepared a manual to guide the participants through the operations, and each participant was taught how to use our system to manipulate objects and use the proposed method. They were free to play with the system until they were comfortable before conducting the study in different settings. All the subjects reported being familiar with the functionalities in less than 15 minutes. During the experiments, the technical staff was available in case of any technical questions.

A. Time Consumption

This section measures how our method saves time consumed by interactive 3D scene synthesis. Table I shows the average time spent on different user interactions, including: (1) navigating the scene where users adjust views to interact with scenes from different perspectives. (2) searching objects where users search favourite objects through keywords, styles and sketches. (3) adding objects where users drag searched results into scenes or duplicate objects. (4) removing objects from scenes. (5) translating objects through the three control panels. (6) rotating objects. (7) rescaling objects. (8) Clutterpalette [10] or the proposed method. (9) other time consumption, including elaborations, considerations, misoperations, etc., which are not trackable by our system. We add timers to every unit operation to ensure that our system correctly records each consumed time, so the systematic errors are negligible. Engineering details of the timers are included in Section G of the supplementary document, available online. Note that we still allow participants to use operations from Setting Tradition to give further preferences to them, and counting on this time, our method still shows significant interactive time savings.

Each cell in Table I contains an average time (in seconds) and a standard deviation in brackets. According to Table I, our method significantly reduces the overall time required to craft a 3D scene. Note that our method is faster due to the efficient and intelligent interactive framework proposed instead of the automation. A Kruskal-Wallis H-Test shows significant statistical differences in the total time between Setting Tradition (or Clutterpalette) and Setting SceneDirector, with the p-values < 0.001. Because our method directly operates on groups of objects, the typical routine for searching and adding objects is much less conducted. Similarly, editing multiple objects saves much more time consumed on operating interactive frameworks

[3][Online]. Available: https://b.kujiale.com/
[4][Online]. Available: https://planner5d.com/

TABLE I
TIME CONSUMPTIONS ON SETTING TRADITION (S1), SETTING CLUTTERPALETTE (S2), AND SETTING SCENEDIRECTOR(S3). EACH CELL INCLUDES AN AVERAGE TIME IN SECONDS AND A STANDARD DEVIATION. THE COLUMN "METHODS" DENOTES CLUTTERPALETTE [10] FOR S2 AND OUR METHOD FOR S3

	Navigating	Searching	Adding	Removing	Translating	Rotating	Rescaling	Methods	Others	Total
S1	24.3 (20.5)	71.6 (99.5)	33.3 (18.2)	0.7 (1.3)	26.3 (20.2)	17.3 (13.7)	6.3 (8.7)	0.0 (0.0)	124.2 (62.8)	309.0 (191.9)
S2	25.5 (21.6)	0.0 (0.0)	6.9 (13.5)	2.2 (3.3)	24.2 (14.6)	15.6 (10.8)	4.4 (5.7)	110.6 (63.6)	93.1 (57.4)	299.0 (152.8)
S3	9.7 (9.0)	14.7 (20.9)	10.4 (6.1)	1.8 (4.7)	11.5 (10.6)	4.7 (5.2)	3.5 (3.8)	32.9 (23.3)	68.1 (43.6)	158.2 (78.8)

TABLE II
USER SATISFACTION

Measurement	Interactive Satisfaction	Result Satisfaction
Tradition	3.1 (0.9)	3.7 (0.9)
Clutterpalette	3.6 (0.7)	3.6 (0.8)
SceneDirector	3.9 (0.9)	3.7 (0.9)

TABLE III
AESTHETIC AND PLAUSIBILITY. THE "AVERAGE" ROW SHOWS THE OVERALL PROPORTION OF THE FOUR SELECTIONS. THE "STANDARD DEVIATION" ROW IS CALCULATED W.R.T. INDIVIDUAL PROPORTIONS OF SUBJECTS

Measurement	Average	Standard Deviation
Tradition	0.283	0.083
Clutterpalette	0.284	0.092
SceneDirector	0.342	0.12
Unsure	0.091	0.108

than Clutterpalette [10]. Although most participants wished to customize their rooms, the time required for transforming (translating, rotating, and rescaling) objects is still improved by our method since groups of objects are plausibly arranged by SceneDirector. Another overwhelmingly convenient feature is it prevents users from "making mistakes", simply because humans can not pay attention all the time, e.g., mis-deleting an object and backtracking this instruction (Ctrl+Z in our system). As claimed that our method gives users precognition during the real-time cursor movements in the scene, it also helps the participants design scenes. To request fewer objects from our system, we observe that some users lowered the "#Objects" weight while others simply manually removed the unwanted objects. Hence, the removing time of SceneDirector is slightly higher.

We observe that the standard deviation values in Table I are relatively large. This is possible because of the following two reasons. First, the room types randomly assigned to users were different, and living-dining rooms were more complicated than the bedrooms in most cases. Therefore, the subjects assigned with the former typically consumed more time on thinking and arranging. Second, their background skills of them also vary. Some subjects reported strong skills in interacting with the virtual world (e.g., video games), and some subjects even majored in arts, but others were only familiar with document editing.

B. User Satisfaction

This section measures how satisfied users are with our system. Table II shows user satisfaction statistics with our method, where each cell includes an average value and a standard deviation in brackets. Once each subject finished with all the settings, we asked them to rate their overall satisfaction with the results. Likert-scale is adopted from 0 "results are inaesthetic and implausible" to 5 "results are very aesthetic and plausible". We also asked them to mark how comfortable and convenient they were during the entire interaction process. The Likert-scale ranges from 0 "the interaction is annoying and inconvenient" to 5 "the interaction is delightful and convenient".

Compared to the industrial solution, our method of simultaneously editing objects was favored by most of the participants according to the "interactive satisfaction", and a Kruskal-Wallis H-Test also shows a significant difference, with a p-value of 0.0026. After interviewing the subjects after their user studies, we found that most enjoyed exploring and foreseeing potential layouts in real-time. As for "result satisfaction", the results of industrial solutions are considered standard and plausible layouts. Since we asked the participants to customize plausible scenes until they were satisfied, the results did not show a big difference. Our method interactively generates layouts faster than the existing solutions, while our method keeps the resulting quality because it gives users more information during crafting scenes.

C. Aesthetic and Plausibility

This section further measures the aesthetics and plausibility of our method. We conducted another user study to measure the results of our method quantitatively. Another 33 participants were invited to evaluate the generated scenes. The newly recruited participants comprised university students, office workers, engineers, designers, etc. They were also paid to participate in the study. More statistics are attached in Section E of the supplementary document, available online, including the age, gender, composition, etc. There was no overlap between this user group and the previous user group. They were only presented with a series of questions without being informed about the previous study. Each question contained three rendered scenes generated from the three settings. The participants were asked to select a favored scene from the presented images, or they could select favoring all of them. For each question, a subject could select up to one favored scene. The questions were shuffled for each questionnaire, and the rendered scenes presented to the subjects in each question were also shuffled. In total, each subject received 30 questions. A snapshot of the online questionnaire for this study is shown in the supplementary document, available online.

Table III shows the results with standard deviations. The average scores are first computed for each participant and then computed over the participants. The standard deviations are also

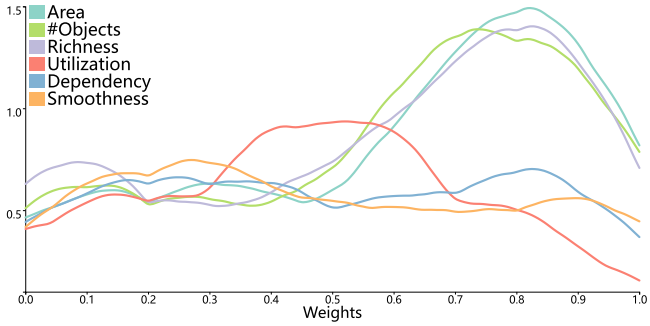


Fig. 13. Distributions of the weights selected by the subjects.

first computed for each participant and then computed over them. Since the scenes generated by our method were favored by nearly one-third of the subjects, our method generated competitive results compared to the industrial solution and the method of editing single objects.

D. Attribute Tuning

As a method of attributing layouts, we need to evaluate the meaningfulness of the proposed attributes to users and whether users with different preferences could benefit from tuning attributes. A meaningful attribute should provide users with different object arrangements by tuning its weight, so we aim to acquire the statistics of the various weights selected by users. We conducted an independent user study to collect how users select a comfortable setting. The participants were the same group from Section V-A. They were initially informed about the meaning of the six attributes. Subsequently, a technical staff majoring in interior design controlled the cursor and showed them the synthesis process of three layouts: a coffee table set, a dining table set, and a double bed set. During each layout adjustment w.r.t. the cursor movements, the technical staff constantly asked questions about the arrangements of objects, e.g., "Do you want the scene to become larger or more compact?", "Do you want the scene to change gradually or variously?", etc. Based on the participants' feedback, the staff iteratively tuned the weights until they captured a favourite arrangement.

The kernel density estimation of the weights is visualized in Fig. 13, where we choose a kernel as $(1 - |(y - y')/\delta|^2)/\delta$ for all $|y - y'| \leq \delta$, where y' denotes the original data points. The bandwidth δ is empirically set as 0.2 since we want to preserve the original distribution instead of a smooth curve. From the curves, the distributions of the weights spread across the x-axis, indicating that all the attributes give rational plays to their roles. For example, to enable a private layout, users typically reduce the weight for dependency. They raised the weight for richness, while some users still wanted the layout concept to include fewer objects, thus reducing the weights of #objects. Some attributes have their own frequently chosen values, such as 0.5 for utilization or 0.8 for the area, which makes sense since the layout attributes are meaningful to users, e.g., most users want the layout concept to expand as large as possible. Hence, they raise the weight of the area.

VI. Discussions and Limitations

The Size of CGSs. Our method can generate layouts according to CGSs, but if a layout strategy does not exist in any CGS, our method cannot generate a result accordingly. In other words, if an interactively generated layout meets the user's preference, the user will consider the result novel. In contrast, the result is no longer novel if the preference is unsatisfied. A more extensive and diverse CGS-based dataset can provide a larger variety and thus alleviate this problem. Additionally, our method requires a finite power set for each original layout in a CGS. Although the original layouts are guaranteed to be included, this may reduce the smoothness and need the attribute "smoothness" to alleviate it.

Breaking Concepts of CGSs. Since each CGS is decided by its object pool and the layouts of these objects, any operation affecting the existence or arrangements of the objects can potentially break the concept contained by the original CGS. To avoid collisions, we simply remove all collided objects in CGSs if the newly inserted objects collide with doors, windows or the objects in the previous groups. Avoiding the collision makes a scene more valid, but this could break the strategy of arranging objects, e.g., if the cabinets are removed in Fig. 7(c) due to a collision w.r.t. a door, the entire layout will be no longer private. Similarly, the layout attributes force a CGS to generate layouts according to specified weights, which could also break the strategy. For collisions, a more innovative strategy is to readjust both involved entities, possibly requiring expressing relations at a "group" level, i.e., learning semantics and functionalities between the former and new groups.

About Sequences of Editing Controlled Objects. Each action edits multiple objects using a CGS. However, if multiple objects belong to two CGSs instead of a single CGS, our method may require the user to edit the objects with two actions. For example, if a CGS contains a TV set and a cabinet, editing the TV set also results in fixing the cabinet. However, if the cabinet is in another CGS, we are limited to fixing the TV set and cabinet using a single action. The user is thus required to fix them with two interactive sessions. As discussed above, the collision strategy will prevent the user from adding new objects, which may introduce even more sessions. Thus, an improvement may also consider merging CGSs.

Unwanted Object Movements. When a user is editing objects, some of them may behave inconsistently with the user's preference. Our method uses the idea of layout attributes to solve this problem, i.e., the user can control the movements of objects by tuning the weights of the attributes. However, through the experiment in Section V-A, there are still a few movements requiring users to fine-tune the layout, which is quantified by the time consumption. For example, a user may delete or re-transform several unwanted objects after editing multiple objects. Ideally, an improvement will further eliminate extra interactions and focus on mere sequences of editing multiple objects.

VII. Conclusion and Future Works

We have proposed a method for simultaneously editing multiple objects with the idea of coherent group set and layout

attributes. Our work is the first interactive scene synthesis system supporting the interactive control of multiple objects. We hope our work will be inspiring for the follow-up works.

To express a specific layout concept, although this paper presents a way to approximate it using 10-20 examples, examples are never sufficient. We should either create a larger set of examples or multiplex existing examples more efficiently and creatively. The former consumes more expenditure. The latter is considered a methodological improvement.

Our contribution is to give users easy control of multiple objects, i.e., giving users quick overviews of various potential layouts. The user preferences are acceptable in two forms in our implementation: user-specified positions and user-specified weights. The former is used for suggesting where to place a layout, and the latter is used for suggesting how to put a layout. We have also explored hand gestures as an alternative to the former. We can adapt existing industrial solutions or Clutterpalette [10] for objects within the group to fine-tune the results. In the future, we are interested in designing more reasonable metrics to give users more options for selecting groups and exploring more potential forms of user inputs to strengthen user preferences in scene synthesis.

Acknowledgments

The authors would like to thank all reviewers for their thoughtful comments.

References

[1] J. D. N. Dionisio, W. G. B. III, and R. Gilbert, "3D virtual worlds and the metaverse: Current status and future possibilities," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 1–38, 2013.

[2] W. Li, J. Talavera, A. G. Samayoa, J.-M. Lien, and L.-F. Yu, "Automatic synthesis of virtual wheelchair training scenarios," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces*, 2020, pp. 539–547.

[3] S.-H. Zhang, C.-H. Chen, Z. Fu, Y. Yang, and S.-M. Hu, "Adaptive optimization algorithm for resetting techniques in obstacle-ridden environments," *IEEE Trans. Visual. Comput. Graph.*, vol. 29, no. 4, pp. 2080–2092, Apr. 2023.

[4] S.-H. Zhang, Z.-P. Zhou, B. Liu, X. Dong, and P. Hall, "What and where: A context-based recommendation system for object insertion," *Comput. Vis. Media*, vol. 6, no. 1, pp. 79–93, 2020.

[5] F. D. Ching and C. Binggeli, *Interior Design Illustrated*. Hoboken, NJ, USA: Wiley, 2018.

[6] 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 Trans. Graph.*, vol. 30, no. 4, 2011, Art. no. 86.

[7] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan, "Example-based synthesis of 3D object arrangements," *ACM Trans. Graph.*, vol. 31, no. 6, 2012, Art. no. 135.

[8] S. Qi, Y. Zhu, S. Huang, C. Jiang, and S.-C. Zhu, "Human-centric indoor scene synthesis using stochastic grammar," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5899–5908.

[9] K. Wang, Y.-A. Lin, B. Weissmann, M. Savva, A. X. Chang, and D. Ritchie, "PlanIT: Planning and instantiating indoor scenes with relation graph and spatial prior networks," *ACM Trans. Graph.*, vol. 38, no. 4, 2019, Art. no. 132.

[10] L.-F. Yu, S.-K. Yeung, and D. Terzopoulos, "The clutterpalette: An interactive tool for detailing indoor scenes," *IEEE Trans. Visual. Comput. Graph.*, vol. 22, no. 2, pp. 1138–1148, Feb. 2016.

[11] S. Zhang, Z. Han, Y.-K. Lai, M. Zwicker, and H. Zhang, "Active arrangement of small objects in 3D indoor scenes," *IEEE Trans. Visual. Comput. Graph.*, vol. 27, no. 4, pp. 2250–2264, Apr. 2021.

[12] H.-L. Lu, *Residential Interior Design*. Shenyang, China: Liaoning Science and Technology Publishing House, 2010.

[13] T.-K. Zheng, *Interior Design For Home*. Wuhan, China: Huazhong University of Science & Technology Press, 2011.

[14] A. B. Vasiliki Asaroglou, *Furniture Arrangement: In Residential Spaces*. Scotts Valley, CA, USA: CreateSpace Independent Publishing Platform, 2013.

[15] S. Zhang, Z. Han, and H. Zhang, "User guided 3D scene enrichment," in *Proc. 15th ACM SIGGRAPH Conf. Virtual-Reality Continuum Appl. Ind.*, 2016, pp. 353–362.

[16] M. Savva, A. X. Chang, and M. Agrawala, "SceneSuggest: Context-driven 3D scene design," 2017, *arXiv: 1703.00061*.

[17] S.-K. Zhang, Y.-X. Li, Y. He, Y.-L. Yang, and S.-H. Zhang, "MageAdd: Real-time interaction simulation for scene synthesis," in *Proc. 29th ACM Int. Conf. Multimedia*, 2021, pp. 965–973.

[18] M. Mitton and C. Nystuen, *Residential Interior Design: A Guide to Planning Spaces*. Hoboken, NJ, USA: Wiley, 2016.

[19] K. Wang, M. Savva, A. X. Chang, and D. Ritchie, "Deep convolutional priors for indoor scene synthesis," *ACM Trans. Graph.*, vol. 37, no. 4, 2018, Art. no. 70.

[20] kujiale.com, "Kujiale," Dec. 2020. [Online]. Available: https://www.kujiale.com/

[21] planner5d.com, "Planner5D," Dec. 2020. [Online]. Available: https://planner5d.com/

[22] Y.-G. Peng, *Architectural Space Combination Theory*. Beijing, China: China Architecture & Building Press, 2008.

[23] 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 Trans. Graph.*, vol. 36, no. 6, pp. 1–13, 2017.

[24] Z. Zhang et al., "Deep generative modeling for scene synthesis via hybrid representations," *ACM Trans. Graph.*, vol. 39, no. 2, pp. 1–21, 2020.

[25] 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 Trans. Graph.*, vol. 31, no. 4, 2012, Art. no. 56.

[26] Y. Liang, S.-H. Zhang, and R. R. Martin, "Automatic data-driven room design generation," in *Proc. Int. Workshop Next Gener. Comput. Animation Techn.*, Springer, 2017, pp. 133–148.

[27] T. Weiss et al., "Fast and scalable position-based layout synthesis," *IEEE Trans. Visual. Comput. Graph.*, vol. 25, no. 12, pp. 3231–3243, Dec. 2019.

[28] 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 Trans. Visual. Comput. Graph.*, vol. 28, no. 9, pp. 3082–3092, Sep. 2022.

[29] M. Li et al., "GRAINS: Generative recursive autoencoders for indoor scenes," *ACM Trans. Graph.*, vol. 38, no. 2, pp. 1–16, 2019.

[30] D. Ritchie, K. Wang, and Y.-A. Lin, "Fast and flexible indoor scene synthesis via deep convolutional generative models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6182–6190.

[31] S.-K. Zhang, W.-Y. Xie, and S.-H. Zhang, "Geometry-based layout generation with hyper-relations among objects," *Graphical Models*, vol. 116, 2021, Art. no. 101104.

[32] Y. He, Y. Liu, Y. Jin, S.-H. Zhang, Y.-K. Lai, and S.-M. Hu, "Context-consistent generation of indoor virtual environments based on geometry constraints," *IEEE Trans. Visual. Comput. Graph.*, vol. 28, no. 12, pp. 3986–3999, Dec. 2022.

[33] K. Xu, K. Chen, H. Fu, W.-L. Sun, and S.-M. Hu, "Sketch2Scene: Sketch-based co-retrieval and co-placement of 3D models," *ACM Trans. Graph.*, vol. 32, no. 4, 2013, Art. no. 123.

[34] A. Chang, W. Monroe, M. Savva, C. Potts, and C. D. Manning, "Text to 3D scene generation with rich lexical grounding," 2015, *arXiv:1505.06289*.

[35] R. Ma et al., "Language-driven synthesis of 3D scenes from scene databases," in *Proc. SIGGRAPH Asia Tech. Papers*, 2018, Art. no. 212.

[36] K. Chen, Y. Lai, Y.-X. Wu, R. R. Martin, and S.-M. Hu, "Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information," *ACM Trans. Graph.*, vol. 33, no. 6, 2014, Art. no. 208.

[37] M. Fisher, M. Savva, Y. Li, P. Hanrahan, and M. Nießner, "Activity-centric scene synthesis for functional 3D scene modeling," *ACM Trans. Graph.*, vol. 34, no. 6, 2015, Art. no. 179.

[38] K. Chen, Y.-K. Lai, and S.-M. Hu, "3D indoor scene modeling from RGB-D data: A survey," *Comput. Vis. Media*, vol. 1, no. 4, pp. 267–278, 2015.

[39] A. Luo, Z. Zhang, J. Wu, and J. B. Tenenbaum, "End-to-end optimization of scene layout," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3754–3763.

[40] G. Xiong, Q. Fu, H. Fu, B. Zhou, G. Luo, and Z. Deng, "Motion planning for convertible indoor scene layout design," *IEEE Trans. Visual. Comput. Graph.*, vol. 27, no. 12, pp. 4413–4424, Dec. 2021.

[41] P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun, "Interactive furniture layout using interior design guidelines," *ACM Trans. Graph.*, vol. 30, no. 4, 2011, Art. no. 87.

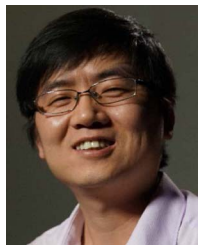
[42] M. Yan, X. Chen, and J. Zhou, "An interactive system for efficient 3D furniture arrangement," in *Proc. Comput. Graph. Int. Conf.*, 2017, pp. 1–6.
[43] W. Liang, J. Liu, Y. Lang, B. Ning, and L.-F. Yu, "Functional workspace optimization via learning personal preferences from virtual experiences," *IEEE Trans. Visual. Comput. Graph.*, vol. 25, no. 5, pp. 1836–1845, May 2019.
[44] Y. Zhang, H. Huang, E. Plaku, and L.-F. Yu, "Joint computational design of workspaces and workplans," *ACM Trans. Graph.*, vol. 40, no. 6, pp. 1–16, 2021.
[45] Y. Li, J. Zhang, Y. Cheng, K. Huang, and T. Tan, "DF 2 net: Discriminative feature learning and fusion network for RGB-D indoor scene classification," in *Proc. AAAI Conf. Artif. Intell.*, 2018, Art. no. 862.
[46] M. Fisher, M. Savva, and P. Hanrahan, "Characterizing structural relationships in scenes using graph kernels," in *Proc. ACM SIGGRAPH Papers*, 2011, pp. 1–12.
[47] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt, "Graph kernels," *J. Mach. Learn. Res.*, vol. 11, pp. 1201–1242, 2010.
[48] H. Fu et al., "3D-front: 3D furnished rooms with layouts and semantics," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 10 933–10 942.
[49] H. Kim, G. Albuquerque, S. Havemann, and D. W. Fellner, "Tangible 3D: Hand gesture interaction for immersive 3D modeling," in *Proc. 11th Eurograph. Conf. Virtual Environ.*, 2005, pp. 191–199.
[50] J. Kela et al., "Accelerometer-based gesture control for a design environment," *Pers. Ubiquitous Comput.*, vol. 10, no. 5, pp. 285–299, 2006.
[51] N. H. Dardas and M. Alhaj, "Hand gesture interaction with a 3D virtual environment," *Res. Bull. Jordan*, vol. 2, no. 3, pp. 86–94, 2011.
[52] Y. Li, X. Wang, Z. Wu, G. Li, S. Liu, and M. Zhou, "Flexible indoor scene synthesis based on multi-object particle swarm intelligence optimization and user intentions with 3D gesture," *Comput. Graph.*, vol. 93, pp. 1–12, 2020.
[53] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 945–953.

Shao-Kui Zhang received the BS degree of software engineering from Northeastern University, Shenyang, in 2018. He is currently working toward the PhD degree with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include computer graphics, 3D scene synthesis and intelligent 3D scene interaction.

Hou Tam received the bachelor's degree in computer science and technology from Tsinghua University, in 2021. He is currently working toward the master's degree with the Department of Computer Science and Technology, Tsinghua University.

Yike Li received the bachelor's degree in computer science from the Wuhan University of Science and Technology, Wuhan, in 2020. She is currently working toward the master's degree with the Academy of Arts & Design, Tsinghua University.

Ke-Xin Ren received the bachelor's degree in landscape architecture from the China Central Academy of Fine Art, Beijing, in 2020. She is currently working toward the master's degree with the Academy of Arts & Design,Tsinghua University.



Hongbo Fu received the BS degree in information sciences from Peking University, and the PhD degree in computer science from the Hong Kong University of Science and Technology. He is a professor with the School of Creative Media, City University of Hong Kong. He has served as an associate editor of *The Visual Computer*, *Computers & Graphics*, and *Computer Graphics Forum*. His primary research interests include computer graphics and human computer interaction.

Song-Hai Zhang (Member, IEEE) received the PhD degree of computer science and technology from Tsinghua University, Beijing, in 2007. He is currently an Associate Professor with the Department of Computer Science and Technology, Tsinghua University. His research interests include computer graphics and virtual reality.