Motion Planning for Convertible Indoor Scene Layout Design

Guoming Xiong*, Qiang Fu*, Hongbo Fu, Bin Zhou, Guoliang Luo, Zhigang Deng+

Abstract—We present a system for designing indoor scenes with convertible furniture layouts. Such layouts are useful for scenarios where an indoor scene has multiple purposes and requires layout conversion, such as merging multiple small furniture objects into a larger one or changing the locus of the furniture. We aim at planning the motion for the convertible layouts of a scene with the most efficient conversion process. To achieve this, our system first establishes object-level correspondences between the layout of a given source and that of a reference to compute a target layout, where the objects are re-arranged in the source layout with respect to the reference layout. After that, our system initializes the movement paths of objects between the source and target layouts based on various mechanical constraints. A joint space-time optimization is then performed to program a control stream of object translations, rotations, and stops, under which the movements of all objects are efficient and the potential object collisions are avoided. We demonstrate the effectiveness of our system through various design examples of multi-purpose, indoor scenes with convertible layouts.

Index Terms—Indoor scene synthesis, motion planning, convertible layout.

1 Introduction

Furniture layout synthesis has been extensively studied in the past decades. On one hand, recent related efforts have been mainly focused on the creation of plausible stationary layouts by optimizing the positions and orientations of objects with various layout constraints. On the other hand, there widely exist indoor scenes with multiple layouts, which need to be transformed frequently for different functional purposes. For example, as shown in Figure 1, the seats in a high-speed train need to be frequently adjusted to ensure that they have the same directions as the moving direction of the train. The cabinets in a library often have a movable design to maximize the use of the limited space. Using movable walls can generate various spatial divisions for a large room. The composability of objects can be both in- and cross-class such as modular furniture. Some of such multi-purpose indoor scenes are especially designed for small rooms. For instance, we might need to change a layout of our living room to hold a party, or re-arrange desks and chairs in a classroom from a lecture setting to a group-discussion setting.

To cope with such a demand, a well-designed indoor scene with convertible layouts should have an effective layout conversion process. The convertible layouts can be specified either by using the same set of objects but with different object arrangements, or by using a reference layout to guide the movement of the objects in a source layout to a new arrangement. We focus on the second case and refer to the new arrangement as a target layout. It is expected the objects in the reference layout should be similar to those in the source layout, in terms of both functionality and configuration. The convertible layouts can be designed by following certain mechanical constraints (such as rotating shafts and lead rails) to make the conversion physically executable. The automatic design of such indoor scenes, however, remains challenging due to the following main reasons.

- First, since the numbers of objects in the source and reference layouts might not always be the same, the correspondences between the two layouts, which can be one-to-one or one-to-many mappings from the reference to the source, should be established to determine the starting/ending positions and orienta-

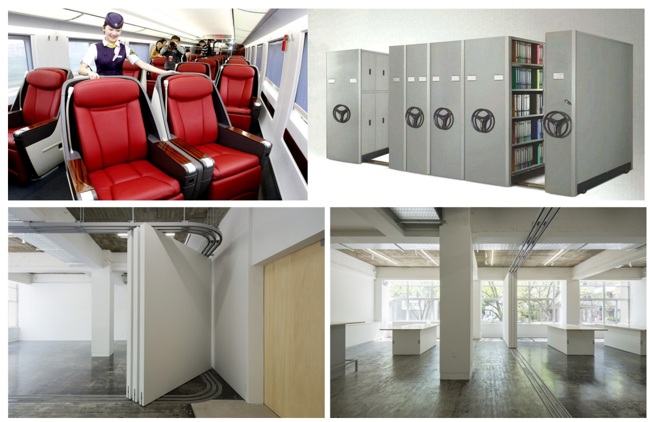


Fig. 1. **Top:** Examples of convertible layouts in the real world. **Bottom:** Movable walls along lead rails can divide a large room into smaller ones.

- *Joint first authors. + Corresponding author (zdeng4@uh.edu)
- G. Xiong and G. Luo are with Virtual Reality and Interactive Techniques Institute, East China Jiaotong University, China.
- Q. Fu, and Z. Deng are with the Department of Computer Science, University of Houston, USA.
- Q. Fu is also with the School of Digital Media and Design Arts, Beijing University of Posts and Telecommunications, China.
- H. Fu is with the School of Creative Media, City University of Hong Kong.
- B. Zhou is with the School of Computer Science, Beihang University, China.

tions of all the objects in the convertible layouts.

- Second, the movements of objects in convertible layouts are typically not free-form but controlled by certain mechanical constraints. A metric to evaluate the choice of motion mechanisms is needed to explore a low-cost mechanical system satisfying the layout conversion process.
- Lastly, since the movements of the objects are spacetime-related during the layout conversion, a control stream that performs the movements of the associated motion mechanisms should be determined to avoid object collisions and improve the conversion efficiency.

Aiming at efficiently executing the layout conversion with low-cost motion mechanisms, we need to perform a joint space-time optimization to program the movement sequence of all objects in the layout conversion process. The configurations of the associated motion mechanisms, including the connection positions, the types of rotating shafts, and the starting/ending positions of lead rails, should be optimized as well.

With the above motivation and insight, we present a novel framework to design indoor scenes with convertible layouts and well-planned motions, as illustrated in Figure 2. Given a source layout (typically with more objects) and a reference layout, our system first establishes object-level correspondences between the two layouts. Then we calculate the positions and orientations of the source objects guided by the associated reference objects to generate a target layout. Next, the initial paths of all movable objects and the connection positions of the associated motion mechanisms are determined based on the guidelines derived from mechanical motion constraints. Afterwards, a joint space-time optimization is performed to fine-tune the control stream of all movable objects to address the potential object-object collisions during the layout conversion and make the conversion process efficient. We show various results and potential applications to demonstrate the applicability and effectiveness of our system for the synthesis of indoor scenes with convertible layouts.

Our work has the following main contributions: 1) a unified framework to design convertible layouts for indoor scene synthesis; 2) the application of motion mechanisms as constraints in the designed convertible layout to make it executable; and 3) the introduction of a joint space-time optimization algorithm to compute a control stream of the movements and stops of objects for the layout conversion process.

2 Related Work

In this section, we first briefly review the recent related works on indoor scene synthesis and then discuss various existing research efforts on the analysis of motion mechanisms from the perspective of computer graphics, and its applications to the multi-purpose design of both man-made objects and indoor scenes.

Indoor Scene Synthesis. In the past decades, many approaches have been proposed to synthesize indoor scenes with plausible layouts. Benefiting from existing efforts that explore the structure or object relationship of indoor scenes [1], [2], [3], an increasing number of digital 3D models can be obtained for indoor scene creation. To facilitate indoor scene creation, some existing approaches have focused on the arrangement of small objects to enrich the details of created indoor scenes [4], [5], [6], while the others are more concerned about the layout generation of furniture-level objects [7], [8]. Various interior design knowledge, insight, and priors, such as pre-defined guidelines (e.g., [9], [10]), indoor scene examples (e.g., [11], [12], [13]), human-object relationships (e.g., [14], [15], [16]), and deep network priors (e.g., [17], [18]) have been studied to generate a static arrangement of 3D furniture models according to their aesthetic or functional features. However, the above methods are less concerned about the feasibility of possible transitions between different layouts.

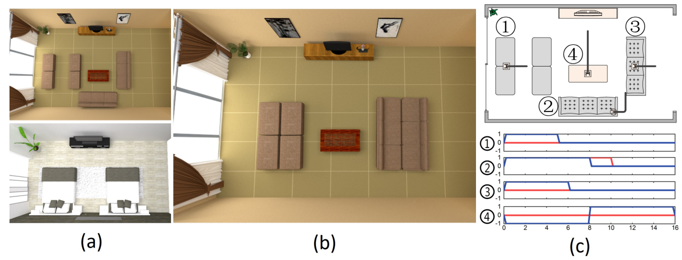


Fig. 2. Given a source layout ((a)-top) and a reference layout ((a)-bottom), our system generates a target layout (b) and designs a workable conversion process driven by motion mechanisms for translation and orientation ((c)-top) and the programming of an efficient control stream for such a conversion ((c)-bottom), to make the source and target layouts convertible.

Motion Mechanisms. Revealing the motion mechanisms of man-made objects, such as the degrees of freedom (DOFs) of their movable parts, has attracted much attention in the computer graphics and computer-aided design communities. For example, Mitra et al. proposed an approach for mechanical assembly visualization that incorporates motion arrows, frame sequences, and animations to convey the causal chain of motions and mechanical interactions between parts [19]. Hu et al. presented a method to learn a model for the mobility of parts in 3D objects [20]. These efforts have inspired the design of man-made objects with movable parts or motion mechanisms. For example, researchers proposed to model objects with interactive mechanisms from multi-view images or scanning data [21], [22], or to model works-like prototypes, namely, objects with interactive parts such as movable cabinet doors and drawers [23], [24]. In our work, we utilize motion mechanisms as the constraints to ensure workable convertible layout designs. Aiming at such a goal, we do not intend to model precise motion mechanisms or their relations in a convertible layout system. Instead, we focus on some common types of motion mechanisms including shafts and lead rails, and apply their mechanical constraints to optimize the programming of both the path and motion in the layout conversion process.

In addition, with similar motivations to ours, some previous works on robotics and automatic control have studied issues of indoor navigation [25], [26] and multi-

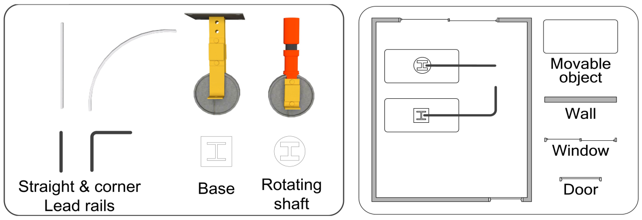


Fig. 3. **Left**: Motion mechanisms and their symbols. **Right**: Symbols that represent indoor scenes and movable objects driven by the associated motion mechanisms.

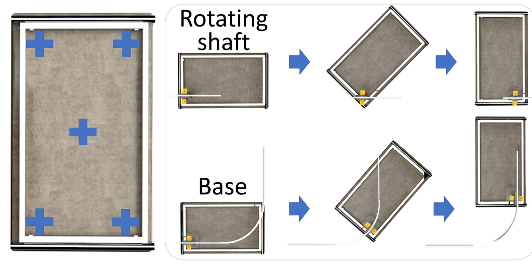


Fig. 4. **Left**: Pre-defined positions (blue crosses) to install a base or rotating shaft on a movable object. **Right**: Examples of the movements driven by a rotating shaft and a base passing a corner.

agent systems [27], [28], [29]. These methods typically focus on free movement, i.e., without mechanical constraints to limit the DOFs of movement, while our work pays more attention to motion planning constrained by motion mechanisms to ensure designed convertible layouts feasible at a low cost.

Multi-purpose Design. Designing man-made objects or indoor scenes with multiple functionalities or for multiple purposes has caught increasing attention in recent years. Assembling parts from cross-category man-made objects is one of the potential solutions to the design of multi-purpose objects. Fu et al. presented a system that leverages human poses to inspire the assembling of parts from different object categories [30]. Hu et al. use functionality models for functional hybrid creation [31]. Besides, some approaches were also proposed to program the way to divide objects into assembled pieces, which can be used to design knock-down toys and furniture [32], [33]. The divided pieces are then analyzed to be assembled into different configurations or kinds of objects for meeting the need of multi-purpose design [34], [35].

Yet designing multi-purpose indoor scenes remains challenging. Although existing methods for designing objects with functionalities such as stackabilization and foldability [36], [37] offer some inspirations for designing scenes with convertible layouts, it is still not clear how to automatically design convertible layouts with motion mechanisms. Garg et al. presented an interactive system for computational design of a reconfigurable entity, i.e., an object or a collection of objects, whose transformations between various states define its functionality or aesthetic appearance [38]. Their motivation is similar to ours. The main difference is that, their system assists users to manually refine the transformation process by highlighting certain periods on the timeline when collisions happen. In other words, they rely on users to resolve object collisions. In contrast, our system takes an automatic approach by automatically programming the layout conversion process (including both transformation paths and motions). This further improves the efficiency, especially in some scenarios, where more than one pair of objects might collide with each other. Besides, our system also considers motion mechanisms and their constraints in path and motion planning, to make generated convertible layouts more physically executable than pure computer animations.

3 Mechanical Constraints and Guidelines

In this section we first introduce the motion mechanisms that we utilize to construct a mechanical system for layout conversion. Then we summarize the guidelines regarding how to design an efficient system with such motion mechanisms.

We focus on three kinds of fundamental motion mechanisms, that is, lead rails, bases, and rotating shafts, as illustrated in Figure 3. A lead rail controls the translation movement of indoor objects. In our implementation, each object is set on a lead rail through a base or a rotating shaft. We specify five potential positions at the bottom of a movable object, as illustrated in Figure 4 (Left). The difference between a base and a rotating shaft is that a rotating shaft can rotate an associated object freely in any position of a rail, while a base can only drive an associated object to translate along a rail. To make the designed motion mechanisms assemblable, we employ modular lead rails. Namely, the component of a lead rail need to be either a straight unit or a right angle corner units, and the lead rail in a room need to be either horizontal or vertical for less damage caused by its installation on the floor. Note that the base can also turn the orientation of an associated movable object through translations along a corner rail (e.g., Figure 4). For simplicity, we assume that movable indoor objects are either light enough or have omni-directional wheels so that they can be fluently translated or rotated by their installed motion mechanisms. We also assume that an object driven by its associated mechanisms can translate and rotate in a uniform speed.

In this work, a layout conversion system is denoted as $\mathcal{S}$, which consists of several motion mechanisms. Let n_s, n_c, n_r be the numbers of straight rail units, corner rail units, and rotating shafts, respectively. The time cost of an object o during layout conversion includes the movement time of o, and the waiting time of o in which it stops and waits for the movement of other objects. Since the translation and rotation of o are synchronous, according to the characteristics of the chosen mechanisms, we use the maximum time to define the time cost as $T_o = \max(T_t + T_w^t, T_r + T_w^r)$, where T_t, T_r, T_w^t and T_w^r are the required steps for translating, rotating, waiting for translation, and waiting for rotation, respectively. In this manner, we summarize a twofold guideline to ensure the motion mechanisms in a convertible layout system to be as simple as possible, and the conversion process to be efficient as well. For the first purpose, we expect the system to have as few mechanisms as possible and define a complexity cost to evaluate the designed mechanical system as follows:

$$C(\mathcal{S}) = n_s + \omega_1 \cdot n_c + \omega_2 \cdot n_r, \quad (1)$$

where weights ω_1 and ω_2 are set to 5 and 10 in our implementation, respectively. We use larger weights for corner units and rotating shafts, since they are more expensive to produce than straight units due to the arc shape of corner units and extra motors for rotating shafts. For the second purpose, we use $\max(T_o), \forall o$, as the total time cost of the system $\mathcal{S}$. Such a cost is utilized to guide the movement sequence adjustment, aiming at an efficient layout conversion process without object collisions.

4 Our Approach

Our approach consists of three phases to program a simple and efficient layout conversion process. First, we establish the correspondences of the objects between the given source and reference layouts to generate a target layout, which has the same set of objects in the source layout and a similar object arrangement with the reference layout. Second, we discretize the timeline of movement as the initial control stream (including only translations and rotations) and explore initial paths for all the objects that can be moved from the source layout to the target layout. Finally, our approach performs a joint space-time optimization that fine-tunes both the control stream (including stops) and paths in order to avoid potential object collisions in the layout conversion process.

4.1 Establishing Object Correspondences

We represent an indoor layout by a 2D floor plan, with the information including the positions, orientations, and sizes of all associated indoor objects. Without loss of generality, we assume that 1) the given source and reference layouts have similar room sizes; 2) the source layout could have more objects than the reference layout; 3) the rooms of all layouts have real-world scales. We go through all pairs of objects between the two layouts, and establish the object correspondences through their similarities in both position and size. The rationale of our algorithm is that shorter paths in layout conversion more likely lead to lower time and complexity cost, and similar sizes are more likely to facilitate the exchange between objects. Note that, as the source layout could have more objects than the reference layout, such a correspondence can be either a non-injective or a non-surjective mapping. In other words, one object in the reference layout can be matched with multiple objects or none in the source layout.

Let $\mathcal{O}$ and $\tilde{\mathcal{O}}$ be two sets of objects in the given source and reference layouts, respectively, $\{\mathbf{p}_i\}$ be the positions of the source objects, and $\{\tilde{\mathbf{p}}_j\}$ be similarly defined for the reference objects. The areas of the object projections from the top view in the source and reference layouts are denoted as $\{s_i\}$ and $\{\tilde{s}_j\}$, respectively. To establish object correspondences, we define the distance between a pair of objects $o_i \in \mathcal{O}$ and $\tilde{o_j} \in \tilde{\mathcal{O}}$ as follows:

$$D(o_i, \tilde{o}_j) = ||\mathbf{p}_i - \tilde{\mathbf{p}}_j||_2^2 + \omega * |s_i - \tilde{s}_j|, \quad (2)$$

where the weight ω, set to 0.01 in our experiments, is to balance their scales. We calculate such a distance for each pair of objects in the two given layouts. The object correspondence is established in an order from the largest object in $\tilde{\mathcal{O}}$ to the smallest one in $\tilde{\mathcal{O}}$. For an object $\tilde{o}_j \in \tilde{\mathcal{O}}$, since $\tilde{o}_j$ might be related to more than one object in $\mathcal{O}$, we expect that all the related objects should have small distances (Equation 2) to $\tilde{o}_j$ and can be merged into a larger one with a similar size as $\tilde{o}_j$. To this end, we first sort the objects in $\mathcal{O}$ based on their distances to $\tilde{o}_j$ from small to large, and choose the top-K ($K \geq 1$) sorted objects such that $|\tilde{s}_j - S(\bigcup_{k=1,\cdots,K} o_k)| < |\tilde{s}_j - S(\bigcup_{k=1,\cdots,K+1} o_k)|$ as the *related objects* to $\tilde{o}_j$, where $S(\bigcup_{k=1,\cdots,K} o_k)$ is the area of the object union $\{o_1, \cdots, o_K\}$.

If there is only a single source object related to $\tilde{o}_j$ (i.e., $K = 1$), we determine its position and orientation in the target layout by directly transferring the position and orientation of $\tilde{o}_j$ to the related source object. When $K \geq 2$, the positions and orientations of the source objects related to $\tilde{o}_j$ in the target layout become less determined. Since this is essentially a packing problem, we apply commonly used rules to guide the packing optimization: the transformed source objects, denoted as $\mathcal{O}'_j$, should cover $\tilde{o}_j$ as much as possible, while the intersection between the transformed source objects should be minimized. This can be mathematically formulated as:

$$\zeta(\mathcal{O}'_j, \tilde{o}_j) = \begin{cases} \infty, \exists o_k, o_{k'} \in \mathcal{O}'_j, \delta(o_k, o_{k'}) > 0 \\ -\sum_{o_k \in \mathcal{O}'_j} \delta(o_k, \tilde{o}_j), \text{otherwise} \end{cases}, \quad (3)$$

where $\delta(o_i, o_j)$ is the overlapping area of objects o_i and o_j with certain poses, i.e., positions and orientations. Hence the poses of objects $\{o_k\}$ are resolved by solving $\arg\min \zeta(\mathcal{O}', \tilde{o}_j)$.

To solve the above minimization problem (i.e., the unknown positions and orientations of the source objects related to $\tilde{o}_j$), we enumerate all possible positions and orientations of these source objects by sweeping them over $\tilde{o}_j$ and rotating them by 90 degree each time (we only focus on four axis-aligned orientations in this work). Note that there might exist certain source objects that are not related to any reference object. We keep them unchanged or let users adjust their positions and orientations in the target layout. A new layout (i.e., the target layout) can be obtained after all the source objects are processed in this way. Since automatic establishment of object correspondences might lead to an undesired target layout, we provide an interface to allow users to manually refine the target layout by adjusting the positions and/or orientations of certain objects.

4.2 Initial Programming

The purpose of initial programming is to suggest feasible paths of objects for layout conversion as well as their possible movements, based on the aforementioned mechanical constraints (Section 3).

Inspired by the work of Garg et al. [38], we also utilize the timeline of each object to handle the movement. The main difference between their work and ours is that we discretize the timeline of each object into slices so that our system can edit the tag of each slice (i.e., stay, translate, or rotate) in the timeline to control its movement. For

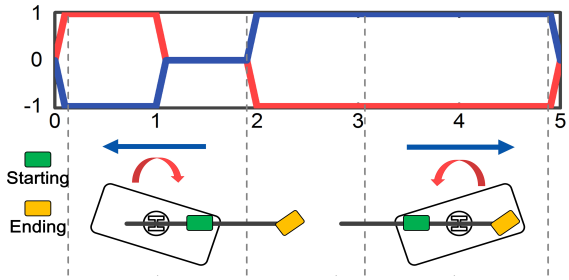


Fig. 5. An example of control stream in the timeline (Top), and the motion of the associated object in two sample time slices (Bottom). We illustrate the position and pose of the object on the lead rail, where the starting/ending positions and poses are shown by the green/yellow rectangles, respectively. Note how the values of translation and rotation in the control stream impact the object movement.

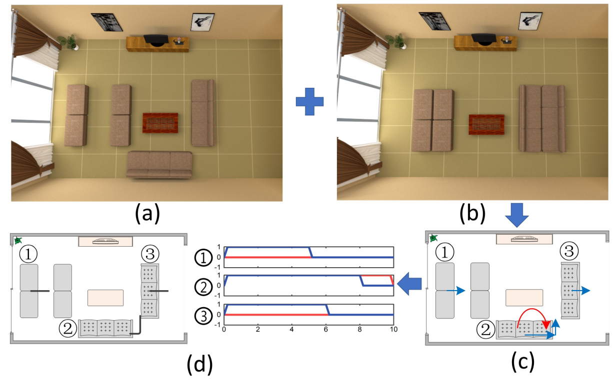


Fig. 6. The positions and orientations of the objects in the source layout (a) and the target layout (b) are utilized to determine their motions (c). We then obtain the initial paths and movements (d) based on the mechanical constraints.

example in Figure 5, each slice consists of two ternary values (indicating front/back motions and stop, respectively) representing the control streams for translation and rotation, denoted as $T(o_i, t)$ and $R(o_i, t)$, respectively. $T(o_i, t) = 1$ and $T(o_i, t) = -1$ respectively represent the object o_i translating to and away from the ending position at time t. $T(o_i, t) = 0$ means waiting for translation. Similarly, $R(o_i, t) = 1$ and $R(o_i, t) = -1$ respectively represent the object o_i rotating clockwise and counterclockwise at time t, and $R(o_i, t) = 0$ means waiting for rotation. $T(o_i, t)$ and $R(o_i, t)$ are illustrated in blue and red, respectively in the timeline of Figure 5. We assume that an object moves uniformly along a lead rail. This uniform speed assumption would ensure our design results executable in real-world scenarios, where motors driving the object movement often have uniform speeds. Specifically, in our implementation, the translation speed on the lead rail is set to 1 unit step (i.e., the length of a straight rail unit) per time step, while it takes 4 time steps for an object to pass a corner rail unit, and the rotation speed for rotating shaft is 18-degree per time step.

Figure 6 shows the workflow of path initialization. After the object correspondence is established, the starting and ending positions/directions of all movable objects are known from the source and target layouts. Based on these known states, we can compute the basic translation direction and rotation angle for each object. Note that since the rotation can be performed via either a corner rail or a rotating shaft, the control stream on translation and rotation for each object can only be determined after the initial path is programmed. Then we intend to search for an initial path for each object, following the complexity cost in Equation 1 that forces the path to have as few mechanisms as possible and as short as possible, since the path distance also depends on the numbers of lead rail units. We also expect each object to have no collision with the room and the minimal intersection with other objects in the course of its movement along the path. In our implementation, we use the collision detector of Unity3D for fast object collision detection.

To achieve this, we design a search algorithm, which performs the following operations in order. 1) If a line segment connecting the starting and ending positions is horizontal or vertical, we choose it as an initial path unless certain obstacles exist on it. 2) If not, as illustrated in Figure 7 (a), we have two right-angle sides (green and blue), and choose the one where the object would not collide with the room and has fewer collisions to the other objects in the source layout as an initial path. Note that if the object needs to perform a 180-degree rotation without any rotating shaft, translation on the rail with only one corner (supporting 90-degree rotation only) is not enough, as illustrated in Figure 7 (d). This can be addressed by adding a corner rail at the beginning or end of the right-angle side (Figure 7 (e)). 3) If both right-angle sides result in object-object or object-room collisions, we first insert a point (in red) in the middle of the line segment between the starting and ending positions (Figure 7 (b)) to create paths with two corners (green and blue). We then move the inserted midpoint to the two ends (Figure 7 (c)) until a certain path does not have any object-room collisions. This path is then chosen as the initial path. If all the above operations failed, user intervention is required to adjust the target layout or directly design the initial path for a certain object. Besides, we also choose the connection positions (deciding the starting and ending positions) of the bases or rotating shafts on the associated objects. Specifically, among the five possible positions (Figure 4), we choose the one that allows the installation of motion mechanisms and would lead to fewest collisions during the search process. In this manner, the initial path can be obtained with the minimum complexity cost, while avoiding potential collisions with obstacles. For example, in Figure 8, our algorithm can produce different initial paths for different settings of obstacles.

4.3 Conversion Fine-tuning

At the final step our approach fine-tunes the layout conversion in spacetime by programming the initial control stream, and then determines the exact motion mechanism configurations. In the first task, we adjust the discretized control stream in order to avoid object collisions during layout conversion, and extend the initial paths to enable the execution of the programmed control stream if needed. In the second task, we mainly determine the types of mechanisms (i.e., a base or rotating shaft), after the motions of all objects are programmed. Besides, we also discuss how to extend the proposed control stream programming

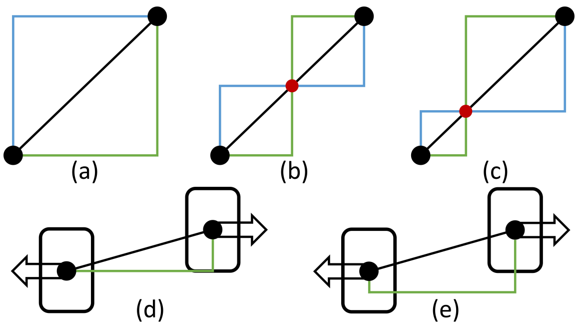


Fig. 7. **Top:** Potential paths (highlighted in blue and green) between the starting and ending positions (black points), and the inserted midpoint (red). **Bottom:** An extra corner rail should be added (from (d) to (e)) for 180-degree rotation. Each arrow indicates the direction of an associated object.

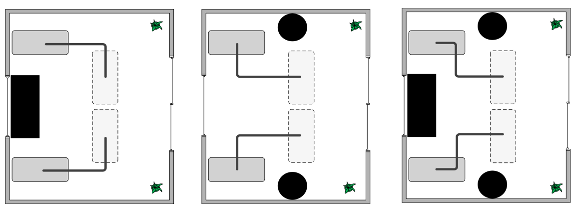


Fig. 8. Examples of initial path searching results in three scenarios of obstacles (black).

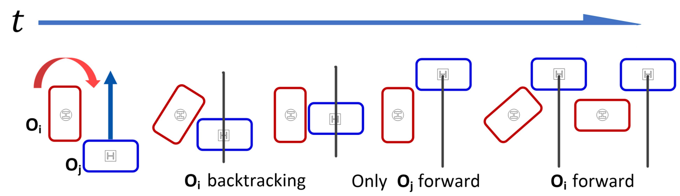


Fig. 9. An example to illustrate the forward and backtracking manipulations when object collision happens between a rotating object (red) and a translating object (blue). In this example, the object o_j has a higher priority.

algorithm to unconstrained motion planning.

Control Stream Programming. The layout conversion requires a joint space-time optimization strategy to fine-tune both $T(o_i,t)$ and $R(o_i,t)$ in the timeline simultaneously. This is because even though the path and movement of each object from its starting position to its ending position are known after initial programming, naively letting these objects simultaneously move would easily cause object-object collision. The key idea of a well-planned control stream is to choose the right object to move at the right time and stop the others that could lead to collisions at the same time. More specifically, let sets $\mathcal{T}_{o_i}$ and $\mathcal{R}_{o_i}$ be the initial control streams for translations and rotations of the object o_i during the conversion, respectively. Since the initial control streams do not have stops, the fine-tuning process is to insert the movements from the sets $\mathcal{T}_{o_i}$ and $\mathcal{R}_{o_i}$ to new collision-free control streams $T(o_i,t)$ and $R(o_i,t)$, with adequate intervals for stop (i.e., zero-value slices).

To achieve this we define forward and backtracking manipulations to handle the movement insertion. The forward manipulation is to move all objects towards their ending positions/orientations, while the backtracking manipulation is to pause or even reverse the movements of certain objects due to the detected collisions. Prior to these manipulations, we first define the priority of an object to determine which one should be adjusted when two objects o_i and o_j collide with each other during layout conversion. Let $P(o_i,o_j)$ be the priority between a pair of objects, with $P(o_i,o_j)=1$ indicating that o_i has a higher priority than o_j, and $P(o_i,o_j)=0$ otherwise. Due to the time cost described in Section 3, we set $P(o_i,o_j)=1$ if o_i has a longer distance to its ending position than o_j.

In each dispersed time slice t, once the forward manipulation is performed on o_i, there will be one of the four types of feedback via collision detection with other objects, denoted as: $F(o_i,t)=[1,1]$, $F(o_i,t)=[1,0]$, $F(o_i,t)=[0,1]$ and $F(o_i,t)=[0,0]$. They respectively mean object o_i can both translate and rotate, only translate, only rotate, and neither translate nor rotate (i.e., be blocked). Based on the element values of $F(o_i,t)$, the forward manipulation pops the associated $\mathcal{T}_{o_i}$ and/or $\mathcal{R}_{o_i}$ into $T(o_i,t)$ and/or $R(o_i,t)$, respectively.

If two objects o_i and o_j collide with each other and both $F(o_i,t)=[0,0]$ and $F(o_j,t)=[0,0]$, we call the movements of o_i and o_j blocked. To address this problem, we employ a backtracking manipulation, which performs an opposite task that pushes $T(o_i,t)$ and/or $R(o_i,t)$ back to $\mathcal{T}_{o_i}$ and/or $\mathcal{R}_{o_i}$, respectively. The feedback of backtracking manipulation is denoted as $B(o_i,t)$. For two collided and blocked objects o_i and o_j, in which o_i is the one with the lower priority, the backtracking manipulation first pushes back translation movement and then rotation movement at time $t-1$ to see if o_j is able to move in time t. If so, $B(o_i,t)=1$, and $B(o_i,t)=0$ otherwise. The backtracking manipulation iterates until the blocking problem is addressed. In Figure 9 we show an intuitive case to illustrate the forward and backtracking manipulations. Our system leverages the two types of manipulations for control stream fine-tuning, following the time cost in Section 3, which encourages the synchronized translations and rotations of all objects. The pseudo-code of this process is provided in Algorithm 1.

Algorithm 1 tunes the timing of the movement sequence(s) and records the tuned sequences(s) in the control stream. Note that the backtracking manipulations might lead to an object o_i with a lower priority rolled back to its initial status. If the forward manipulation for o_i is still blocked, our algorithm heuristically adds an extra movement pair at the beginning of $\mathcal{T}_{o_i}$ and $\mathcal{R}_{o_i}$ for object o_i, and also extends its initial path. As an example in Figure 2, the final programmed path of the tea table is actually created by such a manipulation. This path is not in the initial path determined in Figure 6. The added movement pair consists of a translation or rotation that enforces an associated object to move in an opposite direction against the object with a higher priority, and an opposite motion that puts the object back to its previous status. A new movement pair is inserted into the middle of the previously added movement pair if more than one movement pair is required. Since rotation movements require rotating shaft(s)

Algorithm 1: Control stream programming

```
Input: Movement sets 𝒯_{o_i} and ℛ_{o_i}
Output: Control stream sets {T(o_i, t)} and {R(o_i, t)}
{T(o_i, t)} and {R(o_i, t)} are all-zero sets;
Time t = 0;
while 𝒯_{o_i} ≠ ∅ and ℛ_{o_i} ≠ ∅, ∀o_i do
    Do forward manipulation and get F(o_i, t), ∀o_i;
    for i = 1 to N do
        if F(o_i, t) ≠ [0, 0] then
            Pop 𝒯_{o_i} and/or ℛ_{o_i} to T(o_i, t) and/or
              R(o_i, t) based on F(o_i, t);
        end
    end
    if ∃F(o_i, t) = [0, 0], F(o_j, t) =
      [0, 0], and o_i collides with o_j then
        Δt = 1; Calculate priority P(o_i, o_j);
        Do backtracking manipulation for object o_i
          who has lower priority and get B(o_i, t);
        if B(o_i, t) = 1 then
            Push the non-zero T(o_i, t − Δt) and/or
              R(o_i, t − Δt) to 𝒯_{o_i} and/or ℛ_{o_i};
            Set T(o_i, t − Δt) = 0 and/or R(o_i, t −
              Δt) = 0 ;
        end
        while B(o_i, t) = 0 do
            Push the non-zero T(o_i, t − Δt) and/or
              R(o_i, t − Δt) to 𝒯_{o_i} and/or ℛ_{o_i};
            Set T(o_i, t − Δt) = 0 and/or R(o_i, t −
              Δt) = 0 ;
            Δt = Δt + 1;
            Do backtracking manipulation for object o_i
              and update B(o_i, t);
        end
        Do forward manipulation again for object o_i
          and o_j;
        update {T(o_i, t)}, {R(o_i, t)}, 𝒯_{o_i} and ℛ_{o_i}
          based on F(o_i, t);
    end
    t = t + 1;
end
return {T(o_i, t)} and {R(o_i, t)};
```

with higher cost, the first choice is translation, and rotation is used only if the translation is blocked.

Mechanism Configuration Determination. Following the complexity cost in Equation 1, our approach adjusts the motion mechanism configurations to make the designed mechanical system as simple as possible. The discrete control stream with a uniform speed as we assumed, would ensure the paths to be constructed by integral lead rail units. However, such paths might lead to the ending positions of certain objects no longer as expected in the target layout. For example, two couches might not be sufficiently close to each other to form a bed. Such a subtle error can be eliminated by slightly adjusting the positions of the related objects in the source layout. Specifically, if the positions of the objects after the layout conversion need to be slightly adjusted, we add the offsets (i.e., how the ending position of the object should be adjusted) to the source layout. On the other hand, if the movement of a certain object can be conducted with a base, we will not choose any rotating shaft (for a lower complexity cost). If a certain object only rotates one step (18-degree in our implementation), we would change the direction of such an object in the source layout to eliminate the required rotation. Note that this step is mainly to determine the types of motion mechanisms (i.e., a base or rotating shaft), since their connection positions have been determined in the previously initial path generation step (Section 4.2). This step would not impact Algorithm 1, since it does not change the programmed motion.

Unconstrained Motion Planning. Our system can also be extended for unconstrained motion planning, which is able to generate layout conversions with free-form object movements between multiple indoor layouts. Assuming that many future indoor objects can move themselves (e.g., ratchair [39]), we can easily adapt our method to such scenarios by disabling the mechanical constraints (which enforce the paths to be either horizontal or vertical), and performing our control stream programming algorithm to tackle potential collisions. We will also show some experimental results of unconstrained motion planning, and discuss the advantages of both constrained and unconstrained motion planning in the subsequent section.

5 Experiments and Discussions

In this section, we present various experimental results generated by our method to show its effectiveness. We also show some potential applications to validate the usefulness and scalability of our system.

Experimental Results. In Figure 10, we show two cases of convertible layout design. In the first case, we use a source layout with two different reference layouts, leading to two different mechanical systems by our method. In the second case, we choose two slightly different source layouts with the same reference layout. Our system again successfully programs different conversion processes. The time costs of these results from top to bottom are 8, 19, 14 and 21 steps in relation to the mechanical constraints (Section 3), respectively. The costs indicate that the first designed convertible layout in each case of Figure 10 performs better than the second one. Thus our system is able to quantitatively assess different convertible layout designs and choose better source and/or target layouts.

In Figure 11, we show two more complex cases to evaluate our system for rooms with obstacles or rooms with large space. In the top case, we have two similar sets of source and target layouts, without and with obstacles. Our system generates efficient conversion results and successfully navigates the obstacles. In the bottom case, we display the results of two convertible layout designs for a large classroom created by our system. These results show that our system can tackle large, indoor scenes. Typically, it takes under one minute to generate convertible layouts by our system, including the time for user intervention. We found that our system typically required 0-2 user manipulations for completing the design, while each intervention took about 10 seconds

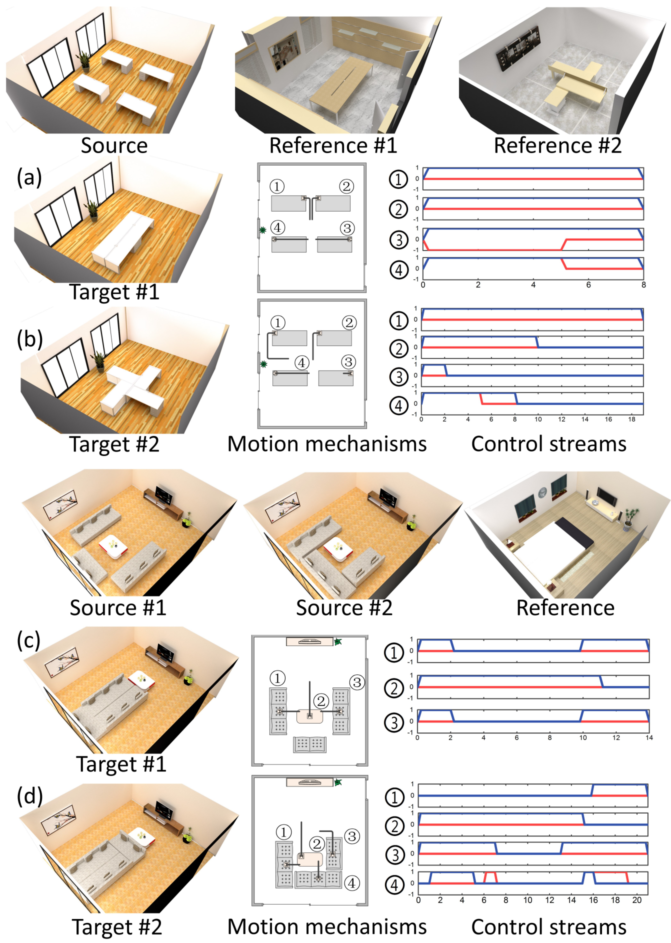


Fig. 10. Two examples of convertible layout design with two different reference layouts (Top) and two different source layouts (Bottom).

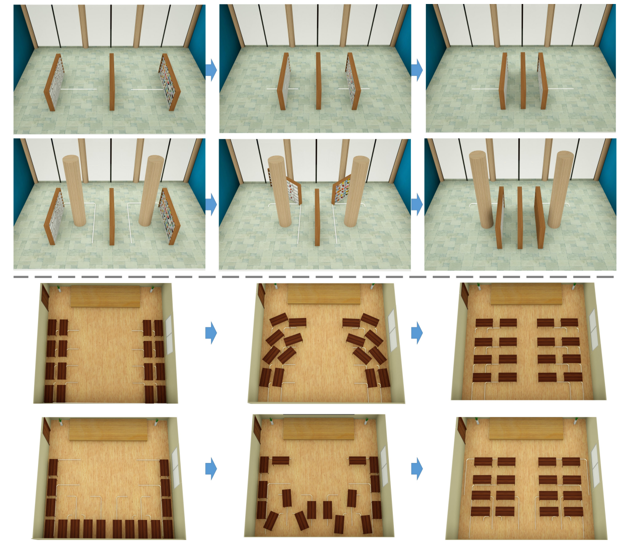


Fig. 11. Additional examples of convertible indoor layouts designed by our system. **Top:** Motion planning results without (Left) and with (Right) obstacles. **Bottom:** convertible layout design for large rooms. We show two results given different source layouts and the same target layouts.

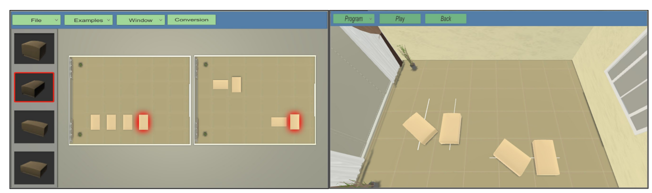


Fig. 12. The user interface of our system for 2D floor plan editing (Left) and 3D scene visualization (Right).

on average. All the experiments were tested on a PC with Intel's Core i5-2.60GHz CPU, 4GB RAM, and the Unity3D platform.

User Interface. Figure 12 shows the user interface of our system. It consists of three panels: the menus (top), a main panel for displaying both a 2D floor plan and a 3D scene (bottom), and an object panel for adding furniture when creating a 2D floor plan (bottom-left). The system UI assists users to quickly build indoor layouts (e.g., Figure 12-left), and intuitively shows the programmed layout conversion process (e.g., Figure 12-right). As mentioned in Section 4.1, users can also manually modify target layouts when the results by the automatic object correspondence establishment algorithm are not satisfactory. For example in Figure 12-left, the right floor plan is the target layout, and the object highlighted in red can be manually modified by the user.

Applications. Our system can be used for several potential applications. For example, in Figure 13, we show two commonly seen applications of convertible layout design in real world. In the first case (Figure 13(Top)), we show how our system can program the seats in a high-speed rail with the capability to adjust their directions. A straightforward approach to this task is to manually turn the orientations of the seats one by one, as shown in Figure 1. Our system automatically programs a more efficient way for layout conversion with synchronous movements, rather than one by one rotating all chairs. In the second case (Figure 13(Bottom)), our system designs a stowable exhibition hall, in which the movable showcases can be arranged in different forms.

In Figure 14, we show two more indoor scene designs created by our system. In these two examples, the convertible layouts are designed with the user-specified, auto-lift objects (labeled in blue in the floor plans). These examples show the potential and scalability of our system for the design of more complex, multi-purpose, and stowable indoor scenes.

As discussed in Section 4.3, our method can be easily extended to unconstrained motion planning. In Figure 15, we show three cases of layout conversion without mechanical constraints (in the left, middle and right columns). The right column also shows that our system can handle non-rectangular room layouts. We observe that a large room typically affords multiple solutions to convertible layouts given multiple reference layouts, and a non-rectangular room could require more translation movements whose directions are neither horizontal nor vertical. Therefore, it would be useful to choose the self-moving furniture to design free-form layout conversions for large or non-rectangular rooms. In contrast, due to the limited layout variations in small rooms, layout conversion under mechanical constraints would be a more suitable choice.

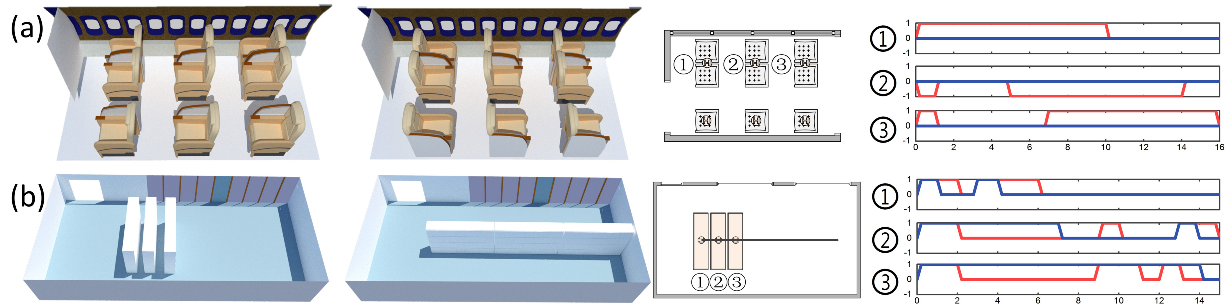


Fig. 13. Two application examples of our system. We show the source (left) and target (right) layouts in each case.

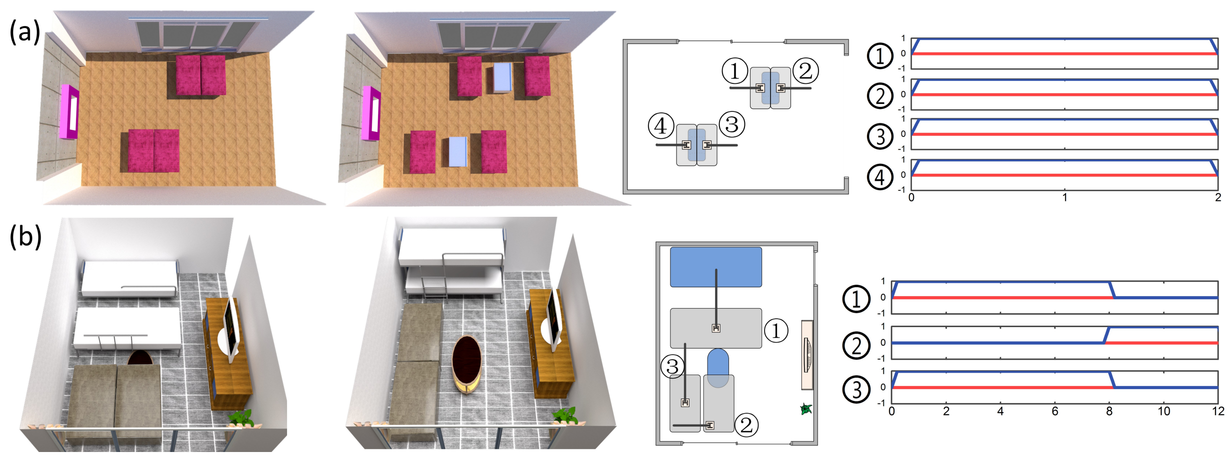


Fig. 14. More complex convertible layout designs with auto-lift objects (highlighted in blue in the floor plans). We show the source (left) and target (right) layouts in each case.

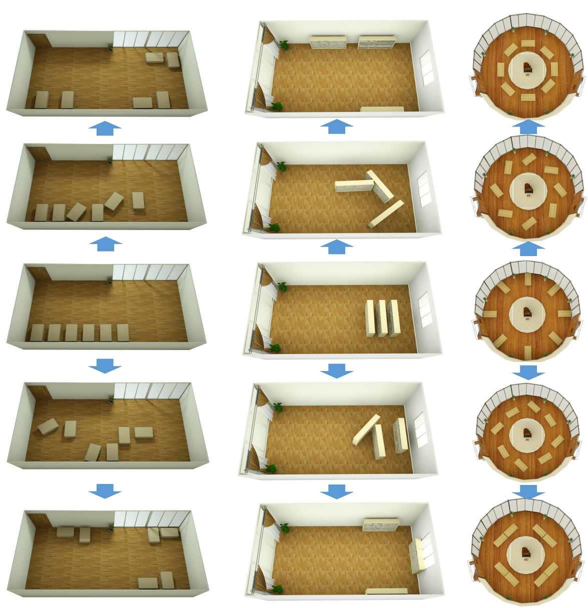


Fig. 15. Examples of unconstrained layout conversion programming, where each auto-driving agent moves freely. The layouts in the middle row can convert to two different layouts in the top and bottom rows with our programmed conversion process.

TABLE 1
The *computational time* of designing convertible layouts with (W/) our system, and the average *interaction time* of the participants for the same tasks without (W/O) our system. All the examples with our system listed in this table are automatically generated.

Figures	2	10-(c)	10-(d)	13-(a)	13-(b)	14-(b)
W/ (sec)	6.5	6.5	14.5	2.1	8.8	7.5
W/O (sec)	37.4	35.3	100.9	40.1	115.1	32.3

User studies. We conducted two user studies to evaluate the efficiency of our system for convertible layout design. In the first one, we compare the time cost between our system and participants for motion planning, when the target layout and initial path are known. This user study mainly evaluates our control stream programming algorithm. In the second one, we test our system in the design of the target layout and initial path, by comparing its time cost to the participants. This is to validate the efficiency of our convertible layout design system.

In the first user study, we invited 10 computer science undergraduate students (6 males, & 4 females, aged from 23 to 25) to participate in the study. All of them have basic training in 3D modeling. Each participant was given six different layout pairs (i.e., the source and target layouts) with the associated initial motion paths, presented in a random order. They were asked to plan the motion for converting each source layout to the corresponding target layout, with and without our system. For the latter, the participants had to manually plan the movements of objects through a simple interface where the participants could adjust the object motion in a timeline to address potential object collisions, highlighted using the method of [38]. Table 1 summarizes the comparison results between the *computational time* of our system and the average *interaction time* of the participants without our system. In this table, all the cases with our method do not need any user intervention. It can be seen that even with the highlight for object collision, the participants needed significantly more time to design each convertible layout without our system. Only for the case in Figure 13-(a) our method still requires user assistance to adjust the object priorities, which additionally takes about 12 seconds. But the total time (14.1 seconds) needed to create this example with our system is still significantly less than that (40.1 seconds) needed by the participants (without our system).

In the second user study, 5 undergraduate students (4 males & 1 female; aged from 23 to 25) were invited to design target layouts and initial paths, given nine different source layouts and the associated reference layouts. We taught the participants how to examine whether a designed convertible layout is workable or not by showing them several positive examples as well as the layout conversion process in advance. In this process, the participants manually manipulated the position/direction of each object within each source layout to generate a target layout through the UI of our system. Afterwards, we provided a 2D drawing interface to assist the participants in drawing the paths (obeying our mechanical constraints) between the source and target layouts. Note that our system did not suggest any object correspondence or potential path here. We then tested the results designed by the participants through our system to determine whether this task was completed. If not, we would identify to the participants where the issue existed and let them refine the target layout or paths until the updated results passed the test. On average, each participant spent about 119 seconds completing each design task, while our system needed only less than 1 second for computation and about 9 seconds for user intervention. Moreover, we also analyzed the quality of the target layouts and initial paths created by the participants and our system, by using them as the inputs of the control stream programming algorithm. We then compared both the complexity costs and time costs of these results (i.e., convertible layouts). On average, the complexity costs for the participants' results and ours are 41.0 and 42.4, respectively. Assuming each time step of movement takes 1 second, the time costs for the participants' results and ours are 13.9 and 13.6 seconds, respectively. We found that most of the paths designed by the participants and our system are similar. Because the participants can frequently adjust their target layouts, in some cases, the participants' results could have lower costs than ours. However, considering the design time surges almost twelve-fold for human users without our system, it shows the usability and efficiency of our system for convertible layout design.

Limitations. Despite the above-demonstrated effectiveness, our current system still has several limitations:

- First, the quality of the programmed layout conversion process relies on the input layouts of our system. For example, our system requires that the source and reference layouts are similar in terms of both geometry and function. When two given layouts are significantly different, it is difficult to design a reasonable conversion process even for professional designers. Our system might fail to produce convertible layouts for input scenes with too limited space. For example in Figure 16-Top, when the layout converts from the left scene to the right one, the conversion is stuck in a status shown in the middle, due to the lack of adequate space to support the rotation of the desk. This problem might be addressed by manual intervention, e.g., specifying certain objects to be auto-lift.
- Second, we have several simplified assumptions at both the object correspondence establishment step and the target layout creation step: (i) we only consider the distance similarity and size similarity to establish object correspondences while ignoring the semantic and functional similarities; (ii) we determine the orientation of an object by fitting it to the corresponding one in the reference layout, which could lead to wrong orientations (Figure 16-Bottom) especially for objects from different categories. Such problems are currently fixed through user assistance in our system.
- Third, besides the target layout refinement, there are two scenarios that might need user assistance in our current system: (i) since large scenes often have many objects leading to complex collision scenarios (e.g., Figure 11-bottom), user assistance might also be required to replace a certain suggested initial path with the other one, to improve the quality of the initial path and thus decrease the difficulty of the subsequent motion planning task; (ii) our motion planning is done in a greedy manner. This means that the objects with high priorities have a higher influence on the solution, and might fail to generate a solution of the control stream programming caused by complex object collisions. To fix such an issue, we allow users to change the priorities of the collided objects.
- Lastly, the mechanical constraints considered in this work are somewhat limited. Therefore, our current

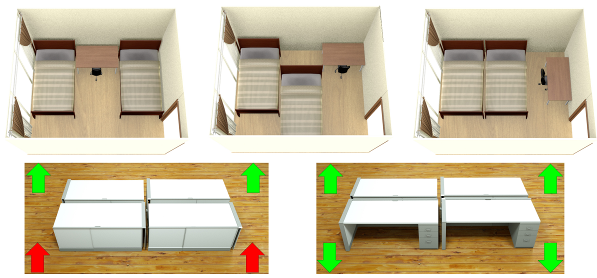


Fig. 16. **Top:** Two indoor layouts (Left and Right) without a feasible solution of layout conversion, since the room size limits the rotation of the desk (Middle). **Bottom:** The target layouts before (Left) and after (Right) user intervention to manually fix the orientations of the two desks at the bottom. We highlight the correct and incorrect orientations with green and red arrows, respectively.

system might be still insufficient to handle the convertible layout designs with foldable or deformable indoor objects. Besides, our current system only focuses on large movable furniture, but not on small or lightweight furniture objects like chairs. It might be more economical to move these objects manually after the layout conversion, rather than automatically move them through complex motion mechanisms.

6 Conclusion

In this paper, we have presented a system for motion planning that enables the design of convertible indoor scene layouts, with a minimal amount of user assistance. Our system first establishes the object correspondences between two given source and reference layouts, and then initially programs the paths and associated movements of all the objects based on the mechanical constraints. Finally, our system fine-tunes the control stream and optimizes the motion mechanisms to avoid object collisions and improve conversion efficiency. In this way, our system can assist the design of indoor scenes with convertible layouts, and inspire multi-purpose indoor scene creation.

As the future work, we plan to extend our current system to handle more types of motion mechanisms and the reconfigurable function of indoor objects such as foldability, to design more complex convertible indoor layouts. Meanwhile, discovering the composability of difficult shapes and functionality of indoor objects, or leveraging a large indoor layout database to search for potential convertible layouts, will also be interesting to explore.

Acknowledgements

We thank the anonymous reviewers for the constructive comments. This work was partially supported by grants from the the NSFC (No.61902032, No.61962021), NSF (IIS-1524782), Research Grants Council of the Hong Kong Special Administrative Region, China (No. CityU 11237116), the Open Project Program of the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (VR-LAB2019B01), City University of Hong Kong (No. 7004915), the Fundamental Research Funds for the Central Universities (2020RC17), the China Postdoctoral Science Foundation under Grant (2019M662261), and the Key Research and Development Program of Jiangxi Province under Grant (20192BBE50079).

References

[1] M. Fisher, M. Savva, and P. Hanrahan, "Characterizing structural relationships in scenes using graph kernels," *ACM transactions on graphics (TOG)*, vol. 30, no. 4, p. 34, 2011.

[2] A. Sharf, H. Huang, C. Liang, J. Zhang, B. Chen, and M. Gong, "Mobility-trees for indoor scenes manipulation," in *Computer Graphics Forum*, vol. 33, no. 1. Wiley Online Library, 2014, pp. 2–14.

[3] 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 (TOG)*, vol. 36, no. 6, p. 201, 2017.

[4] R. Ma, H. Li, C. Zou, Z. Liao, X. Tong, and H. Zhang, "Action-driven 3d indoor scene evolution." *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, pp. 173–1, 2016.

[5] L. Majerowicz, A. Shamir, A. Sheffer, and H. H. Hoos, "Filling your shelves: Synthesizing diverse style-preserving artifact arrangements," *IEEE TVCG*, vol. 20, no. 11, pp. 1507–1518, 2014.

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

[7] 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 (TOG)*, vol. 33, no. 4, pp. 1–12, 2014.

[8] 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 (TOG)*, vol. 31, no. 4, p. 56, 2012.

[9] P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun, "Interactive furniture layout using interior design guidelines," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, pp. 87:1–87:10, 2011.

[10] L. F. Yu, S. K. Yeung, C. K. Tang, D. Terzopoulos, T. F. Chan, and S. J. Osher, "Make it home: automatic optimization of furniture arrangement," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, pp. 1–12, 2011.

[11] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan, "Example-based synthesis of 3d object arrangements," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, pp. 1–11, 2012.

[12] X. Chen, J. Li, Q. Li, B. Gao, D. Zou, and Q. Zhao, "Image2scene: Transforming style of 3d room," in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 321–330.

[13] R. Ma, A. G. Patil, M. Fisher, M. Li, S. Pirk, B.-S. Hua, S.-K. Yeung, X. Tong, L. Guibas, and H. Zhang, "Language-driven synthesis of 3d scenes from scene databases," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, p. 212, 2018.

[14] M. Fisher, M. Savva, Y. Li, and P. Hanrahan, "Activity-centric scene synthesis for functional 3d scene modeling," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 179, 2015.

[15] M. Savva, A. X. Chang, P. Hanrahan, and M. Fisher, "Pigraphs: learning interaction snapshots from observations," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 139, 2016.

[16] S. Qi, Y. Zhu, S. Huang, C. Jiang, and S.-C. Zhu, "Human-centric indoor scene synthesis using stochastic grammar," in *CVPR'18*, 2018, pp. 5899–5908.

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

[18] 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 (TOG)*, vol. 38, no. 2, p. 12, 2019.

[19] N. J. Mitra, Y.-L. Yang, D.-M. Yan, W. Li, and M. Agrawala, "Illustrating how mechanical assemblies work," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 58, 2010.

[20] R. Hu, W. Li, O. Van Kaick, A. Shamir, H. Zhang, and H. Huang, "Learning to predict part mobility from a single static snapshot," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, p. 227, 2017.

[21] M. Xu, M. Li, W. Xu, Z. Deng, Y. Yang, and K. Zhou, "Interactive mechanism modeling from multi-view images," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 6, p. 236, 2016.

[22] M. Lin, T. Shao, Y. Zheng, N. J. Mitra, and K. Zhou, "Recovering functional mechanical assemblies from raw scans," *IEEE TVCG*, vol. 24, no. 3, pp. 1354–1367, 2018.

[23] B. Koo, W. Li, J. Yao, M. Agrawala, and N. J. Mitra, "Creating works-like prototypes of mechanical objects," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 6, 2014.

[24] B. Gao, X. Chen, J. Li, and D. Zou, "Modeling interactive furniture from a single image," *Computers & Graphics*, vol. 58, pp. 102–108, 2016.

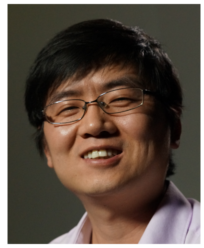
[25] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3357–3364.

[26] K. Xu, L. Zheng, Z. Yan, G. Yan, E. Zhang, M. Niessner, O. Deussen, D. Cohen-Or, and H. Huang, "Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields," *ACM Transactions on Graphics*, vol. 36, no. 6, p. Article No. 202, 2017.

[27] L. Peng and Y. Jia, "Consensus of a class of second-order multi-agent systems with time-delay and jointly-connected topologies," *IEEE Transactions on Automatic Control*, vol. 55, no. 3, pp. 778–784.
[28] Y. Liu and Y. Jia, "An iterative learning approach to formation control of multi-agent systems," *Systems Control Letters*, vol. 61, no. 1, pp. 148–154, 2012.
[29] K. Oh, M. Park, and H. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, no. 53, pp. 424–440, 2015.
[30] Q. Fu, X. Chen, X. Su, and H. Fu, "Pose-inspired shape synthesis and functional hybrid," *IEEE TVCG*, vol. 23, no. 12, pp. 2574–2585, 2017.
[31] R. Hu, O. van Kaick, B. Wu, H. Huang, A. Shamir, and H. Zhang, "Learning how objects function via co-analysis of interactions," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 47, 2016.
[32] P. Song, C.-W. Fu, and D. Cohen-Or, "Recursive interlocking puzzles," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 128, 2012.
[33] B. Koo, J. Hergel, S. Lefebvre, and N. J. Mitra, "Towards zero-waste furniture design," *IEEE TVCG*, vol. 23, no. 12, pp. 2627–2640, 2017.
[34] P. Song, C.-W. Fu, Y. Jin, H. Xu, L. Liu, P.-A. Heng, and D. Cohen-Or, "Reconfigurable interlocking furniture," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, p. 174, 2017.
[35] J. Zhou and X. Chen, "Convertible furniture design," *Computers & Graphics*, vol. 70, pp. 165–175, 2018.
[36] H. Li, I. Alhashim, H. Zhang, A. Shamir, and D. Cohen-Or, "Stackabilization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, p. 158, 2012.
[37] H. Li, R. Hu, I. Alhashim, and H. Zhang, "Foldabilizing furniture." *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, pp. 90–1, 2015.
[38] A. Garg, A. Jacobson, and E. Grinspun, "Computational design of reconfigurables." *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 90–1, 2016.
[39] T. Parshakova, M. Cho, A. Cassinelli, and D. Saakes, "Ratchair: Furniture learns to move itself with vibration," in *ACM SIGGRAPH 2016 Emerging Technologies*. New York, NY, USA: Association for Computing Machinery, 2016.

Guoming Xiong is a Ph.D. student of East China Jiaotong University. He received the B.S. degree in computer application from Army Officer Academy of PLA, in 2011. His research interests include Computer Graphics and Virtual Reality.

Qiang Fu is an Assistant Professor in the School of Digital Media and Design Arts, Beijing University of Posts and Telecommunications. Before that, he was a postdoctoral fellow with the Department of Computer Science, University of Houston, US. He received the Ph.D. degree in computer science from Beihang University, China, in 2018. His research interests include Computer Graphics and Virtual Reality.



Hongbo Fu is a Professor in the School of Creative Media, City University of Hong Kong. He received the PhD degree in computer science from the Hong Kong University of Science and Technology in 2007 and the BS degree in information sciences from Peking University, China, in 2002. His primary research interests fall in the fields of computer graphics and human computer interaction. He has served as an associate editor of The Visual Computer, Computers & Graphics, and Computer Graphics Forum.

Bin Zhou received his B.S. and Ph.D. degrees in Computer Science from the Beihang University, China, in 2006 and 2014, respectively. He is currently an Assistant Professor with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University. His research interests include Computer Graphics, Virtual Reality, Computer Vision and Robotics.

Guoliang Luo earned his PhD in Computer Science at University of Strasbourg in 2015. His research interests includes the Computer Graphics and Artificial Intelligence. He is currently an Associate Professor at East China Jiaotong University. He was enrolled in the Ganjiang Outstanding Youth Talent Program in 2018.

Zhigang Deng received the BS degree in mathematics from Xiamen University, China, the MS degree in computer science from Peking University, China, and the PhD degree in computer science from the Department of Computer Science, University of Southern California, in 2006. He is a professor of computer science with the University of Houston. His research interests include computer graphics, computer animation, and HCI.