Automatic Generation of Commercial Scenes

Shao-Kui Zhang
shaokui@tsinghua.edu.cn
Tsinghua University
Haidian, Beijing, China

Jia-Hong Liu
xhljh1127@163.com
Tsinghua University
Haidian, Beijing, China

Yike Li
liyike0206@gmail.com
Tsinghua University
Haidian, Beijing, China

Tianyi Xiong
xty620682@gmail.com
Tsinghua University
Haidian, Beijing, China

Ke-Xin Ren
496377121@qq.com
Tsinghua University
Haidian, Beijing, China

Hongbo Fu
hongbofu@cityu.edu.hk
City University of Hong Kong
Hong Kong

Song-Hai Zhang*
shz@tsinghua.edu.cn
Tsinghua University & BNRist
Haidian, Beijing, China

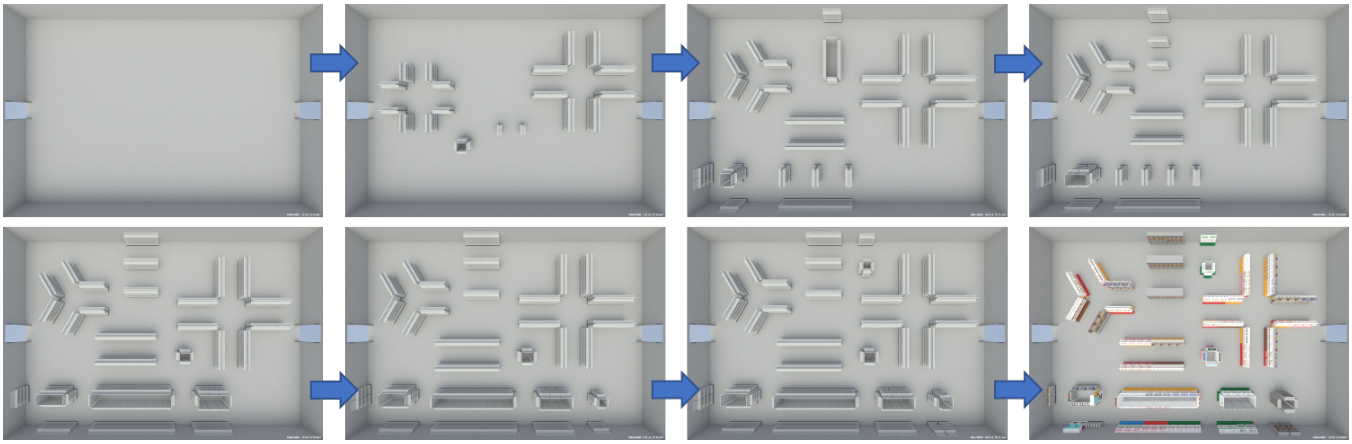


Figure 1: We present a method to synthesize market-like scenes automatically. Initialized with an empty space (Left-Up), our system iteratively adds patterns and modifies them with respect to constraints formulated by commercial design rules. When an optimization converges, commercial objects are automatically inserted following the generated patterns (Right-Bottom).

ABSTRACT

Commercial scenes such as markets and shops are everyday scenes for both virtual scenes and real-world interior designs. However, existing literature on interior scene synthesis mainly focuses on formulating and optimizing residential scenes such as bedrooms, living rooms, etc. Existing literature typically presents a set of relations among objects. It recognizes each furniture object as the smallest unit while optimizing a residential room. However, object relations become less critical in commercial scenes since shelves are often placed next to each other so that pre-calculated relations of objects are less needed. Instead, interior designers resort to evaluating how groups of objects perform in commercial scenes, i.e., the smallest unit to be evaluated is a group of objects. This paper presents a system that automatically synthesizes market-like commercial scenes in virtual environments. Following the rules of commercial layout design, we parameterize groups of objects as "patterns" contributing to a scene. Each pattern directly yields a human-centric routine locally, provides potential connectivity with other routines, and derives the arrangements of objects concerning itself according to the assigned parameters. In order to optimize a scene, the patterns are iteratively multiplexed to insert new routines or modify existing ones under a set of constraints derived from commercial layout designs. Through extensive experiments, we demonstrate the ability of our framework to generate plausible and practical commercial scenes[1].

*Corresponding Author, with Tsinghua University and Beijing National Research Center for Information Science and Technology (BNRist).

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

ACM ISBN 979-8-4007-0108-5/23/10.
https://doi.org/10.1145/3581783.3613456

[1]Code, video and supplementary document are publicly available at: https://github.com/Shao-Kui/3DScenePlatform#commercialscene.

CCS CONCEPTS

• **Computing methodologies → Computer graphics**.

KEYWORDS

Commercial Place, Markets/Shops Planning, Scene Synthesis

ACM Reference Format:

Shao-Kui Zhang, Jia-Hong Liu, Yike Li, Tianyi Xiong, Ke-Xin Ren, Hongbo Fu, and Song-Hai Zhang. 2023. Automatic Generation of Commercial Scenes. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3581783.3613456

1 INTRODUCTION

Whereas substantial literature has appeared demonstrating the automatic modeling of residential scenes [11, 33, 40], the automatic generation of realistic commercial indoor configurations has not yet received the attention it deserves. A reasonable market-like commercial scene benefits the perceptions of how customers understand and construct a sense of direction. Stores and shops are less likely to be desolated. It also prevents customers from getting lost and helps them accurately locate their targets, such as fruits or snacks. More importantly, the consumption experiences increase because of it, thus extending the shopping time and reducing the dead space [7].

Designers refer to some principles when designing commercial scenes, such as: 1) Ensuring that all shops and commodities can be reached and seen; 2) Making roads as coherent as possible without dead ends; 3) Showing consumers the simplest possible routines; 4) Raising the diversity and flexibility of space in order to help customers locate themselves. However, it would be too tedious and impractical to craft every scene manually.

Commonly, automatically modeling a residential room requires plausible and aesthetic spatial relations between/among objects [19, 32, 51]. Therefore, existing solutions of synthesizing indoor scenes formulate self constraints [40, 44], pairwise relations [9, 22, 35], n-ary relations [47], etc, based on furniture objects. Subsequently, during optimization, each object is treated as a unit. All units are transformed according to the relations and constraints among them, where the importance of objects might be different, and some objects may dominate others. For example, a dinning table would dominate several chairs.

In contrast, in a commercial room, the placement of individual objects becomes less critical since designers would conversely focus more on the overall performance of a scene. For example, how to utilize a given area well (given the expensive rent of a shopping mall) or how to make customers stay longer (thus potentially selling more commodities). In such a scenario, the importance of individual objects is similar. Thus, objects have the equal priority. In other words, modeling a commercial room focuses more on procedurally planning groups of objects, which are organizational connections that emerge when customers move into interior spaces of buildings. In contrast, objects such as stores and shelves are placed successively [49].

In this paper, we present a method to automatically synthesize market-like scenes so that the layouts of commodity objects are subsequently arranged, as shown in Figure 1. The synthesis is achieved by organizing groups of objects into "patterns" [23, 31], considering the components of commercial scenes. As demonstrated by [49], each "pattern" expresses a plausible layout of a local flow and an aesthetic arrangement of commercial objects based on it. Thus, our problem is addressed by mathematically formulating patterns and then algorithmically merging a set of patterns as a whole scene. In general, six patterns are derived from the four conceptual and fundamental principles [23, 31]. Furthermore, each pattern is parameterized, e.g., a cross pattern can be "two rows and two columns" or "three rows and three columns" given different parameters (Section 4). We employ an iterative approach with appropriate loss functions guiding the optimization to synthesize a commercial scene. The loss functions prevent the synthetic and intermediate results from being implausible or impracticable in many aspects, including space utilization, elasticity and roundness (Section 5). Subsequently, the optimization iteratively attempts to add more appropriate patterns, tune existing patterns' parameters, or adjust relative transformations (translations/rotation) among patterns from a given empty room shape (Section 6).

To evaluate our method, first, we quantitatively demonstrate the competitive plausibility and aesthetics of our resulting scenes compared with the scenes by the interior designers. Leveraging the immersive VR techniques, we then measure how the synthetic commercial scenes help users efficiently and effectively route in them, compared with the scenes crafted by 3 professional designers related to commercial space design. Finally, in VR, we measure how our method pays as much commercial value as possible compared with designers.

In sum, this paper makes the following contributions:

- We present a method that automatically synthesizes market-like scenes.
- We propose a computational representation of commercial scenes by decomposing them into patterns.
- We quantitatively formulate the plausibility and efficiency of commercial scenes in configurations of patterns.

Note that our method focuses on generating commercial scenes such as markets and shops within a single space. For larger spaces such as malls, our method can be generalized by replacing market-like patterns with mall-like patterns.

2 RELATED WORKS

Generating scenes has been extensively explored in the past decades. Automatic indoor scene synthesis focuses on arranging furniture objects functionally and aesthetically. [22, 26, 35, 40, 43] synthesize scenes by mathematically formulating residential interior rules, such as hierarchical relationships, relative distances, etc. [16, 27, 33, 34, 50] also synthesize indoor layouts. However, they leverage neural networks by feeding random numbers or top views to the networks. A few works propose to utilize geometries of objects and rooms to yield layouts [11, 47]. Additional inputs have been incorporated to augment scene synthesis. Types of visual inputs range from sketches [37] to RGB images [20, 39, 46] or RGB-D scans [4, 5]. Texts are interpreted as scene graphs [3, 21]. Existing example scenes are multiplexed to derive new scenes [9, 36] while existing real-world blocks are considered constraints when synthesizing scenes [12]. The key difference between existing synthesis

techniques and ours is the former considers "objects" as the targets of optimization instead of decomposed patterns, which cannot be directly described using only translations, rotations, and scales.

Existing literature develops various ways to capture user behaviours to encode human-centric priors when synthesizing contents. [17, 48] capture how humans interact with the functional workspace and rearrange the layout so that the workspace becomes more efficient. Fu et al. [11] define functional purposes among furniture objects to synthesize new scenes. Fisher et al. [10] incorporate human activities and learn action maps [28] to guide the conversion from RGB-D scans to scenes. Savva et al. [29] generate poses and layouts by learning priors of the human poses and object geometries during human interaction with the objects. A few works directly accept active user inputs to generate results satisfying user preferences [41, 42, 44, 45]. We propose using commercial scene patterns, which are the human activities of routing scenes. Thus, we directly optimize the patterns to synthesize a human-centric layout.

Previous works have investigated the necessity of commercial scene planning extensively. For example, Chandon et al. [2] show that shelf facing directions have a strong impact on consumers' evaluation at the point of purchase. Alawadhi and Yoon [1] perform 3D computer simulations and recognize the critical role of store layouts in controlling shopping motivations. Ebster [6] proposes incorporating nodes, edges, and districts in a store as cognitive map features for better shopping orientation. Several layout types have also been studied. Levy and Weitz [15] generally discuss two forms of layout in retail spaces: grid and free-form. Ebster [6] describes the boutique, star, and arena layouts as three types of frequently used freeform layouts. The enclosed counter layout and the single-routed forced-path layout are also mentioned. Design software such as Edraw, ConceptDraw, and SmartDraw could be utilized to efficiently develop a flat-level indoor layout plan.

3 OVERVIEW

Given an empty space, for example, representing a market-like scene, the problem is formulated as optimizing a set $L^* = \{l_i | i \in [1, n]\}$, such that each pattern l_i is correctly transformed into the space. As illustrated in Fig. 2, we define four representative types (including two variations) of patterns based on interior rules, where each type expresses how humans travel in it. For example, the most straightforward pattern, "Linearity", implies that humans walk straight from one end to the end, while "Circulation" implies that they will travel back to the starting point. "Rescaling" a pattern refers to adjusting the parameters of l_i instead of simply stretching it w.r.t the axes as a whole. For example, rescaling the patterns of the Linearity type requires two coordinates, while rescaling the patterns of the Circulation type requires the inner and outer radii. "Transforming" a pattern also implies adjusting the positions and rotations of l_i, e.g., moving a pattern from the left to the right corner of a square room (Section 4).

Given the representations of patterns described above, our goal is to integrate them into an optimization framework. The framework needs an appropriately designed cost function. This function requires several constraints that guide the synthetic process of patterns. Each constraint function corresponds to a criterion about the realism or functionality of existent patterns in the scenes The optimization framework adjusts the patterns based on the quantitative values of the constraint functions. For example, layout patterns cannot overlap, and typical markets need elastic spaces to enable resting and free activities. If a few patterns overlap, the optimization tries to separate the collided patterns. At the same time, it may reduce the number of patterns or rescale the existent patterns to address the elastic spaces (Section 5).

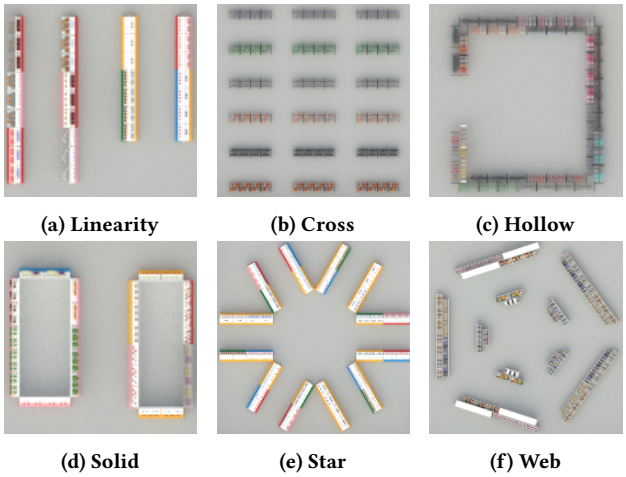


Figure 2: Example for four representative types of patterns with additional variations. The "Hollow" and "Solid" are the variations of "Circulation". The "Star" and "Web" are variations of "Radiation".

The search space of L^* is highly complex since patterns are interdependent. The patterns' scales and their connections depend on various constraints, so it is hard to have a closed-form solution or a global optimization function referring to a one-of-a-kind result. Consequently, we resort to the Metropolis-Hastings algorithm, a stochastic optimization method widely used in the domain of scene synthesis [18, 26, 40]. To search for a good approximation to the global optimum, we also demonstrate our technique to propose movements from L_i to L_{i+1} to avoid an exhaustive search in the complex space (Section 6).

4 PATTERNS

A commercial layout guides how humans route in a given space, i.e., humans should find their customized routines in their continuous exploration, instead of aimlessly hovering in a "maze", so a scene should provide clear and purposeful pathways to humans [49]. This section presents how to express such layouts computationally by formulating patterns according to interior designs [23, 31]. Figure 2 shows four representative types of patterns, where each type plays the role of a template for deriving various patterns to compose the final layout. We will first introduce the rationales for selecting the following patterns and then demonstrate the formulations.

Linearity. The linear pattern is suitable for long and narrow flows, with only one main road connecting two points. The commodities are arranged on both sides of the central passage. The advantages of this pattern are direct and simple, with high browsing

rates of shops or commodities. The disadvantage is that passengers are hard to revisit a linear pattern and quickly feel bored. If a pattern l_i belongs to the type of Linearity, it is expressed as $(x_i^0, y_i^0, x_i^1, y_i^1)$, i.e., the two endpoints of the pattern. Thus, a linearity pattern could be a horizontal flow, a vertical flow, or even a sideways flow. When a linear pattern is constructed, objects are placed on the left and right sides of the flow. To fully utilize the space as much as possible, the outer side of this pattern can also be filled with objects, e.g., the two patterns in Figure 2a.

Cross. Generally, a cross pattern is crisscrossed by several "lines", as shown in Figure 2b, thus forming a grid shape. Cross patterns are suitable for square-shaped commercial spaces with a large single-floor area. They can pack a lot more items in the space and provide relatively equal exposure for every commodity, while customers are easier to get lost. If a pattern l_i belongs to a cross pattern, it is expressed as (n_i^w, n_i^h, Ω_i), which sets the number of horizontal flows n_i^w, the number of vertical flows n_i^h, and the width of clearances Ω_i between flows. Similar to linear patterns, objects in cross patterns are placed along the left and right sides of the flows.

Circulation. In the circular pattern, the local area forms a circular loop. Shops and commodities are placed on the inner or outer "rings" of the pattern, which has no or few intersections with other patterns. The circular pattern is popular in modern commercial spaces, and customers easily revisit it, while the disadvantage is the low space utilization. A circular pattern is expressed as (r_i^0, r_i^1), where r_i^0 and r_i^1 are the long and short side lengths. Two variations exist for the circular pattern: hollow pattern and solid pattern. First, users could enter the interior of a hollow pattern, as shown in Figure 2c, where objects are organized in both the interior and exterior of the pattern. The hollow space reserves elastic areas (see Section 5). Second, objects are organized following the exterior in a hollow pattern, which is suitable for small rooms. The variations of the same pattern share the same rule of parameterization.

Radiation. A radial pattern is suitable for spacious areas or scenarios where we want to divert customers into other parts of the scenes. Typical radial patterns are shaped similar to spider webs. The advantage of a radiation pattern is its high utilization rate. However, the disadvantages are obvious: customers are hard pressed to visit all items and revisit specific items, i.e., we cannot guarantee that customers could browse or re-browse commodities, thus resulting in an unbalanced layout. Radial patterns also easily cause dead ends. If a pattern l_i belongs to Radiation with N branches, it is expressed as $(\theta_i, d_i, \{\vec{\gamma}_1, \vec{\gamma}_2, ..., \vec{\gamma}_N\})$, which denotes an orientation θ_i, a length d_i of branches, and a set of direction vectors $\{\vec{\gamma}_1, \vec{\gamma}_2, ..., \vec{\gamma}_N\}$ representing the branches. Two variations also exist for the radial pattern. First, similar to a "star" shape, branches originating from the center are independent, as shown in Figure 2e, where the gap spaces between the branches can be used for other patterns. Second, similar to a "web" shape, we could break through the branches on both sides and enable customers to walk across branches, as shown in Figure 2f, which consumes more space but allows more flexible routines.

5 CONSTRAINTS

As discussed, patterns are optimized with constraints in order to generate plausible scenes [13, 24]. Given a set of patterns L, a room shape R and the entrance/exit of a commercial space W, the overall cost function is shown in Equation 1, which is a weighted sum of several specified and detailed constraints. $f_u(\cdot)$, $f_e(\cdot)$ and $f_r(\cdot)$ respectively denote "space utilization", "elasticity", and "roundness", which will be illustrated in the following discussions. The summation of the coefficients λ equals to 1.

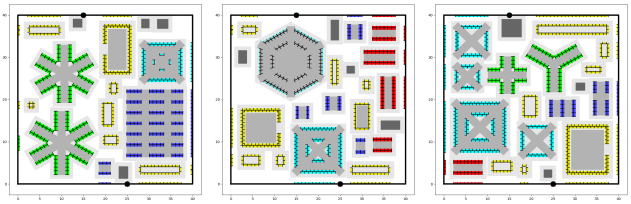


Figure 3: Plotting patterns of commercial scenes. Each color represents a type of patterns: Linearity (Red), Cross (Blue), Circulation (Yellow), Radiation (Green), and Empty (Black). Each pattern is discretized with several placeholders for placing objects such as fruit counters or vegetable counters. We also mark whether each pattern has the internal space allowing users to route (Dark Gray) and the external space belonging to the pattern (Light Grey).

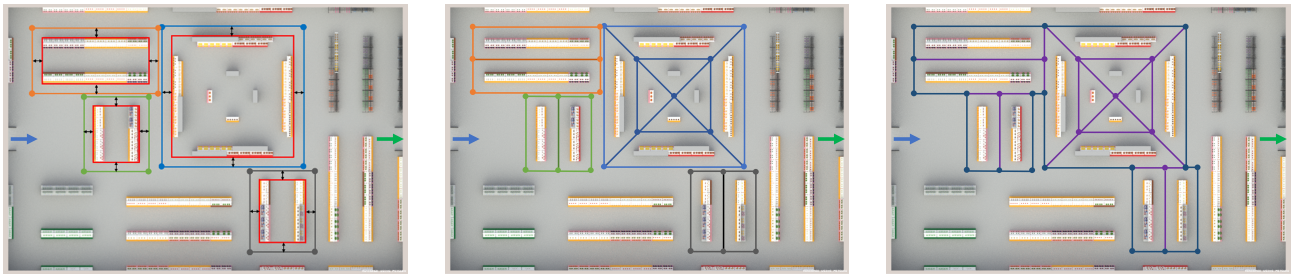
$$\Phi(L, R, W) = \lambda_u f_u(L, R) + \lambda_e f_e(L, R) + \lambda_r f_r(L, W). \quad (1)$$

Utilization. Since rooms for commercial purposes are expensive due to the reprises, we have to utilize the spaces as much as possible so that more shops and shelves are placed. $f_u(L, R)$ returns the space utilization of a set of patterns L in a given room R, as shown in Equation 2. $\beta(\cdot)$ denotes the "buffer" operation on shapes of patterns, as illustrated in Figure 3 colored with light grey, where each pattern is broadened so that pathways are reserved for humans. $\alpha(\cdot)$ returns the area and takes the room shape R.

$$f_u(L, R) = \sum_i \beta(l_i)/\alpha(R). \quad (2)$$

Elasticity. Markets typically reserve a few elastic spaces where no commercial objects are placed. Instead, objects such as benches and sofas are placed in practice. This is necessary since customers need a place to rest or organize items and market owners need places to hold activities. An "empty" pattern is introduced to represent an elastic space. The empty pattern is not inserted into the pattern set L, but it participates in other calculations in the optimization, such as collisions (see Section 6). Subsequently, elastic areas are defined by the summation of the areas of all empty patterns, and the elasticity $f_e(L, R)$ returns the summation ratio to the area of the room shape R.

Roundness. An excellent commercial space design makes customers stay longer. It keeps them browsing all the commodities as far as possible, which is also known as "routines of customers" [49]. Keeping longer shopping sessions refers to the "roundness" [13, 24]. As shown in Figure 5, the solid lines refer to the roundness (i.e., the most extended pathways) of the presented scenes. Mathematically, the roundness $f_r(L_i)$ equals the ratio of the longest path to the total length. Converting a set of patterns, a few elastic spaces, and the gates to a structural graph is inherently difficult for two reasons. First, links (edges) between patterns depend on their gap spaces



(a) Width of Edges (b) Local Connections (c) Merging Edges

Figure 4: The illustration of converting patterns to graphs. (a) Each pattern occupies its surrounding area for half of the pathway (Black Arrows). (b) Local connections of each pattern form the subgraphs, e.g., the orange, blue, brown and green connections, where the circles represent vertices and the lines represent edges. (c) The adjacent exterior parts (Blue) are merged to represent a single route. For example, the left-up (Orange) and middle (Blue) connections are eventually merged into the concatenation, because the two corresponding patterns share the mutual pathway. The blue and green arrows represent the entrances and exits, respectively.

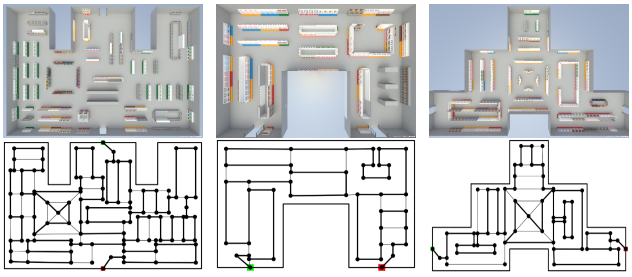


Figure 5: A few examples of constructing graphs. The top row refers to the generated scenes, and the bottom row refers to the corresponding graphs. The Green and Red refer to an entrance and exit, respectively. The solid lines denote the Roundness, which is the longest path.

which are not explicit graph structures. Second, if two patterns are too close to each other, their adjacent space is considered a single route instead of two.

Thus, we calculate the roundness by (1) converting patterns to local subgraphs, (2) merging edges of adjacent subgraphs, and (3) solving the NP-complete longest-path problem [30]. The first two procedures are illustrated in Figure 4. The subgraphs initially occupy the pathways nearby, and their interiors are connected according to the patterns. Then, an adjacent edge is merged if two patterns share the same pathway. For the longest path, every vertex is visited no more than once, so no circuit exists in the longest path. Figure 5 shows a few examples of the generated longest paths for roundness, with the corresponding scenes. See a mathematical and algorithmic formulation of the roundness in Section A.

6 SCENE SYNTHESIS

Our method applies a Metropolis-Hastings algorithm for optimization to acquire an exemplary configuration efficiently. Given an empty configuration $L_0 = \emptyset$, the optimization follows two steps iteratively to explore potential solutions, as shown in Equation 3, which expresses a proposed movement from L_i to L_{i+1}. The acceptance or rejection of the movement is based on the Metropolis-Hastings acceptance rule, as shown in Equation 4.

$$A(L_i, L_{i+1}) = \min\{1, \frac{p(L_{i+1})}{p(L_i)}\}, \tag{3}$$

$$p(\cdot) = \frac{1}{Z} \exp \frac{\Phi(\cdot) + f_c(\cdot)}{T}. \tag{4}$$

$\Phi(\cdot)$ is defined in Section 5. $f_c(\cdot)$ is the collision detection since we never want patterns colliding with each other or patterns touching the walls. $f_c(\cdot)$ returns 0 if no collision and returns $-\infty$, otherwise. In other words, we reject a movement if it leads to one or more collisions. Z is a constant ensuring the probability $\in [0, 1]$. The T is the temperature in the annealing process. We define it as $T = \max(\eta, i/\mathbf{t})$, i.e., the sampler aggressively explores various potential results near the beginning, given a large value of T. However, the optimization tends to refine the results near the end during T decreasing according to round i. We empirically set $\eta = 0.5, \mathbf{t} = 1000$. We use the following strategies to introduce a proposed movement. For each iteration, a random strategy is selected:

Adding/Removing Patterns. According to the constraints, if a room is too spacious, we should try adding more patterns. In contrast, some existent patterns should be removed if a room is too crowded. Thus, adding/removing patterns is one of the strategies for proposing movements. The chance of adding or removing patterns decreases with the increase of the number of patterns existing in the scene. Specifically, this probability is proportional to $(1 - f_u(\cdot) - f_e(\cdot))/Z_a$, where Z_a is a normalization factor, i.e, if the scene is already filled up with different patterns and elastic spaces, the system has less chance of inserting new patterns or removing existent patterns. Manually selecting patterns is also feasible. For user-selected patterns, our method will optimize the scene only using these selected patterns without adding or removing patterns.

Translating/Rotating Patterns. Patterns are also iteratively translated/rotated so that the relative pathways among them are appropriate and collisions are avoided. The centers of them conduct translations of patterns. For example, the center of a linear pattern is the middle point of the two endpoints, the center of a radial pattern is the start point of all of its branches, etc. To generate

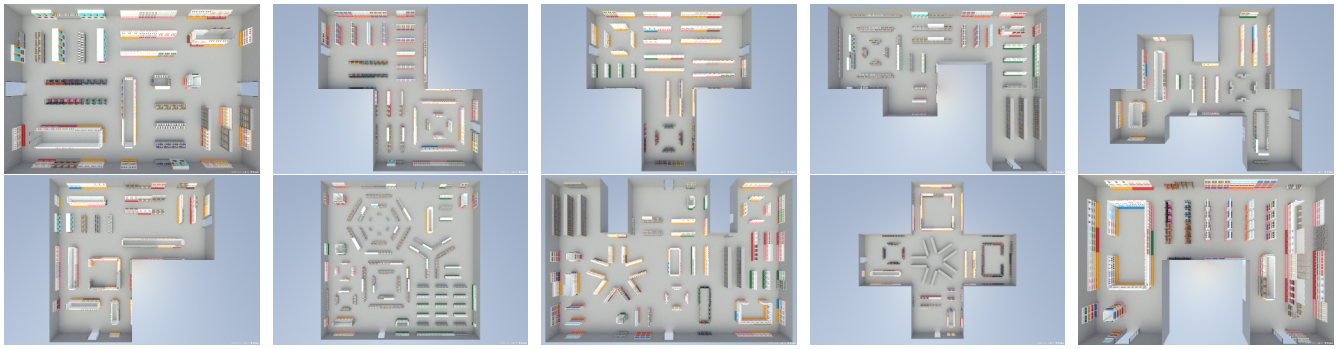


Figure 6: Results of generated commercial spaces (markets/shops).

well-aligned scenes, we disable arbitrary orientations of patterns. The proposed rotation must make a local area inside the pattern parallel with at least one wall.

Rescaling Patterns. This refers to tuning parameters of involved patterns, e.g., widening/shrinking cross patterns by Gaussian noises or adding/removing a branch in a radial pattern.

Swapping Patterns. In many cases, a pattern cannot satisfy the constraints no matter how we tune its parameters. Thus, a proposed movement can directly swap an inappropriate pattern with another pattern, e.g., changing an existing circular pattern to a new cross pattern. The parameters are inherited according to the size of the pattern before swapping.

An entire iteratively synthetic process is shown in a supplementary video. After this optimization converges, patterns are fixed. Objects are organized next to each other, following the patterns, as shown in Figure 3. Each pattern is assigned a category containing a set of objects. For example, a pattern assigned with "Fruit" sells apples, pears, pitayas, grapes, etc. Additionally, we notice that the walls of the room are also utilized. The segments of walls are assigned to their nearest patterns, thus being merged into the patterns, e.g., adding a new row against the wall to the cross pattern.

7 EXPERIMENTS

7.1 Results and Setup

Figure 6 shows several representative results of our method. The by-product of our method is the longest path inside each generated scene, which refers to the roundness in Section 5. Thus, as shown in Figure 7, our method can also suggest potential routines in supermarkets, e.g., for shopping or sightseeing. Our method is implemented on a CPU with ten cores (3.0GHz). On average, each scene consumes a minute to be generated with a maximum iteration of 20,000, where parallel computing is leveraged.

As shown in Figure 11, We developed a web-based platform to collect a set of elaborated and well-designed market-like scenes efficiently. The 3D models are collected using the Unity assets store, including pitaya, meat, shelves, etc. The models are organized based on their categories to be easily retrieved and added by users. Users could translate, rotate or re-scale objects by a transform controller, cursor movements or a coordinate setter panel. Room shapes are editable. Most importantly, a moderately sized market typically contains nearly 300 shelves with thousands of commodity objects.

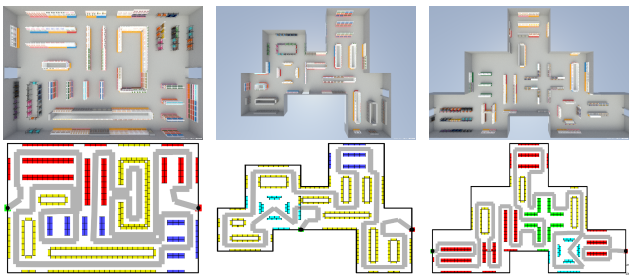


Figure 7: The navigation generated by our method. The top row refers to the generated scenes and the bottom row refers to the navigable routes of the roundness.

Thus, level of details is available to alleviate the GPU consumption, as shown in Figure 11c. Duplicated objects can be aligned with the original objects by pressing a key on a keyboard. Third, users can edit objects in groups by raycasting or box selections. In sum, 3 professional interior designers were recruited to craft more than 50 scenes.

7.2 Aesthetics and Plausibility

This section measures the aesthetics and plausibility of our method, i.e., the generated scenes should be as aesthetic and plausible as possible. Thus, we conduct a user study to quantitatively measure the results of our method. The top-down views are used to simulate macro-perspectives. 50 participants were invited to evaluate the generated scenes and the manually crafted scenes. The participants comprised university students, office workers, freelancers, designers, etc. They were only presented a series of questions without being told how the scenes were generated. Each question contained a synthesized scene and a manually crafted scene. The participants were asked to compare the two presented scenes and mark them respectively on a 5-point Likert scale (from 0: extremely poorly arranged and implausible to 5: totally aesthetic and plausible).

The questions were dynamically generated for each questionnaire, and rendered scenes presented to the subjects in each question were also shuffled, i.e., questions and scenes are presented in random order. Figure 12 shows the questionnaire platform for conducting this user study.

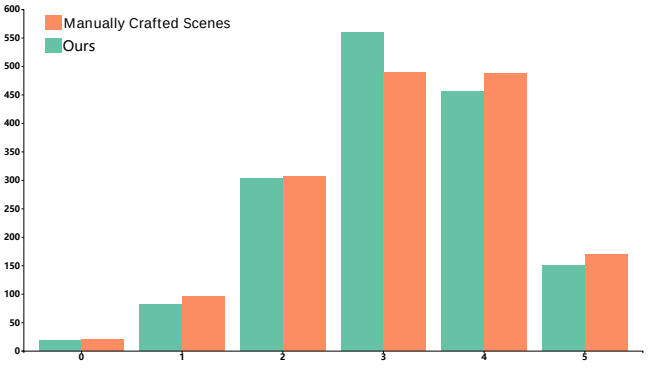


Figure 8: The results of the user study on aesthetic and plausibility. From the distributions, most judgements gave 3 or 4, since most scenes were basically satisfied. Some scenes were highly favored by subjects, thus achieving 5. In contrast, a small number of scenes were not acceptable. For each single mark, the numbers of times of selecting the mark do not vary much, for ours and manual scenes.

The result shows an average of 3.149 of our method, with a standard deviation 1.079. and the 3.171 with 1.126 for hand-crafted scenes. Figure 8 further shows the distribution of the scores respectively for our method and designers, where the Y-axis refers to numbers of times of scores selected by subjects, and the X-axis refers to the scores (0-5). From the distribution, most scenes are satisfied by the subjects (scores 4-5), while the rest of the scenes are highly preferred (score 5) or disliked (scores 1-2). If we look at each unique score, the number of times does not vary much between ours and manual scenes. For the relatively worse and medium results judged by the users, our method performs a slightly higher lower bound. In sum, We evaluated 1500 generated scenes, and 1101 achieved 3 or higher marks, so the rate of plausible scenes is 73.40%. Eventually, a Kruskal-Wallis Test shows that the p-value equals 0.413, which implies that the differences are not statistically significant. Therefore, we could conclude that our method generated similar commercial layouts compared to manual solutions. We also notice that the best results judged by the subjects contain slightly more crafted scenes, implying that the upper limit of our method can be improved in the future.

However, our automatic method is much faster in generating a large number of scenes. Our method takes only 10 seconds to generate a scene, while it takes a designer about 30-40 minutes to manually craft a scene. More discussions on diversity and efficiency are in Sections B and C.

7.3 Efficiency and Effectiveness

It is no better to invite participants to explore scenes immersively. Therefore, we designed a user study to measure how a scene could reveal as many commodities to customers as possible and save as much time for customers buying commodities as possible. Firstly, another VR-oriented client is developed to support this user study, as shown in Figure 13, where participants wear Vive head-mounted displays (HMD) to observe 3D scenes and use controllers to interact with scenes. The client renders scenes with a first-person view to

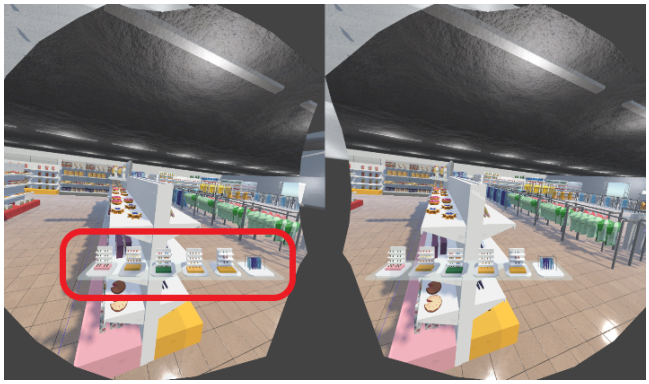


Figure 9: The menu of the first-person view in VR (Red Box), prompting potential objects to be obtained.

mimic a real-world shopping experience. Participants could move by pressing keys on the controllers. Base stations are also available to locate participants and enable them to wander the scenes in small ranges. The VR-oriented client is implemented with Unity and connected to the web-based platform.

This user study invites subjects to play two roles separately: a customer wishing to buy a set of commodities urgently and another customer aimlessly wandering the scene. The former simulates daily shopping, where people go to a market and quickly buy what they want. Under the former assumption, a subject is randomly assigned with several objects, e.g., buying beers, a t-shirt, a bottle of liquid detergent, sandwiches, lobsters, and pet foods. The objects are randomly selected from the available object pool by ensuring that the assigned objects are uniformly distributed w.r.t the categories. In case of high variations, the number of assigned objects is 6 for all subjects. Subsequently, we count the time consumed from entering the market (through the entrance) to finding all assigned objects to leave the market (through the exit). The latter simulates leisure activities where people take a walk or kill time, so the more commercial objects are revealed to them, the more immediate consumption is likely to happen. Under the latter assumption, a limited time is assigned to each subject, and they freely "walk" around in scenes. Subsequently, we count the number of viewed objects during the free activity. In general, the first role measures how efficient a market-like place is and the second role measures how a market-like place could show as many commodities as possible.

The raycasting from the Vive HMD is assembled for catching objects viewed by subjects. For each role, a generated scene and a manually crafted scene by the professional designers in Section 7.1 are randomly assigned, i.e., each role for a subject is conducted twice in two scenes. We ensure that half of the subjects experience hand-crafted scenes first and half experience generated scenes first. We also ensure that the two scenes assigned to each subject are in the same room shape and contain similar numbers (around 200) of shelves/counters. 30 subjects are newly recruited. The subjects are university students, office workers, freelancers, designers, etc. All the subjects follow a "pre-experiment" where they can freely experience the two roles, wander scenes and interact with objects until they felt comfortable with the way to interact with the virtual world

Table 1: Evaluating commercial values. The "Efficency" column refers to the first role, where we measure time consumption in seconds. The "Effectiveness" column refers to the second role, where we count the numbers of objects. Each cell has an averaged value with a stadard deviation in a bracket.

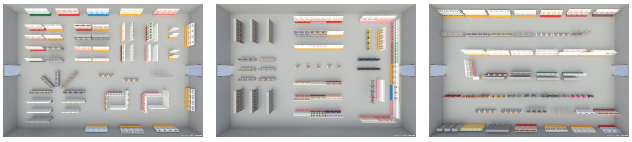
Method	Efficiency (seconds)	Effectiveness (#objects)
Crafted	218.317 (54.164)	81.233 (26.778)
Generated	224.297 (85.131)	80.7 (32.81)

without being dizzy. All the subjects report being comfortable and directional within 20 minutes. In Role 1, each subject is presented with individual objects to be found before entering scenes. A menu is also prompted in Role 1, as shown in Figure 9. Objects to be found are listed in the menu. When a subject successfully finds an object from the menu, the object is removed from the menu accordingly, thus showing the remaining objects to be found. The menu is assembled in case some subjects forget what to find halfway through the user study, potentially adding more variances to the results. During the experiments, technical staff were available in case of any technical questions.

Table 1 shows the results of this user study. The "Efficiency" and "Effectiveness" correspond to the two roles. The "Efficiency" is recorded in seconds and "Effectiveness" refers to the number of objects seen by the subjects. Each cell contains an average value of seconds or numbers and a standard deviation in a bracket. For both roles, the scenes automatically generated by our method achieve comparable results w.r.t the scenes manually crafted by human designers. Thus, we can conclude the contribution to generating efficient and immersive commercial layouts. We observe that the standard deviations are relatively large. This is possible because of the following two reasons. First, the behaviours of subjects are quite different. Some subjects are familiar with Vive HMD and controllers before the user study. Some of them react fast and move fast. In contrast, other subjects are beginners to the Vive HMD and controllers. They may stop halfway through routines and look around. Sometimes, they may miss targets. Second, as an automatic approach for scene synthesis, ours is nevertheless not as excellent as professional designers since a few manually crafted scenes show attractive and humanistic designs. Section 7.2 further conducts another user study to measure the overall designs, thus illustrating the distribution of excellent results from both settings.

8 CONCLUSION

This paper presents an approach that synthesizes commercial scenes by iteratively optimizing decomposed patterns, leveraging constraints formulated by interior design rules. In the future, more consideration should be given to generating more plausible commercial layouts. First, as shown in Figure 10a, more variations of patterns could be explored. For example, a circular pattern can be sheared to add more routines. We should also consider the utilization of the interior space in solid patterns, e.g., the circular pattern in Fig.2d. Similarly, a cross pattern can be partially radial. Second, in Figure 10b, patterns can be considered "sub-patterns" embedded



(a) More Variations (b) Embedded Patterns (c) Serpentine

Figure 10: Future directions of our method, where the scenes are crafted by the professional designers in Section 7.1. (a) more variations of patterns, e.g., shearing circular patterns to add more potential routines. (b) automatic embedding patterns with each other to consume less space. (c) a market with a super-long pattern.

in other patterns so that more fragmented spaces are utilized. Third, Figure 10c shows a very interesting scene crafted. We recognize it as a single pattern filling the room. How to generate this particular case is also worth researching in the future.

Second, this paper measures the constraints of traffic flows ideally, e.g., a person can go through a path without being blocked by others. If two customers meet, they could squeeze up a little or swap to other routines in real-world scenarios. These require further investigation into how the surrounding traffic flows influence human shopping behaviours. Possibly we could combine the crowd-driven constraints into the synthetic process [8, 38].

Third, a commercial scene is open, i.e., customers can enter it from all directions ignoring all the walls. For example, the scenes are still functional if we remove walls or add/swap entrances and exits in our results. However, this may violate the roundness calculation since the roundness is calculated given a starting point and an ending point. In the future, we could also investigate the potential influence and how algorithms dynamically decide the entrances and exits. Our method recognizes each group of objects as every unit, so it is hard to generalize to residential scenes.

Our object-insertion strategy hierarchizes objects into different categories, e.g., cloth counters, frozen foods, etc, and subsequently assigns a category to each pattern. However, this does not take relations between categories into account, e.g., a reasonable market does not place meats and clothes together. Furthermore, some objects in a specific category are not required to be placed a lot, e.g., pet foods or vending machines, because those objects are not frequently used or bought. Thus, an improvement could be applying constraints and relationships between objects.

Our method may optimize a scene to a local minimum. It is hard to judge whether a scene gets into a local minimum by its results since a local minimum may yield a good or bad result. Our method generates multiple scenes concerning every floor plan, which could alleviate the local minimum problem. Future work may also enhance the bad results derived from local minimums.

ACKNOWLEDGMENTS

This work was supported by the Natural Science Foundation of China (Project Number 61832016) and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

REFERENCES

[1] Ahmed Alawadhi and So-Yeon Yoon. 2016. Shopping behavioral intentions contributed by store layout and perceived crowding: An exploratory study using computer walk–through simulation. *Journal of Interior Design* 41, 4 (2016), 29–46.

[2] Pierre Chandon, J Wesley Hutchinson, Eric T Bradlow, and Scott H Young. 2009. Does in-store marketing work? Effects of the number and position of shelf facings on brand attention and evaluation at the point of purchase. *Journal of marketing* 73, 6 (2009), 1–17.

[3] Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D Manning. 2015. Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289* (2015).

[4] Kang Chen, Yukun Lai, Yu-Xin Wu, Ralph Robert Martin, and Shi-Min Hu. 2014. Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information. *ACM Transactions on Graphics* 33, 6 (2014).

[5] Kang Chen, Yu-Kun Lai, and Shi-Min Hu. 2015. 3D indoor scene modeling from RGB-D data: a survey. *Computational Visual Media* 1, 4 (2015), 267–278.

[6] Claus Ebster. 2011. *Store design and visual merchandising: Creating store space that encourages buying*. Business Expert Press.

[7] Reid Ewing and Bartholomew Keith. 2013. *Pedestrian-& Trasit-Oriented Design*. Urban Land Institute and American Planning Association.

[8] Tian Feng, Lap-Fai Yu, Sai-Kit Yeung, KangKang Yin, and Kun Zhou. 2016. Crowd-driven mid-scale layout design. *ACM Trans. Graph.* 35, 4 (2016), 132–1.

[9] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. 2012. Example-based synthesis of 3D object arrangements. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 135.

[10] Matthew Fisher, Manolis Savva, Yangyan Li, Pat Hanrahan, and Matthias Nießner. 2015. Activity-centric scene synthesis for functional 3D scene modeling. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 179.

[11] Qiang Fu, Xiaowu Chen, Xiaotian Wang, Sijia Wen, Bin Zhou, and Hongbo Fu. 2017. Adaptive synthesis of indoor scenes via activity-associated object relation graphs. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–13.

[12] Yu He, Ying-Tian Liu, Yi-Han Jin, Song-Hai Zhang, Yu-Kun Lai, and Shi-Min Hu. 2021. Context-consistent generation of indoor virtual environments based on geometry constraints. *IEEE Transactions on Visualization and Computer Graphics* 28, 12 (2021), 3986–3999.

[13] Urban Land Institute. 1999. *Shopping Center Development Handbook*. Intellectual Property Publishing House and China Water & Power Press.

[14] kujiale.com. 2020. Kujiale. Retrieved Dec 8, 2020 from https://www.kujiale.com/

[15] Michael Levy, Barton A Weitz, and Pandit Ajay. 2009. Retailing Management, New York: The McGraw-Hills/Irwin Companies.

[16] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. 2019. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)* 38, 2 (2019), 1–16.

[17] Wei Liang, Jingjing Liu, Yining Lang, Bing Ning, and Lap-Fai Yu. 2019. Functional workspace optimization via learning personal preferences from virtual experiences. *IEEE transactions on visualization and computer graphics* 25, 5 (2019), 1836–1845.

[18] Yuan Liang, Song-Hai Zhang, and Ralph Robert Martin. 2017. Automatic data-driven room design generation. In *International Workshop on Next Generation Computer Animation Techniques*. Springer, 133–148.

[19] Hong-Li Lu. 2010. *Residential Interior Design*. Liaoning Science and Technology Publishing House.

[20] Andrew Luo, Zhoutong Zhang, Jiajun Wu, and Joshua B Tenenbaum. 2020. End-to-End Optimization of Scene Layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3754–3763.

[21] Rui Ma, Akshay Gadi Patil, Matthew Fisher, Manyi Li, Sören Pirk, Binh-Son Hua, Sai-Kit Yeung, Xin Tong, Leonidas Guibas, and Hao Zhang. 2018. Language-driven synthesis of 3D scenes from scene databases. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 212.

[22] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. 2011. Interactive furniture layout using interior design guidelines. *ACM transactions on graphics (TOG)* 30, 4 (2011), 87.

[23] Farshid Moussavi. 2015. *The Function of Style*. Harvard Graduate School of Design and ActarD Inc.

[24] Yuushi Narazaki. 1985. *Shop Design*. Ohmsha, Ltd. and Science Press.

[25] planner5d.com. 2020. Planner5d. Retrieved Dec 8, 2020 from https://planner5d.com/

[26] Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. 2018. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5899–5908.

[27] Daniel Ritchie, Kai Wang, and Yu-an Lin. 2019. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6182–6190.

[28] Manolis Savva, Angel X Chang, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. 2014. SceneGrok: Inferring action maps in 3D environments. *ACM transactions on graphics (TOG)* 33, 6 (2014), 1–10.

[29] Manolis Savva, Angel X Chang, Pat Hanrahan, Matthew Fisher, and Matthias Nießner. 2016. Pigraphs: learning interaction snapshots from observations. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

[30] Alexander Schrijver et al. 2003. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer.

[31] Daniel Schulz. 2014. *Shopping Centers Planning & Design*. Design Media Publishing Limited.

[32] Anna Bonarou Vasiliki Asaroglou. 2013. *Furniture arrangement: in Residential spaces*. CreateSpace Independent Publishing Platform.

[33] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X Chang, and Daniel Ritchie. 2019. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 132.

[34] Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. 2018. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 70.

[35] Tomer Weiss, Alan Litteneker, Noah Duncan, Masaki Nakada, Chenfanfu Jiang, Lap-Fai Yu, and Demetri Terzopoulos. 2018. Fast and scalable position-based layout synthesis. *IEEE Transactions on Visualization and Computer Graphics* 25, 12 (2018), 3231–3243.

[36] Guoming Xiong, Qiang Fu, Hongbo Fu, Bin Zhou, Guoliang Luo, and Zhigang Deng. 2020. Motion planning for convertible indoor scene layout design. *IEEE Transactions on Visualization and Computer Graphics* 27, 12 (2020), 4413–4424.

[37] Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. 2013. Sketch2Scene: sketch-based co-retrieval and co-placement of 3D models. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 123.

[38] Shanwen Yang, Tianrui Li, Xun Gong, Bo Peng, and Jie Hu. 2020. A review on crowd simulation and modeling. *Graphical Models* 111 (2020), 101081. https://doi.org/10.1016/j.gmod.2020.101081

[39] Hongwei Yi, Chun-Hao P Huang, Dimitrios Tzionas, Muhammed Kocabas, Mohamed Hassan, Siyu Tang, Justus Thies, and Michael J Black. 2022. Human-aware object placement for visual environment reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3959–3970.

[40] Lap-Fai Yu, Sai Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley Osher. 2011. Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph.* 30, 4 (2011), 86.

[41] Suiyun Zhang, Zhizhong Han, Yu-Kun Lai, Matthias Zwicker, and Hui Zhang. 2019. Active arrangement of small objects in 3D indoor scenes. *IEEE transactions on visualization and computer graphics* 27, 4 (2019), 2250–2264.

[42] Suiyun Zhang, Zhizhong Han, and Hui Zhang. 2016. User guided 3D scene enrichment. In *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry*. 353–362.

[43] Song-Hai Zhang, Shao-Kui Zhang, Wei-Yu Xie, Cheng-Yang Luo, Yong-Liang Yang, and Hongbo Fu. 2021. Fast 3d indoor scene synthesis by learning spatial relation priors of objects. *IEEE Transactions on Visualization and Computer Graphics* 28, 9 (2021), 3082–3092.

[44] Shao-Kui Zhang, Yi-Xiao Li, Yu He, Yong-Liang Yang, and Song-Hai Zhang. 2021. MageAdd: Real-Time Interaction Simulation for Scene Synthesis. In *Proceedings of the 29th ACM International Conference on Multimedia*. 965–973.

[45] Shao-Kui Zhang, Hou Tam, Yike Li, Ke-Xin Ren, Hongbo Fu, and Song-Hai Zhang. 2023. SceneDirector: Interactive Scene Synthesis by Simultaneously Editing Multiple Objects in Real-Time. *IEEE Transactions on Visualization and Computer Graphics* (2023).

[46] Shao-Kui Zhang, Hou Tam, Yi-Xiao Li, Tai-Jiang Mu, and Song-Hai Zhang. 2022. SceneViewer: Automating Residential Photography in Virtual Environments. *IEEE Transactions on Visualization and Computer Graphics* (2022).

[47] Shao-Kui Zhang, Wei-Yu Xie, and Song-Hai Zhang. 2021. Geometry-Based Layout Generation with Hyper-Relations AMONG Objects. *Graphical Models* 116 (2021), 101104. https://doi.org/10.1016/j.gmod.2021.101104

[48] Yongqi Zhang, Haikun Huang, Erion Plaku, and Lap-Fai Yu. 2021. Joint computational design of workspaces and workplans. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–16.

[49] Zhang Zhang. 2013. *Best traffic flow designs of shopping center*. Phoenix Science Press.

[50] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. 2020. Deep generative modeling for scene synthesis via hybrid representations. *ACM Transactions on Graphics (TOG)* 39, 2 (2020), 1–21.

[51] Tao-Kai Zheng. 2011. *Interior Design For Home*. Huazhong University of Science & Technology Press.

A CALCULATING ROUNDNESS

Let $G = (V, E)$ be an undirected graph to be constructed for computing $f_r(L_i)$. Each vertex $v \in V$ owns an attribute $p_v = (x, y)$, which represents the position of the vertex in a scene. There is also an attribute d_e for each edge $e = (u, v) \in E \quad (u, v \in V)$, which stands for the length of e. In our context, an edge (u, v) maps a straight path between two points p_v and p_u in the scene through which customers travel. As a result, d_e could be directly calculated as the Euclidean distance between vertices u and v, formulated as:

$$d_e = ||p_u - p_v||_2. \tag{5}$$

Then the key lies in deciding every vertex $v \in V$ with its p_v and every edge $e = (u, v) \in E$ for the scene.

Normally, taking the "width" of an edge into account would be unnecessary, but that is not the case for an actual pathway in a traffic flow. Any path between two points must have a minimum width of W_m to satisfy the continuous exploration. To achieve this, two shelves split by an edge in G must yield a distance no smaller than W_m. When the path is represented by a centered-aligned line segment, any shelf by its sides keeps a distance no shorter than $\frac{W_m}{2}$ from it. Figure 4a of the main paper illustrates the distance $\frac{W_m}{2}$ kept by patterns.

We begin to construct G by constructing a subgraph $G_l = (V_l, E_l)$ for every pattern l in the room. All patterns are designed to be surrounded by several paths, considering the exterior part of G_l. The exterior vertices are connected linearly, thus forming a polygon-shaped ring that bounds a pattern. Generally, the polygon is in a similar shape to the pattern's tight bounding box, but is enlarged due to the $\frac{W_m}{2}$ distance constraints. As for the interior part, the vertices and edges are added following the pattern's properties. For example, a radial pattern is expressed by a center vertex that connects the exterior vertices through multiple edges. A grid pattern may embody a series of interior vertices connected adjacently. For each pattern l, G_l is a union of all vertices and edges in the interior and the exterior. Then G is comprised of the vertices and edges of all patterns, including v_{in} and v_{out}.

Two exterior edges from different patterns may be close to or even overlap with each other when they are put into an actual room. We notice that only one actual route instead of two should be recognized. Supposing $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ are two edges with actual line segments (p_{u_1}, p_{v_1}) and (p_{u_2}, p_{v_2}) on lines $\Upsilon_1 : a_1x + b_1y = c_1$ and $\Upsilon_2 : a_2x + b_2y = c_2$, respectively. For simplicity, the two edges can be merged only when: (i) $a_1b_2 - b_1a_2 = 0$, which means they are parallel; (ii) the distance:

$$d_e = \frac{|c_1 - c_2|}{\sqrt{{a_1}^2 + {b_1}^2}} \leq D_m; \tag{6}$$

(iii) when projected on any line parallel to them, they share an overlap part. Among the conditions above, D_m typically equals $\frac{W_n}{2}$, which is a threshold small enough to merge edges. After a single step of merging, the two original edges become three new edges embodied by actual line segments (p_{i_1}, p_{i_2}), (p_{i_2}, p_{i_3}), and (p_{i_3}, p_{i_4}), with the segment (p_{i_2}, p_{i_3}) in the middle originated from the previously overlapped part. They both lay on the same line L:

$$a_1x + b_1y = \frac{d_{e_1}c_1 + d_{e_2}c_2}{d_{e_1} + d_{e_2}}, \tag{7}$$

which represents a continuous route (p_{i_1}, p_{i_4}) between the original patterns.

After a series of merging operations, the original graph could change a lot. The attributes of vertices and edges should be modified accordingly, while different patterns may now share a common exterior consisting of various vertices connected as a new circuit.

When there is more than one patterns in the room, G could be interpreted as two points with one or several isolated rings. We need to selectively connect them to construct G. For the connection between a point p and a ring r, we project p on r and determine the projection point p_r. Then we add a new edge with segment (p, p_r). However, it is valid only when $||p - p_r||_2 <= D_c$ and (p, p_r) does not intersect with other rings. For the connection between two rings r_1 and r_2, we find two points p_1 and p_2 on them with the minimum distance and add an edge with segment (p_1, p_2). The undirected graph G is eventually completed.

$G = (V, E)$ may not be connected in some cases, such as no pattern exists or several patterns are isolated, where the roundness $f_r(L_i)$ is set to zero. Otherwise, $f_r(L_i)$ is calculated. We consider G a weighted tree, with each edge having a weight equal its attribute d_e. Then we have:

$$f_r(L_i) = \frac{\Sigma_{e_l \in \chi} d_{e_l}}{\Sigma_{e \in \chi} d_e}, \tag{8}$$

where χ is the set of all edges in the longest path. Every vertex is visited less than once, so no circuit exists in the longest path. Since G may not be acyclic, the longest-path problem is NP-complete, so we use an approximation to compute it.

Since G may not be acyclic, the longest-path problem is in NP-complete. Therefore, we use an approximation method to compute the longest path as follows:

(i) Compute the maximum spanning tree $T_1(G)$. We then use it to determine the only path $S = \{v_{in}, v_1, ...v_k, v_{out}\}$ that connects v_{in} and v_{out}. For simplicity, v_{in} is named as v_0, and v_{out} as v_{k+1}. $T_1(G)$ is discarded since we do not need it in the following steps.

(ii) Iterate i for the range from 0 to k. For each i, we iterate all neighbors of v_i and v_{i+1} and check if there exists a path $S_i = \{v_i, v_{n_i}, v_{j_1}, v_{j_2}..., v_{j_l}, v_{n_{i+1}}, v_{i+1}\}$ in G, where $(v_i, v_{n_i}), (v_{n_{i+1}}, v_{i+1}) \in E(G)$, and $\{v_{n_i}, v_{j_1}, v_{j_2}..., v_{j_l}, v_{n_{i+1}}\}$ is a path in $G - S$. Based on triangular inequality, if S_i exists, it is certainly longer than $\{v_i, v_{i+1}\}$. We then substitute the old one in S and form a new path S',which becomes the new S. Then we continue the iteration from v_{i+1}.

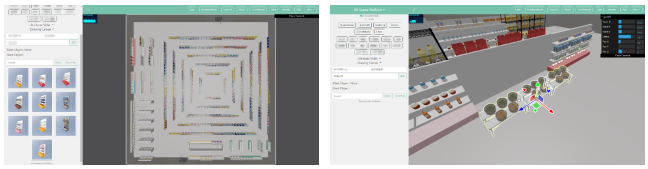
(iii) If the iteration in the former step does not succeed in discovering a replacing path even for once, we continue for step (iv). Otherwise, we perform another iteration for step (ii).

(iv) Compute the maximum spanning tree $T(G)$ that contains S. S stays unchanged in this step.

(v) Iterate e for every $e \in E(G) - \chi$ where $\chi = \{(v_0, v_1), (v_1, v_2), ..., (v_k, v_{k+1})\}$ consists of the edges derived from S. For each e, adding it into $T(G)$ would result in one and only one circuit $C = \{(v_i, v_{i+1}), (v_{i+1}, v_{i+2}), ...(v_{i+j-1}, v_{i+j}), (v_{i+j}, v_{p_1}), (v_{p_1}, v_{p_2}), ...e, ..., (v_{p_q}, v_i)\}$. We then compare the total path length for both $C_1 = \{(v_i, v_{i+1}), (v_{i+1}, v_{i+2}), ...(v_{i+j-1}, v_{i+j})\}$ and $C_2 = \{(v_{i+j}, v_{p_1}), (v_{p_1}, v_{p_2}), ...e, ..., (v_{p_q}, v_i)\}$. If C_2 is longer, then we find a substitution for C_1 in S. e is thus added to $T(G)$, and the shortest edge in C_1 is removed to ensure that $T(G)$ is still a tree. S is changed due to the substitution. On finishing the actions, we break the iteration.

Table 2: The efficiency of our method concerning different sizes of rooms. Times are recorded as seconds.

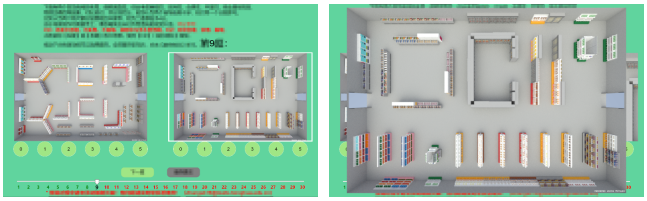
	10m	15m	20m	25m	30m
10m	106				
15m	126	163			
20m	154	187	261		
25m		280	366	423	
30m			441	531	588
40m				669	745
50m					1042



(a) Searching Objects. (b) Transforming Objects.

(c) Level of Details. (d) Box Selections.

Figure 11: The web-based platform for manually designing large-scale commercial layouts, where we can manipulate objects just like using similar industrial applications such as Kujiale [14] or Plannar5d [25]. The "level of details" and "box selections" further enable efficient interactions with a large scale of objects.



(a) (b)

Figure 12: (a): The platform for the user study on the aesthetic and plausibility, where we could answer questions, temporarily save answers and jump to other questions. (b): Users can also optionally zoom in each presented scene.

(a) Fresh Foods (b) Clothes (c) Snacks

Figure 13: The VR-oriented client for conducting the immersive user study to measure the scenes.

(*vi*) If the iteration in (*v*) was ended by a successful exchange, we jump to (*v*) for another iteration. Otherwise, the longest path S is determined, and we output the result.

B DIVERSITY

We generated 1000 scenes and counted the patterns distributed in them. In general, 43.2% of the scenes have incorporated every proposed pattern. 99.0% of the scenes have incorporated three types of patterns. 9.4% of the scenes are dominated by a specific pattern, i.e., a pattern occupies more than 30% of the scene area. For example, a shop is mainly arranged with a cross pattern. The overall assembling rates of the four patterns in all scenes are 29.86% (Linearity), 17.68% (Cross), 19.40% (Circulation), and 33.06% (Radiation), respectively. The assembling rates are counted over areas since a pattern type may have a small number but occupy large areas.

C EFFICIENCY

We ran our method on different sizes of floor plans. The average cost times are shown in Table 2. Each cell refers to a time consumption value in seconds concerning a floor plan size. For example, the cost time for size 20m*20m is 261s. This experiment uses a single core to clarify the time consumption. When leveraging multiple cores, using a ten-core processor, our method achieves ten results given a similar time.