The Visual Computer (2024) 40:6319–6339
https://doi.org/10.1007/s00371-023-03167-4

ORIGINAL ARTICLE

Real-scene-constrained virtual scene layout synthesis for mixed reality

Runze Fan[1] · Lili Wang[1,2] · Xinda Liu[1] · Sio Kei Im[3] · Chan Tong Lam[3]

Accepted: 25 August 2023 / Published online: 12 December 2023

Abstract
Given a real source scene and a virtual target scene, the real-scene-constrained virtual scene layout synthesis problem is defined as how to re-synthesize the layout of the virtual furniture in the virtual scene to form a new virtual scene such that the new scene not only looks similar to the input real and virtual scenes but also is interactive. The goal of this problem is to maximize interactivity and fidelity which are contradictory. To solve this problem, we propose a real-scene-constrained virtual scene layout synthesis method to synthesize the layout of the virtual furniture in the new virtual scene. We split the scene layout synthesis process into 3 interrelated steps: scene matching, matched furniture layout generating, and unmatched furniture layout generating. For scene matching, we propose a deep scene matching network to predict the matching relationship between real and virtual furniture. For matched furniture layout generating, we propose a layout parameters optimization algorithm to predict suitable layouts of the matched virtual furniture. For unmatched furniture layout generating, we propose a deep scene generating network to predict suitable layouts of unmatched virtual furniture. We evaluate the quality of our method to synthesize scenes of different kinds and sizes. The results show that, compared with the heuristic rules-based method, our method has better matching accuracy and location accuracy. We also design a user study to evaluate the interactivity and fidelity. Compared to the manual method and the heuristic rules-based method, our method has a significant improvement in interactivity and fidelity.

Keywords Scene layout synthesis · Scene matching · Scene generating

1 Introduction

Mixed reality combines virtual scenes and real scenes to provide users with a more creative and immersive experience. For example, if the user is in the conference room and wears an augmented reality head-mounted display, the virtual-real fusion scene observed is the sightseeing room. The user can touch the real table in front of the user with the immersion that the user is torching the virtual table, and the user can sit on the real chair with the immersion that the user is sitting on the virtual sofa. In order to realize the above idea, the problem we need to solve is: given a real scene and a virtual scene, based on the constraints of the real scene, how to rearrange the virtual furniture in the virtual scene to form a new virtual scene, and then integrate the new virtual scene with the real scene, and provides the user with the experience of being immersed in a virtual scene similar to the input, both visually and tactilely. We refer to this problem as a real-scene-constrained virtual scene layout synthesis problem.

The virtual scene layout synthesis problem is similar to the scene retargeting problem, which aims at adjusting input virtual scenes into arbitrary sizes while preserving the spatial structure. Different kinds of methods are proposed to solve the scene retargeting problem. One kind of method manipulates scenes on the 'inner level' by adding or removing the

✉ Lili Wang
wanglily@buaa.edu.cn

Runze Fan
by2106131@buaa.edu.cn

Xinda Liu
liuxinda@buaa.edu.cn

Sio Kei Im
marcusim@mpu.edu.mo

Chan Tong Lam
ctlam@mpu.edu.mo

[1] State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, XueYuan Road, Beijing 100191, Beijing, China

[2] Peng Cheng Laboratory, Xingke Road, Shenzhen 518055, Shenzhen, China

[3] Macao Polytechnic University, R. de Luís Gonzaga Gomes, Macao 999078, Macao, China

basic element of scenes [1, 2]. While the other kind manipulates scenes on the 'outer level' by adjusting the furniture size and the distance between them [3, 4]. Another relevant problem is the scene synthesis problem which aims at designing a new scene, and a series of algorithms are proposed to synthesize a virtual scene from scratch[5–7]. The main difference between scene layout synthesis, scene retargeting, and scene synthesis is that we have a real scene as input, which will provide the passive haptic for the synthesized virtual scenes. Take a virtual sightseeing room for example, the scene layout synthesis tries to rearrange the layout of the virtual furniture in the virtual scene based on the input real conference room to provide the user with a visual and tactile experience. The scene retargeting tries to resize the virtual furniture and the distance between them while maintaining the spatial structure according to the input scene target dimensions. The scene synthesis tries to generate a completely new virtual scene, only the category of furniture in the newly generated scene is the same as the input scene. The size of the newly generated furniture and the relationships between the furniture are independent of the input scene.

In this paper, we propose a real-scene-constrained virtual scene layout synthesis method to synthesize the layout of the virtual furniture in the new virtual scene. The layout maintains the relationship between furniture in the input virtual scene as much as possible, and when the user explores the synthesized virtual scene, the input real scene provides the user with as much passive haptic as possible. We focus on the main furniture in the scene without considering some tiny decorations. There are 3 main steps in our scene layout synthesis process: scene matching, matched furniture layout generating, and unmatched furniture layout generating. First, we introduce a deep scene matching network to predict the matching relationship between real furniture and virtual furniture. Then, we propose a layout parameters optimization algorithm to arrange the matched virtual furniture into the new virtual scene, i.e., predicting suitable layouts to match the real scene. At last, we propose a deep scene generating network to arrange the unmatched virtual furniture in the new virtual scene. We evaluate the quality of our method to synthesize scenes of different types and sizes. The results show our method increases the matching accuracy by 15% and improves the location accuracy by a factor of 5.71 in total, compared with the heuristic rules-based method [8]. We also design a user study to evaluate the interactivity and fidelity of our method. Compared to the manual method and the heuristic rules-based method, our method has a significant improvement in touchable time and class correctness. Figure 1 shows the results of scene layout synthesis using our method with a real conference room and a virtual sightseeing room as input.

In summary, our contributions are as follows:

1. A pipeline to synthesize the layout of the virtual furniture in a new virtual scene according to the input virtual furniture and its layout within the constraints of the input real furniture and its layout.

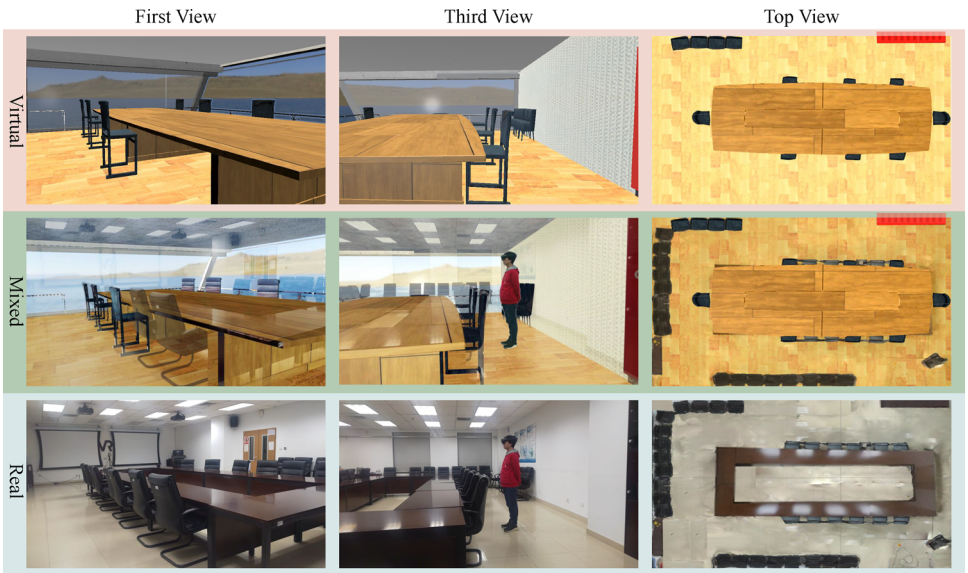


Fig. 1 Mixed reality offers users a more creative and immersive experience. Given a virtual sightseeing room (top row), the user wears an augmented reality head-mounted display in the conference room (bottom row), and the virtual scene and the real scene are mixed (middle row). The user can observe the virtual-real fusion scene (left column) through the HMD and touch the table and chairs in front of the user (middle column). The layouts of the furniture in the virtual scene are synthesized (right column), which not only maintains the relationship between furniture in the input virtual scene as much as possible but also provides the user with as much passive haptic as possible when he explores the synthesized virtual scene

2. A deep scene matching network to predict the matching relationship between real furniture and virtual furniture.

3. A deep scene generating network to arrange the unmatched virtual furniture in the new virtual scene.

2 Related work

2.1 Scene retargeting

Retargeting is an essential topic in computer graphics (CG) and computer vision (CV)[9]. In CV, image retargeting is defined as adjusting input images into arbitrary sizes and simultaneously while preserving the salient regions of the input images[10]. Similarly, 3D scene retargeting is defined as adjusting input scenes into arbitrary sizes and simultaneously while preserving the spatial structure of the input scenes[1]. Many methods have been proposed to retarget a 3D scene, and they can be divided into two categories, one kind manipulates the scene on the 'inner level' while the other manipulates on the 'outer level'. The 'inner level' method retargets the scene by adding or removing the basic component of the scene or the object. [1] first decomposes the input scene into a collection of components, then the constraint relationships between components are built, and retargeting is implied by inserting or removing these components without violating the constraints. Similarly, [2] first decomposes the input irregular 3D architecture into a set of sequences, and then retargets the architecture by replication and scaling the basic structural elements of the architecture while preserving their structures. The 'outer level' method retargets the scene by adjusting the objects' size and the distance between them. [3] constructs a perception-aware function to optimize the size, location of the objects, and distance between them with preserving structural/geometric properties. The redirecting problem in VR can be viewed as a variant of the scene retargeting problem, which tries to retarget a big virtual scene into a small real scene. A series of methods are proposed to solve this problem [4, 11], they either introduce geometric distortions to virtual objects or restructure the scene. These methods also belong to the 'outer level' methods of scene retargeting problem. However, there are many differences between the scene layout synthesis problem and the scene retargeting problem. First, no furniture in the target scene is allowed to be discarded in the scene layout synthesis problem. Second, in the scene layout synthesis problem, the source scene and the target scene differ not only in size but also in structure. Third, the scene layout synthesis problem requires making full use of the furniture in the real scene to maximize interactivity.

2.2 Scene synthesis

Scene synthesis is one of the basic tasks in CG, and there is a long line of work to solve this problem. Early studies first construct rule-based constraints based on prior knowledge, then expressed these constraints as cost functions, and synthesize the scene by optimizing the cost functions [6, 7, 12–15]. With the development of deep learning, learning-based methods become popular and get great performance[16–18]. [19] synthesizes the scene iteratively. It first represents the scene as a multiple channels top-down view image, then uses 4 different CNNs to predict 'whether', 'category', 'location', and 'instance', respectively, and synthesize the scene based on the predicted results. [5] uses a two-stage pipeline to synthesize scenes. It first represents the scene as a graph, then uses GNN to predict a series of actions which is used to guide the synthesis of the graph. With the synthesized graph, the scene is constructed by instantiating the node in the graph as furniture. [20] uses a variational autoencoder to synthesize the scene. [21] used the long-term dependencies to generate the dynamic scene graphs. Different from [5] and [19], [20] represents the scene as a hierarchy based on the spatial relations of the furniture in the scene. In the scene synthesis problem, the furniture to be synthesized is determined by the already synthesized scene. While in the scene layout synthesis problem, the furniture to be synthesized is deterministic. Besides, for the scene synthesis problem, the synthesizer only needs to care about the information of the already synthesized scene. While in the scene layout synthesis problem, the synthesis needs to pay attention to not only the information of the already synthesized scene but also the information of the target scene. Thus, the scene layout synthesis problem is more complicated than the scene synthesis problem.

2.3 Graph matching

The problem of graph matching under node and pairwise constraints is fundamental in areas as diverse as combinatorial optimization, machine learning, CG, and CV [22]. Graph matching refers to establishing node correspondences between two or multiple graphs, which have been long-studied. The graph matching problem can be divided into two general categories: exact matching and inexact matching [23]. Due to the graph matching can be formulated as a quadratic assignment, being well-known NP-complete, it is hard to predict the optimal solution[24]. Different methods are introduced to get the approximate solution of the graph matching problem [25–30]. With the development of deep learning, learning-based methods are widely used in various fields [31–35]. In CV, a lot of methods are applied

for graph matching on images. [22] uses a CNN to extract node features and constructs multiple different matrix layers to predict the matching result. [36] replaces the CNN used in [22] with GNN to better structural and geometric feature extraction. GNNs are deep learning-based methods that operate on the graph domain, and due to their convincing performance, GNNs have become a widely applied graph analysis method recently [37–46]. [47] proposed an accurate stereo matching method based on color segments and edges. In this paper, we combine GNNs with Sinkhorn Network[48] to predict the matching relationship between the furniture in the source scene and the target scene. The goal of this paper is not to propose a new learning-based graph matching method, but to propose a new scene graph matching network to solve the scene matching problem, which is a sub-problem of the scene layout synthesis problem.

3 Method

3.1 Pipeline

We denote the scene as S and the furniture in the scene as F. The subscript v denotes virtual, the subscript r denotes real, the superscript m denotes matched, the superscript u denotes unmatched, and the superscript $'$ denotes newly synthesized. Our goal is to synthesize a new virtual scene $S_v{}'$ while the layout of $S_v{}'$ maintains a similar relationship between furniture as in the input virtual scene S_v. At the same time, it also needs to maintain a similar relationship between furniture as in the input real scene S_r, which will help the users get the passive haptic from S_r when wandering in $S_v{}'$.

As shown in Fig. 2, there are 3 main steps in our scene layout synthesis process: (1) scene matching; (2) matched furniture layout generating; (3) unmatched furniture layout generating. First, we introduce the deep scene matching network to predict the matching matrix M between real furniture F_r in S_r and virtual furniture F_v in S_v. Then we propose a layout parameters optimization algorithm to arrange the matched virtual furniture F_v^m into $S_v{}'$, i.e., predicting suitable layouts of F_v^m to match the corresponding real furniture F_r^m. At last, we propose a deep scene generating network to arrange the unmatched virtual furniture F_v^u in $S_v{}'$, i.e., predicting suitable layouts of F_v^u in $S_v{}'$. In summary, we synthesize the layout of the furniture in the virtual scene on the 'outer' rather than the 'inner' level, i.e., adjusting the furniture's size and center. We do not insert or remove the components of furniture. We begin by dividing the furniture into matched and unmatched (step 1), and then adjusting their sizes and centers (steps 2 and 3).

In steps 1 and 2, the constraints that need to be considered are: (1) the matched F_v^m is similar to its corresponding F_r^m, i.e., the category c (defined in Table 1) of the F_v^m and that of the corresponding F_r^m is the same, the size s of the F_v^m and that of the corresponding F_r^m is similar ($\pm 50\%$); (2) the spatial and functional relationship (SFR) of matched virtual furniture SFR(F_v^m) is similar to that of its corresponding real furniture SFR(F_r^m). In step 3, the constraints that

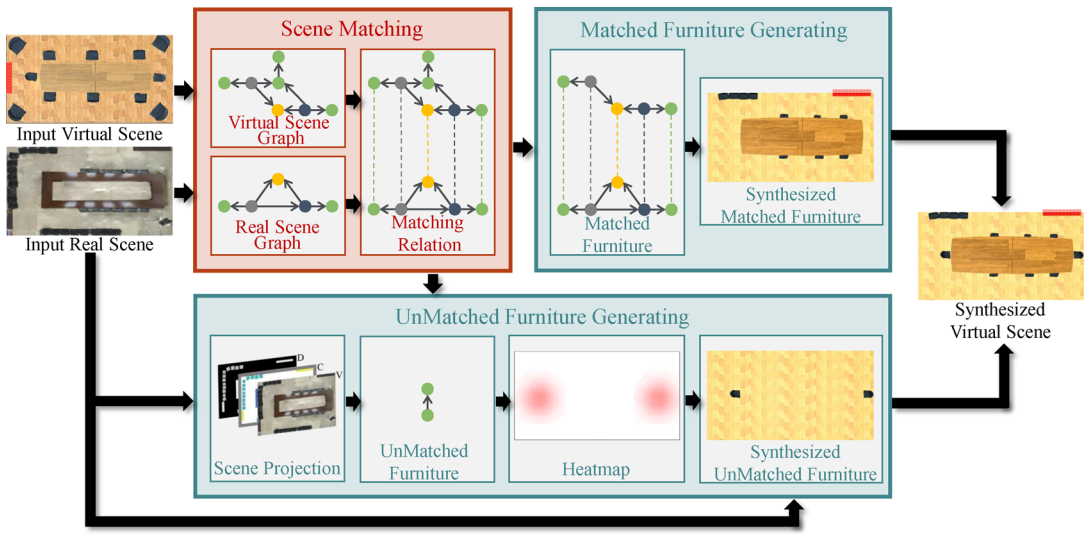


Fig. 2 The pipeline of the real-scene-constrained virtual scene synthesis method. Given the input virtual scene and the input real scene, we first represent them as scene graphs and use the scene matching model to predict the matching relation between the virtual furniture and the real furniture. Then, for the matched furniture, we use the matched furniture generating model to synthesize the new layout of the matched furniture. Next, for the unmatched furniture, we represent the input real scene as scene projection and use the unmatched furniture generating model to predict the heatmap of the unmatched furniture and synthesize the new layout of the unmatched furniture. Finally, the synthesized matched and unmatched furniture together form the synthesized virtual scene

Table 1 Furniture category definition and corresponding interaction type of different furniture

Category	Bed	Table	Cabinet	Chair	Bedside Table	Sofa
Inter-Type	H-Inter	H-Inter	V-Inter	H-Inter	H-Inter	H-Inter

The 'H' denotes 'horizontal', and the 'V' denotes 'vertical'

need to be considered are: (3) the unmatched virtual furniture F_{v}^{u} is resized and placed into $S_{\mathrm{v}}^{'}$ according to S_{r}; (4) the spatial and functional relationship relation of synthesized unmatched virtual furniture $\mathrm{SFR}(F_{\mathrm{v}}^{u'})$ is similar to that of its original virtual furniture $\mathrm{SFR}(F_{\mathrm{v}}^{u})$.

3.2 Scene representation

Before the scene matching and generating, we need to construct the scene representation. Similar to the [5], we use a combined representation of scene graph and scene projection to represent the real scene and the virtual scene, which can fully model the spatial and functional information of the scenes. For the scene graph representation, we use a directed graph to encode the scene, furniture, and room architecture, such as walls and windows, are encoded as nodes, and spatial and functional relationships are encoded as edges. For the scene projection representation, we encode the scene using a multi-channel top-down projection image. We use the scene graph in step 1 and use the scene projection in steps 2 and 3.

3.2.1 Scene graph

Given a scene with furniture and room architecture, a directed graph $\varsigma = (\mathbf{N^F}, \mathbf{N^A}, \mathbf{E})$ is constructed, where $\mathbf{N^F}$ represents the furniture node, $\mathbf{N^A}$ represents the room architecture node, and $\mathbf{E}$ represents the edge.

Furniture node Each furniture node $\mathbf{N^F}$ represents one furniture in the scene and is labeled with the furniture's category, size, orientation, and interaction type, such as touching horizontally and vertically (Table 1). In each scene, we choose a furniture node as the center node (e.g., bed in the bedroom scene, the biggest table in the living room scene), and use a Boolean value to mark the center node, 1 for the center node and 0 for others.

Room architecture node The room architecture nodes $\mathbf{N^A}$ are used to represent walls, windows, and doors in the scenes. Similar to the furniture node, these room architecture nodes are labeled with category, size, and orientation. The reason for constructing the room architecture node is the relation between the furniture, and the room architecture will influence the matching results.

Edge Edge represents the spatial and functional relationships between two furniture nodes, or a furniture node and a room architecture node. Different from previous work [5], which constructs spatial edges from functional edges separately, we assign three attributes to each edge: function, distance, and direction attributes.

The function attribute describes the semantic relationship, which has 6 options: (1) wall, (2) window, (3) door, (4) chain, (5) attach, and (6) center. Wall is selected on the edge from a wall to its nearby furniture (e.g., a sofa in front of a wall); window is selected on the edge from a window to its nearby furniture (e.g., a chair in front of a window); door is selected on the edge from a door to its nearby furniture (e.g., a cabinet by the door); chain is chosen on the edge from one furniture to another furniture nearby with the same category (e.g., a series of sofas); attach is chosen the edge from a bigger furniture (main-furniture in the attach-relationship) attached to a smaller furniture (sub-furniture in the attach-relationship), which includes the bed and the bedside table attached to it, the table and the chair attached to it, the table and the sofa attached to it; center is selected on the edges from the center furniture to a furniture. The distance attribute has 3 options: near, middle, and far. The direction attribute has 4 options: left, right, front, and back.

In this paper, the scene graph of the virtual and real scenes is generated manually. For each furniture, we represent it as a node. We represent the wall, the window, and the door as room architecture nodes. For each scene, it has four walls (top, right, down, and left), multiple windows, and multiple doors. We annotate the scene graph manually by marking the edges between furniture and furniture, and the edges between furniture and room architecture. For each edge, we choose its start node and then the end node, the function attribute is generated automatically based on the categories of the start node and the end node. The distance attribute of the edge is determined by the distance between the two furniture's oriented bounding boxes(OBB): if the distance is within 10% of the largest diagonal of two furniture, the attribute is set as near; else if the distance is within 30% of the largest diagonal of two furniture, the attribute is set as middle; else the attribute is set as far. The direction attribute is defined in the local coordinate frame of the edge start node.

3.2.2 Scene projection

Given a scene, a multi-channel top-down projection image $\rho = (\mathbf{V}, \mathbf{D}, \mathbf{C})$ is used to encode the scene, where $\mathbf{V}$ represents the view channels, $\mathbf{D}$ represents the depth channel, and $\mathbf{C}$ represents the category channel.

View The view $\mathbf{V}$ is an RGB image rendered orthographically from the top of the scene, which contains the visual appearance information of the indoor scene. R, G, and B components are stored in the first 3 channels.

Depth The depth buffer $\mathbf{D}$ corresponding to the view is stored in the fourth channel.

Category The category **C** is a multi-channel image corresponding to the view and depth, whose pixel value is encoded with the category of the projected furniture and room architecture.

In this paper, the scene projection of the virtual and real scenes is generated manually. For **V**, we first move the scene to the center of a $6m \times 6m$ region and then render the image orthographically from the top of the scene, with setting the resolution as 512×512. For **D**, we map the depth with $d = (4 - height)/4$, where $height$ is the height of the furniture.

3.3 Scene matching

Inspired by [22, 36], we formulate the scene matching problem as a graph matching problem. The difference between our scene graph matching and the previous work is our method uses GNNs to update the edge feature and propagate messages from edges and neighbor nodes in the embedding step since the edge has some attributes in our scene graph, while previous methods do not process the edge attributes.

Given the input virtual scene S_{v} and real scene S_{r}, we represent them as scene graph $\zeta_v = (\mathbf{N}_{\mathrm{v}}, \mathbf{E}_{\mathrm{v}}, \mathbf{G}_{\mathrm{v}}, \mathbf{H}_{\mathrm{v}})$ and $\zeta_{\mathrm{r}} = (\mathbf{N}_{\mathrm{r}}, \mathbf{E}_{\mathrm{r}}, \mathbf{G}_{\mathrm{r}}, \mathbf{H}_{\mathrm{r}})$. $\mathbf{N}_{\mathrm{v}}$ and $\mathbf{N}_{\mathrm{r}}$ denote the virtual and real furniture, and $\mathbf{E}_{\mathrm{v}}$ and $\mathbf{E}_{\mathrm{r}}$ denote the edges. The topology of the graphs is encoded by node-edge incidence matrices $\mathbf{G}_{\mathrm{v}}$, $\mathbf{G}_{\mathrm{r}}$, $\mathbf{H}_{\mathrm{v}}$, and $\mathbf{H}_{\mathrm{r}}$. If the cth edge begins at the ith virtual furniture and ends at the jth virtual furniture, $g_{\mathrm{vic}} = h_{\mathrm{vjc}} = 1$. If the cth edge begins at the i^{th} real furniture and ends at the jth real furniture, $g_{\mathrm{ric}} = h_{\mathrm{rjc}} = 1$. Given the embedded feature of virtual and real furniture $\mathbf{Ef}_v$ and $\mathbf{Ef}_r$, we compute the affinity matrix $\mathbf{A}^{m \times n}$ whose elements encode the node-to-node affinity between $\mathbf{N}_{\mathrm{r}}$ and $\mathbf{N}_{\mathrm{v}}$. The target of the scene matching is to predict the matching matrix **M** so that the sum of the node compatibility is maximized, **M** denotes the one-to-one correspondence between $\mathbf{N}_{\mathrm{v}}$ and $\mathbf{N}_{\mathrm{r}}$:

$$\begin{cases} \max\limits_{\mathbf{M}} (tr(\mathbf{M}^T \mathbf{A})) \\ \mathbf{M} \in \{0, 1\}^{m \times n}, \mathbf{M1} \leq \mathbf{1}, \mathbf{M}^T \mathbf{1} \leq \mathbf{1} \end{cases} \tag{1}$$

where m and n are the number of $\mathbf{N}_{\mathrm{v}}$ and $\mathbf{N}_{\mathrm{r}}$. If **M** is a one-to-one mapping, then this optimization can be cast as a linear matching problem, which can be efficiently optimized by the Hungarian algorithm. As shown in Fig. 3, we use the deep graph neural network to predict the feature of nodes. Since the Hungarian algorithm is not differentiable, we use the Sinkhorn operator to approximate the Hungarian algorithm during training phase.

Algorithm 1 Scene Matching Algorithm

Input: the real scene graph ζ_r, the node feature $\mathbf{f}_r^N$, the edge feature $\mathbf{f}_r^E$, and the virtual scene graph ζ_v, the node feature $\mathbf{f}_v^N$, the edge feature $\mathbf{f}_v^E$,.
Output: optimal matching matrix **M**
1: $\mathbf{Ef}_r$=GNNs(ζ_r, $\mathbf{f}_r^N$, $\mathbf{f}_r^E$)
2: $\mathbf{Ef}_v$=GNNs(ζ_v, $\mathbf{f}_v^N$, $\mathbf{f}_v^E$)
3: $\mathbf{A} = \mathbf{Affinity}(\mathbf{Ef}_r, \mathbf{Ef}_v)$
4: **if** Training:
5: $\quad \mathbf{M} = Sinkhorn(\mathbf{A})$
6: **else** :
7: $\quad \mathbf{M} = Hungarian(\mathbf{A})$

Feature definition The node feature $\mathbf{f}^N$ is defined as a $a \times 1$ vector, which concatenates: category (one-hot encoding, $c \times 1$ for c categories), size(2×1), orientation(2×1) and interaction type(one-hot encoding, 2×1). The edge feature $\mathbf{f}^E$ is defined as a $b \times 1$ vector, which concatenates function attribute (one-hot encoding, 6×1), distance (one-hot encoding, 3×1) and direction (one-hot encoding, 4×1).

Feature embedding We use an embedding model to propagate and aggregate the message encoded in edges and room architecture nodes into the node feature (Algorithm 1 line 1–2). The embedding model takes $\mathbf{f}^N$ and $\mathbf{f}^E$ as input and outputs the embedded furniture node feature **Ef**. The embedding model consists of two parts: each part is a GNN module with L layers. For each layer, the GNN module has the same architecture. For $l \in [1, L]$ layer, the feature is propagated, aggregated, and updated as follows:

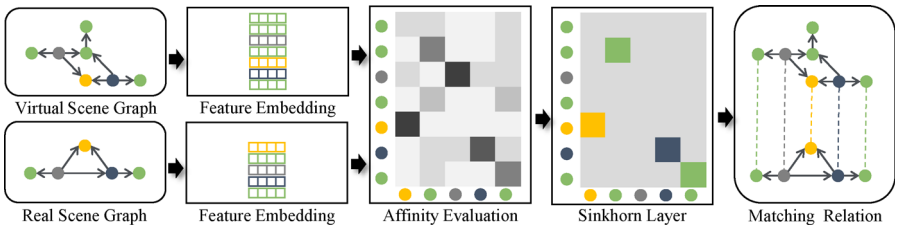


Fig. 3 Given the virtual scene graph and the real scene graph, we first use the embedding model(GNNs, Eq. 2) to embed the node feature. Then, we use the affinity model(fully connected layers, Eq. 3) the evaluate the affinity of the nodes between the virtual and real scene graphs. Finally, we use the Sinkhorn layer(Eq. 4) to predict the matching relation between the nodes in the virtual scene graph and the nodes in the real scene graph

$$\begin{aligned}
\text{UpdateEdge}: \mathbf{f}_l^{E_{yx}} &= \mathbf{MLP_E}(\mathbf{f}_{l-1}^{N_x}, \mathbf{f}_{l-1}^{N_y}, \mathbf{f}_{l-1}^{E_{yx}}) \\
\text{Message}: \mathbf{m}_l^{yx} &= \mathbf{MLP_M}(\mathbf{f}_{l-1}^{N_x}, \mathbf{f}_{l-1}^{N_y}, \mathbf{f}_l^{E_{yx}}) \\
\text{AggregateMessage}: \mathbf{m}_l^{x} &= \sum_{y \in \text{Neighbour}(x)} \mathbf{m}_l^{yx} \\
\text{UpdateNode}: \mathbf{f}_l^{N_x} &= \mathbf{MLP_U}(\mathbf{m}_l^{x})
\end{aligned} \tag{2}$$

For node $\mathbf{N}_x$, which could be any furniture node or room architecture node, it has neighbor nodes $\mathbf{N}_y$ ($y \in \text{Neighbour}(x)$), linked with edges $\mathbf{E}_{xy}$. We first use an mlp $\mathbf{MLP_E}$ to update the edge feature $\mathbf{f}^{E_{yx}}$ (line 1 in Eq. 2). Then, we calculate the message $\mathbf{m}^{yx}$ propagated from each neighbor node with an MLP $\mathbf{MLP_M}$ (line 2). Next, we aggregate all the message $\mathbf{m}^x$ with a summation function (line 3). Finally, we update the node feature $\mathbf{f}^{N_x}$ with an mlp $\mathbf{MLP}_U$ (line 4). For $l = 0, \mathbf{f}_0^E = \mathbf{f}^E, \mathbf{f}_0^N = \mathbf{f}^N$. For $l = L$, the embedded furniture node feature $\mathbf{Ef} = \mathbf{f}_L^N$.

Affinity evaluation With embedded node features $\mathbf{Ef}_{\mathrm{r}}$ and $\mathbf{Ef}_{\mathrm{v}}$, we apply an affinity model to calculate the affinity matrix $\mathbf{A}_{m\times n}$ (Algorithm 1 line 3).

$$\mathbf{A}_{i,j} = \mathbf{Affinity}(\mathbf{Ef}_{ri}, \mathbf{Ef}_{vj}) \tag{3}$$

where $\mathbf{Ef}_{\mathrm{r}i}$ is the embedded node feature for node $\mathbf{N}_{r\,i}^F$, $i \in [0, m-1]$, $\mathbf{Ef}_{\mathrm{v}\,j}$ is the embedded node feature for node $\mathbf{N}_{v\,j}^F$, $j \in [0, n-1]$. The **Affinity** is the affinity model, and we adopt fully connected layers as the affinity model.

Sinkhorn layer Since the Hungarian algorithm is not differentiable, the Sinkhorn operator can be seen as an approximation to the Hungarian algorithm, and it is fully differentiable due to that only matrix multiplication and division are taken. Thus, we use a Sinkhorn layer [29] to solve Eq. 1 during training (Algorithm 1 line 4–5).

$$\begin{aligned}
\mathbf{M}_0 &= \exp(\mathbf{A}/\tau) \\
\mathbf{M}_k &= Nor_c(Nor_l(\mathbf{M}_{k-1})) \\
\mathbf{M}^* &\approx \lim_{k\to\infty}(\mathbf{M}_k)
\end{aligned} \tag{4}$$

where τ is a predefined small positive real number, Nor_c and Nor_l are the row-wise and column-wise normalization for the $k-th$ iteration. It can be proved that the best matching relationship $\mathbf{M}^*$is obtained when $\mathbf{M}^* \approx \lim_{k\to\infty}(\mathbf{M}_k)$ [29]. The Sinkhorn operator can't be realized since the limitation operation to k, an alternative way is to replace $\lim_{k\to\infty}$ with a big positive integer K so that $\mathbf{M}_K$ can converge to $\mathbf{M}^*$.

Loss Given the ground truth matching relationship $\mathbf{M}$ and the predicted matching relationship $\mathbf{M}^*$, we compute the cross-entropy loss between them.

$$\text{Loss} = -\sum_{i,j} \mathbf{M}_{i,j} \log \mathbf{M}_{i,j}^* + (1 - \mathbf{M}_{i,j}) \log(1 - \log \mathbf{M}_{i,j}^*) \tag{5}$$

3.4 Matched furniture layout generating

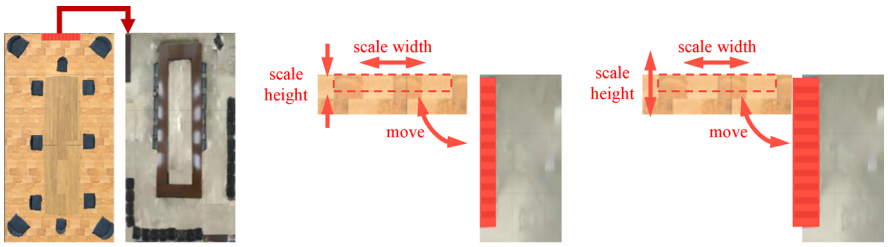
Given the matching matrix $\mathbf{M}$, $F_{v/r}$ can be divided into matched virtual/real furniture $F_{v/r}^m$ and unmatched virtual/real furniture $F_{v/r}^u$. In this section, we synthesize the layout of the matched virtual furniture F_v^m constrained by F_{r}^m, i.e., predicting suitable layouts of F_{v}^m to match the corresponding real furniture F_{r}^m. One of the simplest ways is to set the new layout of F_{v}^m to be the same as F_{r}^m, while it may be unsuitable in some cases. As shown in Fig. 4, we try to synthesize the layout of the virtual red cabinet according to the real brown cabinet. Due to the red and brown cabinets having different aspect ratios, scaling the virtual cabinet directly may reduce the user's perception and immersion of the original virtual scene. We propose a layout parameters optimization algorithm (Algorithm 2) to synthesize the new layout of virtual furniture which not only matches the size of the real furniture but also ensures the user's passive haptics. The layout parameters optimization algorithm only considers the layout and category of the furniture and does not take the lighting conditions and the deterioration of the furniture into account.

Algorithm 2 Layout Parameters Optimization Algorithm

Input: F_v^m's size $\mathbf{Size}_v^m$, the correspondent matched real furniture's size $\mathbf{Size}_r^m$ and center $\mathbf{Ct}_r^m$, the center of the real scene $\mathbf{CTR}_r$
Output: $F_v^{m'}$'s center $\mathbf{Ct}_v^{m'}$ and size $\mathbf{Size}_v^{m'}$
1: $\lambda, \omega, \eta = ParamPredict(\mathbf{Size}_v^m, \mathbf{Size}_r^m, \mathbf{Ct}_r^m, \mathbf{CTR}_r)$
2: $\mathbf{Cor}_r^m = FindCorner(\lambda, \mathbf{Ct}_r^m, \mathbf{Size}_r^m)$
3: $\mathbf{Cor}_v^m = FindCorner(\lambda, \mathbf{Ct}_r^m, \mathbf{Size}_v^m)$
4: $\Delta = \mathbf{Cor}_v^m - \mathbf{Cor}_r^m$
5: $\mathbf{Ct}_v^{m'} = \mathbf{Ct}_r^m + \Delta$
6: $\mathbf{R} = \mathbf{Size}_r^m ./ \mathbf{Size}_v^m$
7: **if** ω:
8: **if** η:
9: $\mathbf{R} = [\max(\mathbf{R}), \max(\mathbf{R})]$
10: **else**:
11: $\mathbf{R} = [\min(\mathbf{R}), \min(\mathbf{R})]$
12: $\mathbf{Size}_v^{m'} = \mathbf{Size}_v^m \cdot \mathbf{R}$
13: $\mathbf{Ct}_v^{m'} = \mathbf{Ct}_v^{m'} + \mathbf{Size}_v^{m'} \cdot (\mathbf{R} - \mathbf{1})/2$

The layout parameters optimization algorithm takes the size $\mathbf{Size}_{\mathrm{v}}^m$ of F_{v}^m, the size $\mathbf{Size}_{\mathrm{r}}^m$ and the center $\mathbf{Ct}_r^m$ of F_r^m, the center $\mathrm{CTR}_{\mathrm{r}}$ of the real scene as inputs, and outputs the center $\mathrm{Ct}_{\mathrm{v}}^{m'}$ and size $\mathbf{Size}_{\mathrm{v}}^{m'}$ of the synthesized virtual furniture $F_{\mathrm{v}}^{m'}$.

We first use the ParamPredict algorithm to predict 3 parameters λ, ω and η to control the layout optimization process (line 1). λ is a 4×1 vector that determines which corner of F_{r}^m and $F_{\mathrm{v}}^{m'}$ are aligned. ω is a Boolean value that determines whether $F_{\mathrm{v}}^{m'}$ is scaled with keeping the original aspect ratio. η is a Boolean value that determines whether $F_{\mathrm{v}}^{m'}$ is allowed to be out of the space occupied by F_{r}^m. Then, we obtain the coordinates of the aligned corners of F_{r}^m and



(a) Matching Relation (b) Scale Directly (c) Layout Parameters Optimization Algorithm

Fig. 4 Based on the matching relation, the red cabinet is matched to the brown cabinet in the real scene (**a**). A straightforward method is to scale the red cabinet directly and move the scaled red cabinet to the location of the brown cabinet (**b**). However, this will change the aspect ratio of the red cabinet and reduce the user's perception and immersion of the virtual scene. Our method allows the virtual furniture to not match the size of the real furniture while ensuring the user's passive haptics. For the red cabinet, we first scale up it while maintaining the aspect ratio, and then move the red cabinet to the location of the brown cabinet by aligning the red cabinet and brown cabinet on their human-facing sides (**c**)

$F_{\mathrm{v}}^{m'}$ (lines 2–3) and translate $F_{\mathrm{v}}^{m'}$ with Δ (lines 4–5). Next, we calculate the original scaling ratio $\mathbf{R}$ (2×1) of $F_{\mathrm{v}}^{m'}$ and F_{r}^{m}(line 6). If $F_{\mathrm{v}}^{m'}$ has to be scaled with keeping $\mathbf{R}$ (line 7) and is allowed to be out of the space occupied by its correspondent matched real furniture (line 8), $\mathbf{R}$ is set as the maximum ratio in 2 directions (line 9). If $F_{\mathrm{v}}^{m'}$ has to be scaled with keeping the aspect ratio and is not allowed to be out of the space occupied by its correspondent matched real furniture (line 10), $\mathbf{R}$ is set as the minimum ratio in 2 directions (line 11). If $F_{\mathrm{v}}^{m'}$ doesn't need to be scaled with keeping the aspect ratio, we keep $\mathbf{R}$ as the original. Finally, we scale $F_{\mathrm{v}}^{m'}$ according to the aligned corner(line 12) and update the center position of $F_{\mathrm{v}}^{m'}$(line 13). As in Algorithm 2, we describe the furniture by its size and center, and we do not consider light variance.

ParamPredict Parameters λ, ω and η control the adjustment process, where ω is related to the category of the furniture and λ and η are not.

For λ, we link the real furniture center $\mathbf{Ct}_{\mathrm{r}}^{m}$ and the real scene center $\mathbf{CTR}_{\mathrm{r}}$), and the vertex closest to this line of the 4 vertices of the rectangle $\mathbf{v}_i$ is defined as the alignment corner.

$$\begin{cases} dis_i = \min(\left|\mathbf{Ct}_{\mathrm{r}}^{m}\mathbf{v}_i\right|, \left|\mathbf{CTR}_r\mathbf{v}_i\right|, \left|\mathbf{Ct}_{\mathrm{r}}^{m}\mathbf{v}_i - \frac{\mathbf{Ct}_{\mathrm{r}}^{m}\mathbf{v}_i \cdot \mathbf{Ct}_{\mathrm{r}}^{m}\mathbf{CTR}_r}{\mathbf{Ct}_{\mathrm{r}}^{m}\mathbf{CTR}_r \cdot \mathbf{Ct}_{\mathrm{r}}^{m}\mathbf{CTR}_r} \cdot \mathbf{Ct}_{\mathrm{r}}^{m}\mathbf{CTR}_r\right|) \\ \lambda = \arg\min(\mathrm{dis}_1, \mathrm{dis}_2, \mathrm{dis}_3, \mathrm{dis}_4) \end{cases} \tag{6}$$

For ω, we set threshold values T for different categories of furniture ($T_{\mathrm{bed}} = 0.1$, $T_{\mathrm{table}} = 0.1$, $T_{\mathrm{carbinet}} = 0.2$, $T_{\mathrm{chair}} = 0.1$, $T_{\mathrm{bedside\ table}} = 0.075$, $T_{\mathrm{sofa}} = 0.1$). The threshold T is determined by competent human perception. For different furniture categories, people have different levels of tolerance for imbalances in their aspect ratios. We test different categories of furniture and determine suitable T. If the difference between the aspect ratio of two matched furniture is smaller than the threshold value, ω is set as 1, else as 0.

$$\omega = 1 \ \text{ if } \left|\frac{\mathbf{Size}_{\mathrm{v}}^{m} - \mathbf{Size}_{\mathrm{r}}^{m}}{\mathbf{Size}_{\mathrm{r}}^{m}}\right| < T \ \text{ else } 0 \tag{7}$$

For η, if the furniture is located in the corner of the wall (if this furniture is near two adjacent walls according to the edges), η is set as 1, else as 0.

3.5 Unmatched furniture layout generating

In this section, we synthesize the layout of the unmatched virtual furniture F_{v}^{u}. Inspired by [5], we treat the unmatched furniture layout generating problem as a prediction problem, i.e., predicting the location probability distribution of the F_{v}^{u} in $S_{\mathrm{v}}^{'}$ (Fig. 5). The location probability distribution $P(\mathbf{Ct}|F_{\mathrm{v}}^{u}, \mathbf{S}_{\mathrm{v}}^{'})$ is conditional probability distribution, with

$$\sum_{\mathbf{Ct} \in \mathrm{AllPosition}} P(\mathbf{Ct}|F_{\mathrm{v}}^{u}, S_{\mathrm{v}}^{'}) = 1 \tag{8}$$

The main difference between our method and the previous work [5] is that: our unmatched furniture layout generating model is a uniform model for all furniture categories and takes the category and size of F_{v}^{u} as inputs. While the previous work uses different models for different categories of furniture. This is because we arrange the F_{v}^{u} into $S_{\mathrm{v}}^{'}$ with keeping its original size to maximize the perception and immersion, while previous work has no constraints on the size of the furniture.

Input The input is the projection of the real scene which encodes the spatial and functional information.

Output The output is a location probability distribution map, i.e., a heatmap, which is a one-channel image with the same size as the input projection image, and each pixel of the heatmap represents the probability that the furniture exists at this pixel, and the sum of the probability is 1.

Hourglass neural network We use the hourglass network [49] to extract the spatial and functional information and use multiple prediction heads implied with CNNs to predict the heatmap for F_{v}^{u}. The hourglass network first uses residual and max pooling layers to process features down to a very low resolution. At each max pooling step, the resolution of the feature maps decreases by two. After reaching the lowest resolution, the hourglass network begins to upsample and combines the features across multiple scales. At each upsampling layer, the resolution of feature maps increases by two. After reaching the output resolution of the network, the prediction head is applied to produce the heatmap.

FiLM Furthermore, the hourglass takes additional inputs, i.e., the category and original size of the unmatched virtual furniture. The additional inputs are injected into the network via featurewise linear modulation (FiLM), a process that applies a learned scale and shifts to the output of every convolutional layer.

Loss Given the ground truth heatmap $\mathrm{HM}_{\mathrm{GT}}$ and the predicted heatmap HM_{PD}, the loss is given by computing the MSE loss of two images:

$$\mathrm{Loss} = \sum_{p \in \mathrm{HM}} \|\mathrm{HM}_{\mathrm{GT}}(p) - \mathrm{HM}_{\mathrm{PD}}(p)\|^2 \tag{9}$$

Location, orientation, and size Given the predicted heatmap, we consider the pixel with the maximum probability as the precise center location of $F_{\mathrm{v}}^{u'}$. For a properly synthesized scene, the furniture should be set with its front direction pointing to the center of the scene. With this principle, we determine the orientation of F_{v}^{u}. For the size, we try to keep its original size. In the case where there is not enough space for the synthesized furniture, we appropriately shrink the synthesized furniture to arrange them in the empty space.

4 Dataset construction

4.1 Indoor Scene Layout Synthesis Dataset

In this paper, we introduce a new dataset, especially for the indoor scene layout synthesis problem. This dataset is constructed based on the 3D-FRONT dataset [50] which is a large-scale dataset of synthetic indoor scenes. To build the Indoor Scene Layout Synthesis Dataset, we carefully select indoor scenes from the 3D-FRONT dataset of different scene types, different design styles, and different decorative styles. The Indoor Scene Layout Synthesis Dataset contains 100 living room scenes, 100 bedroom scenes, 100 kitchen scenes, and 100 office scenes, and $A_{100}^{2} = 9900$ scene layout synthesis results between living room scenes, $A_{100}^{2} = 9900$ scene layout synthesis results between bedroom scenes, $A_{100}^{2} = 9900$ scene layout synthesis results between kitchen scenes, and $A_{100}^{2} = 9900$ scene layout synthesis results between office scenes. For each scene, we remove uncommon furniture. For the bedroom, we only keep the beds, the table, the cabinet, the chair, and the bedside table in them. For the living room, we only keep the table, the cabinet, the chair, and the sofa in them. For the kitchen, we only keep the table, the cabinet and the chair in them. For the office, we only keep the table, the cabinet, the chair, and the sofa in them. These scenes are represented as a flat list of furniture(with category, geometry, location, size, and orientation), and they contain no spatial or functional information between furniture.

4.2 Scene matching annotating

We use a two-step scene matching method to annotate the training data. First, a brute force score-based algorithm is used to initialize the matching matrix $\mathbf{M}$, and then $\mathbf{M}$ is refined manually.

In the brute force score-based algorithm, given two scene graphs ς_1 with m furniture $\mathbf{N}_{1\,i}^{F}$ with $i \in [0, m-1]$ and ς_2 with n furniture $\mathbf{N}_{2\,j}^{F}$ with $j \in [0, n-1]$, we compute the matching score for each node pair $\mathrm{NodePair}_t = [\mathbf{N}_{1\,i}^{F}, \mathbf{N}_{2\,j}^{F}]$ with Eq. 10.

$$\begin{aligned}\mathrm{Score} = {} & \mathrm{NodeScore} + \mathrm{WallScore} + \mathrm{WindowScore} \\ & + \mathrm{DoorScore} + \mathrm{AttachScore} + \mathrm{ChainScore} + \mathrm{CenterScore}\end{aligned} \tag{10}$$

where NodeScore represents the furniture affinity between the two nodes in the pair, which is initialized with 0. If two nodes have the same category, a predefined category score $Score_c$ is added into NodeScore; if two nodes have a similar size $\pm 50\%$, a predefined category size $\mathrm{Score_s}$ is added into NodeScore if two nodes have the same interaction type, a predefined interaction score $\mathrm{Score_i}$ is added into NodeScore. In our implementation, Score_c, Score_s, and Score_i are set as 10,5,3 because of the importance of the different node attributes. WallScore represents the affinity of the furniture-wall relation of two nodes, which is also initialized with 0. If the furniture-wall relations of two nodes are the same, i.e., both nodes have edges whose functional attribute is Wall and distance attribute is near, $WallScore$ is set to a predefined wall score. doorScore and windowScore are set as the same as WallScore. attachScore represents the affinity of the attach-relation of two nodes. If the attach-relations of two nodes are the same, attachScore is set to a predefined attach score. chainScore and centerScore are set as the same as attachScore. In our implementation, the predefined wall, door, window, attach, chain, and center scores are set as 2, 2, 2, 5, 5, 1. We enumerate all possible matching matrices $\mathbf{M}$, sum the matching scores of the node pairs with 1 in $\mathbf{M}$, get

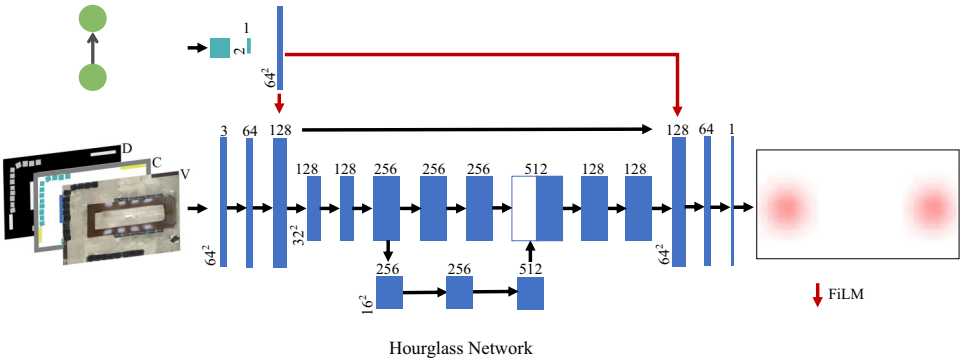


Fig. 5 Given the scene projection of the input real scene and the unmatched virtual furniture, the hourglass network is implied to encode the spatial and functional information of the scene, and the semantic information of the unmatched virtual furniture is injected into the hourglass network via FiLM

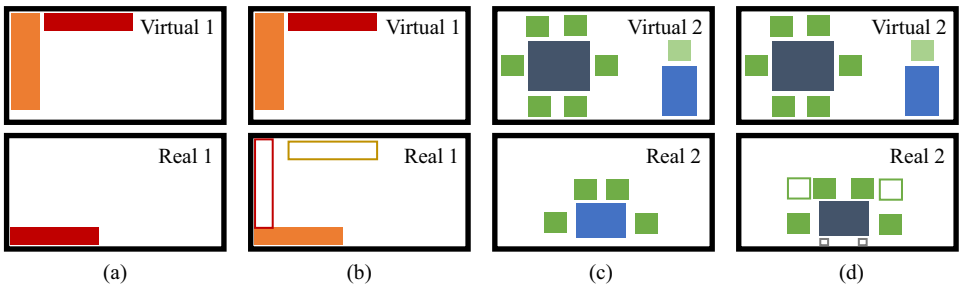


Fig. 6 Examples of manually adjusting matching and generating results during dataset annotation

the score of M, and select the M corresponding to the highest score as $\mathbf{M}^*$.

Second, we adjust $\mathbf{M}^*$ manually. Here are 2 examples to demonstrate the manual adjusting. As shown in Fig. 6, in virtual scene 1, there are two cabinets, while there is only 1 cabinet in the real scene. Based on the brute force score-based algorithm, the red virtual cabinet will match the real cabinet because these two cabinets have the same size and are both close to a wall (a). While we try to match the virtual furniture located in the corner of the walls with the real furniture which is also located in the corner of the walls during manual adjusting. Thus, we manually adjusted the matching relationship and match the orange virtual cabinet with the real cabinet (b). In virtual scene 2, there are two tables surrounded by different numbers of chairs, while there is only 1 table in the real scene. Based on the brute force score-based algorithm, the light blue virtual table will match the real table because these two tables have the same size and are both surrounded by chairs (c), while we try to match virtual furniture with real furniture that has a similar number of sub-furniture during manual adjusting. Thus, we manually adjusted the matching relationship and match the deep blue virtual table with the real table (d). The lighting conditions have no effect on the matching results.

4.3 Unmatched furniture layout generating annotating

To annotate the location probability distribution map for each unmatched furniture of the training data, we use a two-step unmatched generating method. First, we use a grid-based algorithm to arrange unmatched furniture F_1^{u} in scene one S_1 to scene two S_2. The inputs are scene graph ς_1 of S_1, node $\mathbf{N}_1^F$ which denotes F_1^{u}, and scene graph ς_2 of S_2. The output is the best center position $\mathbf{Ct}^*$ of F_1^{u}. We construct a 64×64 uniform grid on a $6\,m \times 6\,m$ floor and place S_2 at the center of the floor. For each cell center of the grid Pos, if Pos is inside S_2 and empty, we move F_1^{u} to Pos and determine the size and orientation. We update ς_2 by adding F_1^{u} as node $\mathbf{N}^{F^{\mathrm{u}}}{}_2$ into ς_2, and adding edges to link $\mathbf{N}^{F^{\mathrm{u}}}{}_2$ with all other nodes. Then, we calculate node affinity Score between $\mathbf{N}_1^{F^{\mathrm{u}}}$ and $\mathbf{N}^{F^{\mathrm{u}}}{}_2$ as in Eq. 10. At last, we obtain $\mathbf{Ct}^*$ of F_1^{u} with the largest score.

Second, we adjust $\mathbf{Ct}^*$manually. Here are 2 examples to demonstrate the manual adjusting. As shown in Fig. 6, in virtual scene 1, the unmatched red virtual cabinet will be generated at the location of the yellow rectangle in the real scene according to the grid-based algorithm. However, we try to maintain the relative spatial relationship between the matched and unmatched furniture during manual adjusting. Thus, we manually adjusted the generating result and place the red virtual cabinet at the location of the red rectangle in the real scene (b). In virtual scene 2, there are two more virtual chairs surrounding the deep blue virtual table than there are real chairs surrounding the real table. Thus, these 2 chairs needed to be generated. Based on the grid-based algorithm, these 2 chairs will be generated at the location of the gray rectangles in the real scene. However, for the sub-furniture surrounding a main-furniture, we try to generate them symmetrically while maintaining their size and the spatial relationship between them. Thus, we manually

adjusted the generating result and place these 2 chairs at the location of the green rectangles in the real scene (d).

After we get CT^*, the ground truth map of the location probability distribution is given by two-dimensional Gaussian distribution:

$$\mathrm{HT}_{64\times64}(x,y) = \frac{1}{2\pi\sigma_1\sigma_2}\exp\left(-\frac{1}{2}\left[\left(\frac{x-\mathbf{Ct}^*[0]}{\sigma_1}\right)^2 + \left(\frac{y-\mathbf{Ct}^*[1]}{\sigma_2}\right)^2\right]\right) \tag{11}$$

where σ_1 and σ_2 are set as 2 in our implementation. The lighting conditions have no effect on the generating results.

5 Experiment

After constructing the dataset, we train our models on this dataset and compare the quality of our method with the heuristic rules-based method [8] on this dataset.

5.1 Training

5.1.1 Deep scene matching network

The GNN architecture is implemented with PYG, the affinity layer and Sinkhorn layer is implemented with Pytorch. We train the deep scene matching network with cross-entropy loss using SGD for optimization with the learning rate initially set as 0.001 and divided by 10 after 1000 iterations. The batch contains 16 samples per iteration which are sampled randomly. The hyperparameters L, τ and ε are set to 10, 1.0 and $1e-4$. Since bedrooms, living rooms, kitchens, and offices have different spatial structures and different types of furniture, four separate models are trained for each scene category.

5.1.2 Deep scene generating network

The whole network is implemented with Pytorch. We train the Deep Scene Generating Network with MSE loss using SGD for optimization with the learning rate initially set as 0.001 and divided by 10 after 10000 iterations. The batch contains 8 samples per iteration which are sampled randomly. Similarly, four separate models are trained for each scene category.

5.2 Metric

We quantity the quality of our scene layout synthesis method with 2 metrics: matching accuracy for virtual and real scene graphs, and location accuracy for unmatched furniture.

Matching accuracy (MA) For each furniture F_v in the virtual scene, we get its correspondent matching vector M_i^{pred} from the predicted matching matrix M^{pred} and M_i^{gt} from ground truth matching matrix M^{gt}, where i is the index of this furniture. We take a scalar product on these two vectors and obtain the matching value ma_i with Eq. 12.

$$ma_i = M_i^{\mathrm{pred}\,T} \cdot M_i^{\mathrm{gt}} \tag{12}$$

Then we compute the matching accuracy with Eq. 13 according to the number m of F_v.

$$\mathrm{MA} = \sum_i ma_i/m \tag{13}$$

Location accuracy (LA) For each unmatched virtual furniture $F_{\mathrm{v}\,i}^u$, we obtain the predicted location $\mathrm{Loc}_i^{\mathrm{pred}}$ and the ground truth $\mathrm{Loc}_i^{\mathrm{gt}}$, and then calculate location accuracy with Eq. 14 by computing the inverse of the average Euclidean distance of all pairs.

$$\mathrm{LA} = q/\sum_i\left(\left\|\mathrm{Loc_i}^{\mathrm{pred}} - \mathrm{Loc_i}^{\mathrm{gt}}\right\|_2\right) \tag{14}$$

In order to evaluate our method more finely, we not only compute the above metrics for all furniture in the whole scene $\mathrm{MA}_{\mathrm{total}}$, and $\mathrm{LA}_{\mathrm{total}}$, but also compute the above metrics separately for each category of furniture, such as $\mathrm{MA}_{\mathrm{table}}$, and $\mathrm{LA}_{\mathrm{table}}$.

5.3 Results

We conduct experiments on the Indoor Scene Layout Synthesis Dataset to evaluate the quality of our method. The quantitative results are shown in Table 2 and 3.

5.3.1 Matching accuracy

For the bedroom, the deep scene matching network can predict the matching relationship precisely and achieve 92% accuracy in total. For different kinds of furniture in the bedroom, the deep scene matching network predicts the matching relationship with different accuracy. For beds, there are usually only one or two beds in the scene, thus it is easy to predict the matching relationship. For bedside tables, due to their strong correlation with beds, it is also easy to process. For tables and cabinets, their sizes change a lot, thus the accuracy is relatively low. For chairs, there are too many kinds of chairs, and they are randomly arranged in the bedroom. For the living room, the deep scene matching network can predict the matching relationship precisely and achieve 96% accuracy in total. For sofas and tables, both have strong semantic character and the sizes of them in the living room

Table 2 Comparison of MA values with bedroom, living room, kitchen, and office scene by using our method and heuristic rules-based method

Scene	Method	MA_{bed}	MA_{table}	$MA_{cabinet}$	MA_{chair}	$MA_{bedside\ table}$	MA_{sofa}	MA_{total}
Bedroom	Ours	1	0.93	0.79	0.82	0.91	/	0.92
	Heu	0.91	0.82	0.71	0.68	0.88	/	0.83
Living room	Our	/	0.97	0.89	0.92	/	0.98	0.96
	Heu	/	0.86	0.72	0.65	/	0.89	0.75
Kitchen	Our	/	0.96	0.81	0.89	/	/	0.88
	Heu	/	0.89	0.69	0.72	/	/	0.71
Office	Our	/	0.93	0.81	0.79	/	0.96	0.92
	Heu	/	0.84	0.77	0.69	/	0.88	0.79

Table 3 Comparison of LA values with bedroom, living room, kitchen, and office scene by using our method and heuristic rules-based method

Scene	Method	LA_{bed}	LA_{table}	$LA_{cabinet}$	LA_{chair}	$LA_{bedside\ table}$	LA_{sofa}	LA_{total}
Bedroom	Ours	1.89	9.10	8.33	10.50	13.12	/	8.09
	Heu	0.81	1.10	1.18	1.92	4.01	/	1.08
Living room	Our	/	7.68	7.14	13.50	/	7.69	5.26
	Heu	/	1.12	1.19	1.81	/	1.07	1.26
Kitchen	Our	/	6.08	6.91	10.21	/	/	7.36
	Heu	/	1.23	2.33	3.25	/	/	2.47
Office	Our	/	7.69	6.77	9.98	/	6.91	7.41
	Heu	/	1.48	2.78	2.41	/	1.71	2.08

change relatively little, thus it is easy to predict. Different from the bedroom, chairs in the living room usually have a strong correlation with the table and are arranged in a row, which makes it easier to process. For the kitchen, the deep scene matching network can predict the matching relationship precisely and achieve 88% accuracy in total. For tables, there are only one or two tables in the kitchen, thus it is easy to match. For cabinets and chairs, the number of cabinets and chairs varies widely from one kitchen scene to another, making it difficult to predict the correct matching relationship. For the office, the deep scene matching network can predict the matching relationship precisely and achieve 92% accuracy in total. Similar to the kitchen and living room, the table and sofa are relatively easy to predict the matching relationship, while the cabinet and chair are harder to predict the matching relationship.

5.3.2 Location accuracy

For the bedroom, the deep scene generating network can predict the location of the unmatched furniture precisely and the location accuracy is 8.09 in total. For different kinds of furniture in the bedroom, the deep scene generating network predicts the location with different accuracy. For beds, they have the lowest location accuracy due to it is hard to arrange a new bed in a bedroom that already has one. However, bedside tables can be easily arranged in the source scene by putting them at the side of beds. Tables and cabinets are usually placed against a wall, which greatly reduces the difficulty of generation. As for chairs, their location can be easily predicted by putting them near the tables. For the living room, the deep scene generating network can predict the location of the unmatched furniture precisely and the location accuracy is 5.26 in total. For different kinds of furniture in the living room, the deep scene generating network predicts the location with different accuracy. For sofas, tables, and cabinets, they have a strong spatial relationship with scene architecture, which greatly constrains the scope of generation. For the kitchen, the deep scene generating network can predict the location of the unmatched furniture precisely and the location accuracy is 7.36 in total. For chairs, they have a strong spatial relationship with tables, making it easy to predict their location. For cabinets, although they are generally placed against a wall, there is no certainty as to which wall they will be placed against, so it is not easy to accurately predict their location. For tables, it is much harder to arrange a new table in a kitchen that already has one. For the office, the deep scene generating network can predict the location of the unmatched furniture precisely and the location accuracy is 7.41 in total. Similar to the kitchen and living room, the table and sofa are relatively easy to predict the location, while the cabinet and chair are harder to predict the location.

5.4 Comparison

As shown in Table 2 and 3, for scene matching, our method outperforms the heuristic rules-based method by 9% in bedroom, by 21% in living room, by 17% in kitchen, and by 13% in office. The reason is that the deep scene matching network is flexible to deal with the structural information, while the brute force score-based algorithm follows a fixed paradigm for processing the structural information. For unmatched furniture layout generating, our method has higher location accuracy than the heuristic rules-based method in bedroom, living room, kitchen, and office. The grid-based algorithm determines the location of the unmatched furniture by calculating similarity, it only focuses on keeping the spatial and functional relationship in the target scene and ignores the fidelity of the source scene.

6 User study

We design a user study to evaluate the interactivity and fidelity of our method.

6.1 User study design

6.1.1 Participants

We have recruited 24 participants, 20 males, and 4 females, between 20 and 30 years old. All of them have had MR experience before. Participants have normal and corrected vision, and none report vision or balance disorders.

6.1.2 Hardware and software setup

We used one set of Hololens HMD to construct our user study, allowing participants to interact with the synthesized scene. The HMD was connected to one workstation with a 3.6GHz Intel(R) Core(TM) i7-9900KF CPU, 16GB of RAM, and an NVIDIA GeForce GTX 3080 graphics card. The tracked physical space hosting the VR applications is 12.6 m × 8.1 m. We used Unity 2021.1 to implement our tasks. The virtual environments were rendered at 60fps for each eye.

6.1.3 EC and CC

There are 2 control conditions (CC1 and CC2) and an experimental condition(EC) in each scene. For EC, CC1, and CC2, our method, manual method, and the heuristic rules-based method are used to synthesize the virtual scenes, respectively. In the manual furniture placement method, the person places the furniture according to the input virtual scene and the real scene, and the placement time of each furniture does not exceed 30 s.

6.1.4 Task

To fully assess the interactivity and realism of the synthesized scenes, participants were asked to touch three synthetic scenes (conference room, sightseeing room, concert hall), 20 times each, and to touch as much of the virtual furniture as possible. Conference room is synthesized with inputs of real and virtual conference rooms (Fig. 7a, b). For real scenes, we first use the Kinect to scan the entire scene and get the point clouds of the scene, then we use point clouds to reconstruct the scene, and manually annotate the furniture and its layout based on the reconstructed scene. The main difference between the input real and virtual conferences room is the number of chairs around the council board. Besides, there are two opposite chairs near the council board in the virtual scene, whereas there are none in the real scene. What's more, there are two sofas, one coffee table, and one armchair in the virtual scene, which do not exist in the real scene. Sightseeing room is synthesized with inputs of the real conference room and virtual sightseeing room (Fig. 7a, c). In the virtual scene, the scene has floor-to-ceiling windows on three sides for sightseeing. Sofas and cabinets are placed against another wall. A dining table with 8 chairs is arranged near the main floor-to-ceiling window. One of the chairs is bigger than the others. Besides, there are two cabinets while no sofa in the real scene, thus it is challenging to arrange the bookshelf and sofa to keep their relationship. Concert hall is synthesized with inputs of the real conference room and virtual concert hall (Fig. 7a, d). The major structure of the virtual scene is simple, it consists of one platform and 16 chairs, and the chairs are neatly placed in 3 rows. Though the virtual scene is very simple, this task is quite challenging, not only because there is no platform in the real scene, but also because there is no space to arrange 3 rows of chairs. What's more, the chairs in the virtual scene keep a distance from the platform, which imposes great restrictions on the synthesis problem. Living room is synthesized with inputs of the real conference room and virtual living room (Fig. 7a, e). Different from other virtual scenes, the scene structure of the virtual living room is quite different from that of the real conference room. In the real scene, there is only one large table surrounded by multiple chairs, whereas in the virtual scene, there are two tables surrounded by chairs and sofas. In addition, in the real scene, there are multiple chairs placed in multiple rows along the wall, whereas in the virtual scene, there is a separate chair placed in the corner.

6.1.5 Procedure

All participants are required to perform the three scenes under all conditions. For each scene, the participants are first required to observe the input real and virtual scenes for one minute. Then the participants wear the Hololens and touch

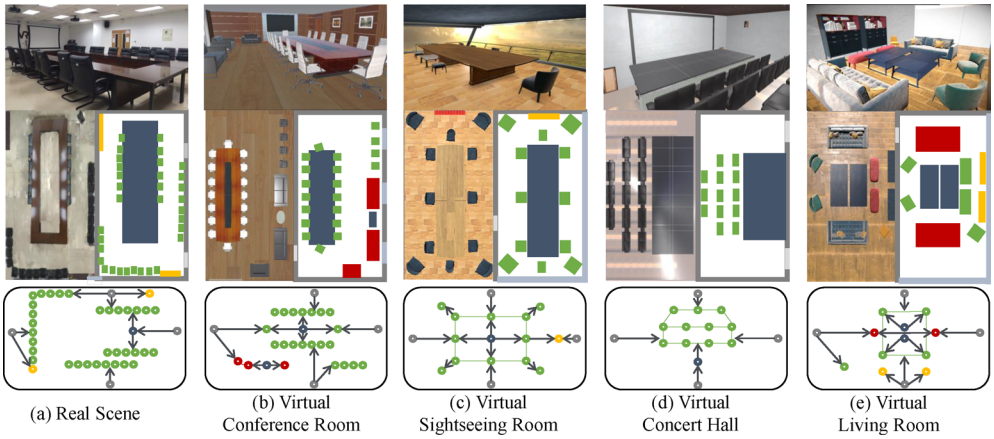


Fig. 7 The real scene and virtual scenes used in the user study. The first one is a real conference room (**a**), the second one is a virtual conference room (**b**), the third one is a virtual sightseeing room (**c**), the fourth one is a virtual concert hall (**d**), and the fifth one is a virtual living room (**e**). Each scene is also described as a sketch denoting different kinds of furniture as rectangles of different colors. The walls are represented as dark grey rectangles, the doors are represented as light grey rectangles, the windows are represented as light blue rectangles, the tables are represented as deep blue rectangles, the chairs are represented as green rectangles, the cabinets are represented as yellow rectangles, and the sofas are represented as red rectangles. Each scene is also described as a scene graph, where nodes of different colors indicate furniture and edges indicate relationships between furniture

the synthesized scenes 20 times and evaluate the interactivity and fidelity of the scene. The order of the synthesized scenes is random. The minimum interval between the tasks is one day, and the maximum interval is three days.

In each touch, participants first select and walk toward the virtual furniture they are interested in, and then touch the surface of the virtual furniture with their hands. When the distance between the participant's hand and the surface of the virtual furniture is less than 3 cm, the HMD sends an audible alert signal to remind the user that the hand is already near the surface of the virtual furniture. Subsequently, the user's hand moves near the surface of the virtual furniture, and if the virtual furniture has corresponding real furniture, the user can experience a sense of touch, and if there is no corresponding real furniture, the user has no touch experience. We do not ask the user to record a touch experience for each touch, because such constant asking would interrupt the user's immersion experience. When we detect that the distance between the participant's hand and the surface of the virtual furniture is less than 3 cm, from the surface of that virtual furniture, a ray of 5 cm is emitted along the normal direction, and if the ray intersects with the surface of the real furniture, then we record a touch experience, otherwise no touch experience. We encouraged participants to touch different parts of the same virtual furniture, and touches with different parts of the same furniture were recorded independently. After the task, participants evaluate interactivity and fidelity through subjective perception scores with a questionnaire.

6.1.6 Metric

We evaluate the interactivity of our methods with touchable time and touch score and evaluate the fidelity with class correctness and experience score. (1) Touchable time: this metric calculates the time of successful touch in 20 touches. (2) Class correctness: this metric calculates the number of times the class of the virtual furniture matches the class of the corresponding real furniture in 20 touches. (3) Touch score: this metric calculates the sum of the user's scores for the first two questions in the questionnaire. The first question is 'How well does the virtual furniture match the real furniture overall'. The second question is 'How well does the interaction surface between the real furniture and the virtual furniture fit'. The participants then give a score for each question, ranging from 0 to 2, 0 for bad, and 2 for good. If $\mathbf{F}_{tgt}{}^{T}$ was unmatched, the score is 0. (4) Experience Score: this metric calculates the sum of the user's scores for the third, fourth, and fifth questions in the questionnaire. The third and fourth questions are 'How well does the synthesized furniture fit with the real/virtual scene'. The fifth question is 'How well does the size of the synthesized furniture match the original size'. The scoring mechanism is the same as the one in touch score.

6.1.7 Statistical analysis

For each metric, the values of EC are compared to CC1 and CC1, respectively. First, the normality of the data was

assessed using the Shapiro–Wilk test. Then, the comparison was performed with a repeated-measures ANOVA if the values showed a normal distribution. When values did not follow a normal distribution, the comparison was made using a Wilcoxon signed-rank test. In addition to the p-value of the statistical test, we also estimate the size of the effect using Cohen's d. The d values are translated to qualitative effect size estimates of Huge ($d > 2.0$), Very Large ($2.0 > d > 1.2$), Large ($1.2 > d > 0.8$), Medium ($0.8 > d > 0.5$), Small ($0.5 > d > 0.2$), and Very Small ($0.2 > d > 0.01$).

6.2 Results and discussion

6.2.1 Touchable time

Table 4 gives the touchable time. The third column gives the average and standard deviation, and the fourth column gives the relative reduction from EC to CC. The fifth to seventh columns provide statistical information about the difference between the EC and CC. Statistical significance is indicated by an asterisk.

Compared with all control conditions of all four scenes, our method significantly improves the touchable time, and the size ranges from 'Large' to 'Huge'. The reason is that the deep scene matching network not only takes fidelity into account but also tries to maximize interactivity. Take conference room for example, the main difference between EC and CC1 is the layout of the synthesized sofas. The sofas only exist in the target scene, a straightforward way to synthesize them is to generate them, which do not violate the rules of semantic consistency. A smart way is to use the chairs to represent them. Despite the lock sacrifice in fidelity, interactivity is greatly improved. In sightseeing room and concert hall, the EC and CC2 have the same matching relationship, however, there are still differences between the interactivity and fidelity. In CC2, participants only have 30 s to synthesize each virtual furniture, it is challenging to fit the real furniture and the virtual furniture well. It may affect the success of the interaction due to that we use the distance between the interactive points to evaluate the interactivity. While the layout parameters optimization algorithm suitably fits the furniture. In living room, we treat the tables in the real scene as two separate tables of the same size, which makes it easy to deal with virtual tables. The main difference between EC and CC2 in living room is the layout of the virtual sofas. In CC2, the virtual sofas are generated, while they are matched to the real chairs in EC. The touchable time significantly improved when the users interact with the sofas.

6.2.2 Class correctness

Table 5 gives the class correctness. Compared with all control conditions of all four scenes, our method significantly improves the class correctness, and the size ranges from 'large' to 'huge'. The reason is that our method is flexible to deal with the spatial relation of furniture. Take conference room for example, the main difference between EC and CC1 is the layout of the synthesized chairs. In the virtual scene, the 12 chairs are neatly placed in 3 rows, while in the real scene, it only has 8 chairs placed in 2 rows. To keep the spatial consistency rules, 4 virtual chairs are generated in the empty space in the real scene. However, in EC, our method

Table 4 Comparison of touchable time values with different scenes and conditions

Scene	Condition	Avg ± std. dev.	(CC-EC) /CC	p	Cohen's d	Effect size
Conference room	EC	6.51 ± 1.06				
	CC1	4.72 ± 1.24	0.38	< 0.001	1.56	Very large
	CC2	5.50 ± 0.47	0.18	0.004	1.24	Very large
	EC1	5.19 ± 0.52	0.25	< 0.001	1.56	Very large
	EC2	5.49 ± 0.21	0.18	0.004	1.32	Very large
Sightseeing room	EC	7.38± 1.09				
	CC1	5.36 ± 1.56	0.37	< 0.001	1.5	Very large
	CC2	6.70 ± 0.09	0.10	0.004	0.84	Large
Concert hall	EC	10.15± 0.21				
	CC1	5.67 ± 0.95	0.79	< 0.001	6.46	Huge
	CC2	9.27 ± 0.60	0.091	< 0.001	1.94	Very large
	EC1	6.43 ± 0.96	0.57	< 0.001	5.30	Huge
	EC2	6.45 ± 0.84	0.14	0.002	0.95	Large
Living room	EC	8.82± 0.98				
	CC1	8.29 ± 0.71	0.40	0.004	1.13	Large
	CC2	5.77 ± 1.12	0.53	< 0.001	1.99	Very large

Table 5 Comparison of class correctness values with different scenes and conditions

Scene	Condition	Avg ± std. dev.	(CC-EC) /CC	p	Cohen's d	Effect size
Conference room	EC	5.52± 0.88				
	CC1	4.11 ± 1.05	0.34	< 0.001	1.69	Very large
	CC2	4.74 ± 0.90	0.16	0.004	0.87	Large
	EC1	4.79 ± 0.66	0.15	0.002	0.92	Large
	EC2	4.34 ± 0.41	0.27	0.005	1.69	Very large
Sightseeing room	EC	6.51 ± 0.60				
	CC1	4.88 ± 1.88	0.33	< 0.001	1.17	Large
	CC2	6.04 ± 0.45	0.08	0.004	0.88	Large
Concert hall	EC	10.03 ± 0.18				
	CC1	6.26 ± 0.49	0.60	< 0.001	10.17	Huge
	CC2	9.39 ± 0.44	0.07	< 0.001	1.91	Very large
	EC1	6.60 ± 0.73	0.52	< 0.001	6.42	Huge
	EC2	5.68 ±0.51	0.76	< 0.001	1.48	Very large
Living room	EC	7.65 ± 0.88				
	CC1	6.92 ± 0.60	0.11	0.005	1.01	Large
	CC2	7.17 ± 0.99	0.07	< 0.001	1.71	Very large

breaks these 12 chairs into 3 groups and synthesized them into correspondent real chairs. Although the spatial relation is partly broken, the arrangement in EC is more reasonable.

6.2.3 Touch score

Table 6 gives the touch score. Compared with all control conditions of all four scenes, our method significantly improves the touch score, and the size ranges from 'large' to 'huge'. The reason is that the deep scene matching network reasonably balances interactivity and fidelity. Take sightseeing room for example, the main difference between our method and CC1 is the layout of the sofa. Similar to conference room, our method uses the chairs that existed in the real scene to represent the sofa even if there is enough empty space to generate it in the real scene. Although the fidelity will be affected, the interactivity has been greatly improved. In CC2, the human adopted the same strategy, which shows that the deep scene matching network makes a good choice between interactivity and fidelity.

6.2.4 Experience score

Table 7 gives the experience score. Compared with all control methods of all four scenes, our method significantly improves the experience score, and the size ranges from 'very large' to 'huge'. The reason is that the deep scene generating network not only considers the spatial relationship between furniture in the virtual scene but also the relationship in the real scene. Take conference room for example, our method generates the virtual coffee table in front of a real chair. While in CC1, the coffee table is synthesized in front of the cabinet. Although the relationship between the coffee table and the sofas is kept, the relationship between the synthesized coffee table and the already existing cabinet is strange and unacceptable.

6.3 Ablation study

We design an ablation study to evaluate the performance of the deep scene matching network and the deep scene generating network.

6.3.1 Ablation study design

The *Participants*, *Hardware and software setup*, *Task*, *Metric*, *Procedure* and *Statistical analysis* are the same as Sect. 5.2.

EC. In EC1 and EC2, we replace the deep scene matching network and the deep scene generating network with the corresponding part of the heuristic rules-based algorithm, respectively, and the other parts of EC1 and EC2 are the same as EC.

6.3.2 Results and discussion

EC1 Our method significantly improves the touchable time, class correctness, touch score, and experience score than EC1, and the size ranges from 'large' to 'huge'. For scene matching, the core is how to maximize interactivity while ensuring fidelity. For EC1, although it uses different kinds of features to encode the spatial and functional information, it may ignore the structural information.

Table 6 Comparison of touch score values with different scenes and conditions

Scene	Condition	Avg ± std. dev.	(CC-EC) /CC	*p*	Cohen's d	Effect size
Conference room	EC	12.42 ± 1.07				
	CC1	9.13 ± 3.18	0.36	< 0.001	1.39	Very large
	CC2	10.68 ± 2.47	0.16	0.003	0.91	Large
	EC1	10.94 ± 1.55	0.13	< 0.001	1.11	Large
	EC2	10.78 ± 1.05	0.15	0.005	1.54	Very large
Sightseeing room	EC	15.16 ± 1.79				
	CC1	9.15 ± 3.28	0.66	< 0.001	2.28	Huge
	CC2	13.61 ± 1.60	0.11	< 0.001	0.92	Large
Concert hall	EC	19.16 ± 0.34				
	CC1	14.42 ± 1.35	0.33	< 0.001	4.83	Huge
	CC2	16.70 ± 2.57	0.15	< 0.001	1.34	Very large
	EC1	15.68 ±3.88	0.22	< 0.001	1.26	Very large
	EC2	12.73 ± 1.04	0.19	< 0.001	1.66	Very large
Living room	EC	17.17 ± 2.10				
	CC1	15.29 ± 1.17	0.12	0.002	1.00	Large
	CC2	9.21 ± 2.11	0.86	0.005	1.45	Very large

Table 7 Comparison of experience score values with different scenes and conditions

Scene	Condition	Avg ± std. dev.	(CC-EC) /CC	p	Cohen's d	Effect size
Conference room	EC	23.93 ± 0.82				
	CC1	16.70 ± 2.08	0.43	< 0.001	4.58	Huge
	CC2	20.42 ± 1.14	0.17	< 0.001	3.54	Huge
	EC1	22.18 ± 0.64	0.08	< 0.001	2.39	Huge
	EC2	17.25 ± 1.44	0.39	< 0.001	5.69	Huge
Sightseeing room	EC	26.23 ± 1.66				
	CC1	20.72 ± 3.70	0.27	< 0.001	1.92	Very large
	CC2	23.33 ± 0.97	0.12	< 0.001	2.14	Huge
Concert hall	EC	30.08 ± 0.20				
	CC1	12.94 ±9.00	1.32	< 0.001	2.69	Huge
	CC2	25.70 ± 2.57	0.17	< 0.001	2.40	Huge
	EC1	15.64 ± 2.53	0.92	< 0.001	8.05	Huge
	EC2	12.14 ± 2.67	1.48	< 0.001	1.84	Very large
Living room	EC	27.16 ± 3.15				
	CC1	21.54 ± 2.76	0.26	0.004	0.91	Large
	CC2	19.91 ± 3.96	0.36	0.003	0.94	Large

EC2 Our method significantly improves the touchable time, class correctness, touch score, and experience score than EC2, and the size ranges from 'large' to 'huge'. For the unmatched furniture layout generating problem, some 'creativity' is also needed to arrange virtual furniture in the real scene. As shown in Fig. 8, due to that the sofas are matched to the chairs against the wall, and there is a table in front of them, thus it is unreasonable to arrange the table chair between the two sofas. For EC2, it forces the coffee table synthesized into this terrible layout to satisfy the original spatial relationship with the sofas. However, our deep scene generating network learns some different paradigms from the dataset and staggers the coffee table and the table, which greatly improves the fidelity.

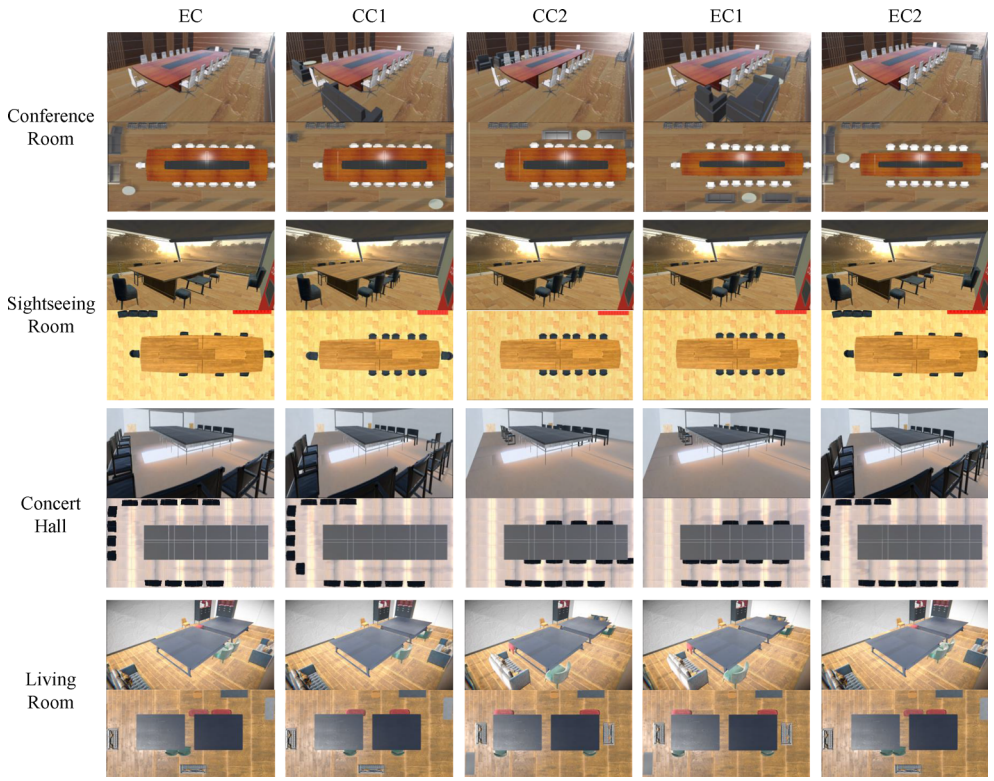


Fig. 8 Qualitative results of the user study. The synthesized conference room, sightseeing room, concert hall, and living room are demonstrated in the first, second, third, and forth rows. We show the first and top views of each synthesized scene under different conditions. As for synthesized conference rooms, the main difference is the layout of the synthesized table and sofas. As for synthesized sightseeing rooms, the main difference is the layout of the synthesized chairs. As for synthesized concert halls, the main difference is the layout of the synthesized chairs. As for synthesized living rooms, the main difference is the layout of the synthesized cabinets, sofas, and chairs. Since both the deep scene matching network and the brute force score-based algorithm match all virtual furniture in the sightseeing room, the synthesized sightseeing rooms are the same for EC and EC2, and the synthesized sightseeing rooms are the same for CC2 and EC1. The synthesized living rooms are the same for CC2 and EC1 because the deep scene generating network and the grid-based algorithm generate the same layout for virtual sofas. The synthesized living rooms are the same for EC and EC2 because the deep scene generating network and the grid-based algorithm generate the same layout for the virtual cabinet

7 Conclusion, limitations, and future work

We introduce the real-scene-constrained virtual scene layout synthesis problem and propose a scene layout synthesis method to solve this problem, which contains 3 steps: scene matching, matched furniture layout generating, and unmatched furniture layout generating. For scene matching, we propose a deep scene matching network to predict the matching relationship between real furniture and virtual furniture. For matched furniture layout generating, we propose a layout parameters optimization algorithm to arrange the matched virtual furniture into the new virtual scene. For unmatched furniture layout generating, we propose a deep scene generating network to arrange the unmatched virtual furniture into the real scene. We evaluate the quality of our method to synthesize scenes of different kinds and sizes, and the results show that, compared with the heuristic rules-based method, our method increases the matching accuracy by 15% and improves the location accuracy by a factor of 5.71, in total. We also design a user study to evaluate the interactivity and fidelity of our method. Compared to the manual method and the heuristic rules-based method, our method has a significant improvement in interactivity and fidelity.

Limitation and future work One limitation of our method is taking little consideration of the vertical direction of the furniture. In future work, we will take the vertical information into account. Another limitation is that the main structure of the real scene and the virtual scene are similar. Our approach may fail if the structure is too different, such as synthesizing a bedroom constrained by a kitchen. In the future, we will build a cross-scene-category dataset and try to solve the cross-category scene layout synthesis problem. Another limitation is that our method only pays attention to the main furniture in the scene, the tiny furniture and decoration styles are ignored. In the future, we will apply some new scene representation methods to model the information of

the tiny furniture and decoration styles and synthesize them. Another limitation is that our method annotates the scene graphs manually which is time-consuming. Although there are a number of rule-based and data-driven methods to generate the scene graphs automatically, manual calibration is still required to get good results. In the future, we will design a neural network-based method to generate the scene graph automatically.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00371-023-03167-4.

Funding This work is supported by the National Natural Science Foundation of China through Project 61932003, 62372026, by Beijing Science and Technology Plan Project Z221100007722004, and by National Key R&D plan 2019YFC1521102.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Huang, C.-K., Chen, Y.-L., Shen, I.-C., Chen, B.-Y.: Retargeting 3d objects and scenes with a general framework. Comput. Graph. Forum **35**(7), 33–42 (2016)
2. Lin, J., Cohen-Or, D., Zhang, H., Liang, C., Sharf, A., Deussen, O., Chen, B.: Structure-preserving retargeting of irregular 3d architecture. ACM Trans. Graph. **30**(6), 1–10 (2011)
3. Dong, Z.-C., Wu, W., Xu, Z., Sun, Q., Yuan, G., Liu, L., Fu, X.-M.: Tailored reality: perception-aware scene restructuring for adaptive vr navigation. ACM Trans. Graph. **40**(5), 1–15 (2021)
4. Dong, Z.-C., Fu, X.-M., Zhang, C., Wu, K., Liu, L.: Smooth assembled mappings for large-scale real walking. ACM Trans. Graph. **36**(6), 1–13 (2017)
5. Wang, K., Lin, Y.-A., Weissmann, B., Savva, M., Chang, A.X., Ritchie, D.: Planit: planning and instantiating indoor scenes with relation graph and spatial prior networks. ACM Trans. Graph. **38**(4), 1–15 (2019)
6. Fu, Q., Chen, X., Wang, X., Wen, S., Zhou, B., Fu, H.: Adaptive synthesis of indoor scenes via activity-associated object relation graphs. ACM Trans. Graph. **36**(6), 1–13 (2017)
7. Fisher, M., Savva, M., Li, Y., Hanrahan, P., Nießner, M.: Activity-centric scene synthesis for functional 3d scene modeling. ACM Trans. Graph. **34**(6), 1–13 (2015)
8. Cant, R.J., Langensiepen, C.S.: Methods for automated object placement in virtual scenes. In: 2009 11th International Conference on Computer Modelling and Simulation, pp. 431–436 (2009)
9. Shamir, A., Sorkine, O.: Visual media retargeting. In: ACM SIGGRAPH ASIA 2009 Courses. SIGGRAPH ASIA '09. Association for Computing Machinery, New York, NY, USA (2009). doi:10.1145/1665817.1665828
10. Ma, L., Lin, W., Deng, C., Ngan, K.N.: Image retargeting quality assessment: a study of subjective scores and objective metrics. IEEE J. Sel. Topics Signal Process. **6**(6), 626–639 (2012)
11. Dong, Z.-C., Fu, X.-M., Yang, Z., Liu, L.: Redirected smooth mappings for multiuser real walking in virtual reality. ACM Trans. Graph. **38**(5), 1–17 (2019)
12. Merrell, P., Schkufza, E., Li, Z., Agrawala, M., Koltun, V.: Interactive furniture layout using interior design guidelines. ACM Trans. Graph. **30**(4), 1–10 (2011)
13. Peng, C.-H., Yang, Y.-L., Wonka, P.: Computing layouts with deformable templates. ACM Trans. Graph. **33**(4), 1–11 (2014)
14. Merrell, P., Schkufza, E., Koltun, V.: Computer-generated residential building layouts. ACM Trans. Graph. **29**(6) (2010)
15. Fu, Q., Fu, H., Deng, Z., Li, X.: Indoor layout programming via virtual navigation detectors. Sci. China Inf. Sci. **65**(8), 1–2 (2022)
16. Zhang, S., Han, Z., Lai, Y., Zwicker, M., Zhang, H.: Stylistic scene enhancement GAN: mixed stylistic enhancement generation for 3d indoor scenes. Vis. Comput. **35**(6–8), 1157–1169 (2019)
17. Vasylevska, K., Kaufmann, H.: Towards efficient spatial compression in self-overlapping virtual environments. In: 2017 IEEE Symposium on 3D User Interfaces (3DUI), pp. 12–21 (2017)
18. Zhao, X., Su, Z., Komura, T., Yang, X.: Building hierarchical structures for 3d scenes with repeated elements. Vis. Comput. **36**(2), 361–374 (2020)
19. Wang, K., Savva, M., Chang, A.X., Ritchie, D.: Deep convolutional priors for indoor scene synthesis. ACM Trans. Graph. **37**(4), 1–14 (2018)
20. Li, M., Patil, A.G., Xu, K., Chaudhuri, S., Khan, O., Shamir, A., Tu, C., Chen, B., Cohen-Or, D., Zhang, H.: Grains: generative recursive autoencoders for indoor scenes. ACM Trans. Graph. **38**(2), 1–16 (2019)
21. Feng, S., Mostafa, H., Nassar, M., Majumdar, S., Tripathi, S.: Exploiting long-term dependencies for generating dynamic scene graphs. In: 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 5119–5128 (2023)
22. Zanfir, A., Sminchisescu, C.: Deep learning of graph matching. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2684–2693 (2018)
23. Yan, J., Yin, X.-C., Lin, W., Deng, C., Zha, H., Yang, X.: A short survey of recent advances in graph matching. In: Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval. ICMR '16, pp. 167–174. Association for Computing Machinery, New York, NY, USA (2016)
24. Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. Eur. J. Oper. Res. **176**(2), 657–690 (2007)
25. Cho, M., Lee, J., Lee, K.M.: Reweighted random walks for graph matching. In: Proceedings of the 11th European Conference on Computer Vision: Part V. ECCV'10, pp. 492–505. Springer, Berlin, Heidelberg (2010)
26. Leordeanu, M., Hebert, M.: A spectral technique for correspondence problems using pairwise constraints. In: Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, vol. 2, pp. 1482–14892 (2005)
27. Hahn, P., Grant, T., Hall, N.: A branch-and-bound algorithm for the quadratic assignment problem based on the Hungarian method. Eur. J. Oper. Res. **108**(3), 629–640 (1998)
28. Wang, T., Ling, H., Lang, C., Feng, S.: Graph matching with adaptive and branching path following. IEEE Trans. Pattern Anal. Mach. Intell. **40**(12), 2853–2867 (2018)
29. Kushinsky, Y., Maron, H., Dym, N., Lipman, Y.: Sinkhorn algorithm for lifted assignment problems. SIAM J. Imag. Sci. **12**(2), 716–735 (2019)
30. Liu, C., Niu, D., Yang, X., Zhao, X.: Graph matching based on feature and spatial location information. Vis. Comput. **39**(2), 711–722 (2023)

31. Li, C., Tang, Y., Zou, X., Zhang, P., Lin, J., Lian, G., Pan, Y.: A novel agricultural machinery intelligent design system based on integrating image processing and knowledge reasoning. Appl. Sci. **12**(15), 7900 (2022)
32. Ji, Z., Chen, K., He, Y., Pang, Y., Li, X.: Heterogeneous memory enhanced graph reasoning network for cross-modal retrieval. Sci. China Inf. Sci. **65**(7), 1–13 (2022)
33. Wu, T., Duan, F., Chang, L., Lu, K.: Human-object interaction detection via interactive visual-semantic graph learning. Sci. China Inf. Sci. **65**(6), 1–2 (2022)
34. Zhou, D., Liu, Y., Li, X., Zhang, C.: Single-image super-resolution based on local biquadratic spline with edge constraints and adaptive optimization in transform domain. Vis. Comput. **38**(1), 119–134 (2022)
35. Chen, Y., Zhang, Q., Guan, Z., Zhao, Y., Chen, W.: Gemvis: a visual analysis method for the comparison and refinement of graph embedding models. Vis. Comput. **38**(9), 3449–3462 (2022)
36. Wang, R., Yan, J., Yang, X.: Learning combinatorial embedding networks for deep graph matching. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 3056–3065 (2019)
37. Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C.: Graph wavenet for deep spatial-temporal graph modeling. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. IJCAI'19, pp. 1907–1913. AAAI Press, Palo Alto, CA (2019)
38. Cao, S., Lu, W., Xu, Q.: Deep neural networks for learning graph representations. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. AAAI'16, pp. 1145–1152. AAAI Press, Palo Alto, CA (2016)
39. Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., Hsieh, C.-J.: Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Amp; Data Mining. KDD '19, pp. 257–266. Association for Computing Machinery, New York, NY, USA (2019)
40. Li, Q., Han, Z., Wu, X.-M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI'18/IAAI'18/EAAI'18. AAAI Press, Palo Alto, CA (2018)
41. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI'18/IAAI'18/EAAI'18. AAAI Press, Palo Alto, CA (2018)
42. Huang, W., Zhang, T., Rong, Y., Huang, J.: Adaptive sampling towards fast graph representation learning. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. NIPS'18, pp. 4563–4572. Curran Associates Inc., Red Hook, NY, USA (2018)
43. Chen, J., Ma, T., Xiao, C.: FastGCN: Fast learning with graph convolutional networks via importance sampling. In: International Conference on Learning Representations (2018)
44. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. ICML'17, pp. 1263–1272. JMLR.org, New York, NY (2017)
45. Spinelli, I., Scardapane, S., Uncini, A.: Adaptive propagation graph convolutional network. IEEE Trans. Neural Netw. Learn. Syst. **32**(10), 4755–4760 (2021)
46. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2009)
47. Wei, H., Meng, L.: An accurate stereo matching method based on color segments and edges. Pattern Recognit. **133**, 108996 (2023)
48. Ryan Prescott Adams, R.S.Z.: Ranking via sinkhorn propagation (2011)
49. Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) Computer Vision-ECCV 2016, pp. 483–499. Springer, Cham (2016)
50. Fu, H., Jia, R., Gao, L., Gong, M., Zhao, B., Maybank, S., Tao, D.: 3d-future: 3d furniture shape with texture. Int. J. Comput. Vis. **129**, 3313–3337 (2021)

Runze Fan is a Ph.D student at the School of Computer Science and Engineering, Beihang University, China. His current research focuses on virtual reality and augmented reality.

Lili Wang is a professor at the School of Computer Science and Engineering, Beihang University, and a researcher of the State Key Laboratory of Virtual Reality Technology and Systems. Her research interests include virtual reality, augmented reality, and rendering.

Xinda Liu is currently working toward the Ph.D. degree at the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing, China. His research interests include machine learning and image processing.

Sio Kei Im is a professor at the Faculty of Applied Sciences, Macau Polytechnic University, and a rese archer at the Engineering Research Center of Applied Technology on Machine Translation and Artificial Intelligence, Ministry of Education. His research interests include video coding, image processing, machine learning for NLP and multimedia.

Chan Tong Lam is currently an Associate Professor with the Faculty of Applied Sciences, Macao Polytechnic University, Macao SAR, China. His research interests include mobile wireless communications, digital signal processing, machine learning in communications, and computer vision in smart cities.