Graphical Models 85 (2016) 46–55

Contents lists available at ScienceDirect

Graphical Models

journal homepage: www.elsevier.com/locate/gmod

Structure guided interior scene synthesis via graph matching

Shi-Sheng Huang[a,*], Hongbo Fu[b], Shi-Min Hu[a]

[a] Tsinghua University, Beijing, China
[b] City University of Hong Kong, Hong Kong, China

ARTICLE INFO

Article history:
Received 14 December 2015
Revised 4 March 2016
Accepted 24 March 2016
Available online 6 May 2016

Keywords:
Scene synthesis
Graph matching
Augmented graph

ABSTRACT

We present a method for reshuffle-based 3D interior scene synthesis guided by scene structures. Given several 3D scenes, we form each 3D scene as a structure graph associated with a relationship set. Considering both the object similarity and relation similarity, we then establish a furniture-object-based matching between scene pairs via graph matching. Such a matching allows us to merge the structure graphs into a unified structure, i.e., *Augmented Graph* (*AG*). Guided by the *AG*, we perform scene synthesis by reshuffling objects through three simple operations, i.e., *replacing*, *growing* and *transfer*. A synthesis compatibility measure considering the environment of the furniture objects is also introduced to filter out poor-quality results. We show that our method is able to generate high-quality scene variations and outperforms the state of the art.

1. Introduction

Recently, 3D interior scenes have received more and more attention due to the huge demand in the industries such as computer games and virtual reality. However, designing and creating 3D digital scenes are still time-consuming even for artists. Fisher et al. [1] provided an efficient solution for 3D scene synthesis from examples based on moderate-to-large scene datasets. However, such a learning based algorithm is still complicated due to the complexity for data collection. Besides, the learned probability model might not always achieve user-desired constraints, such as rigid grid layouts or exact alignment relationships. Such issues might be solved by utilizing the original examples rather than learning from the example dataset.

It is still a desirable way to synthesize scenes directly from a small set of 3D scene examples without learning algorithms. Starting from such a point, Xie et al. [2] introduced a non-learning-based scene synthesis method by grouping the furniture objects into different types of units and reshuffling the interchangeable objects from the same units. Although their method could generate some kind of diverse new scenes, it is still rather limited due to the limited grouping types. In addition, their local analysis ignored the scene's layout structure information, which is, however, a very important guidance cue for scene generation. We observed that there is a latent rule in the layout distribution of the scene furniture objects *locally* and *globally*. Locally, furniture objects often 'contact' with each other, following a certain kind of relation. For example, a *chair* often closely faces a *table*, and a *bedside cabinet* is always at one side of a *bed* with one side aligned. Globally, these furniture objects with the local relationships form a layout structure. Based on the above observations, we carefully analyze the layout structures of the exemplar scenes and synthesize new scenes utilizing the relations between the layout structures, which have not been explored by Xie et al. [2].

Given several 3D interior scenes as examples, our goal is to synthesize new scenes with variations using a geometric approach rather than a learning-based strategy. Although furniture objects vary a lot in geometry, they latently relate with each other according to the relations among objects. In this paper, we first define five kinds of relations between furniture objects (Fig. 4(a-e)), i.e., *support relation*, *vertical contact relation*, *facing relation*, *aligned relation* and *close relation*, which widely exist in the 3D interior scenes. Then we represent each 3D scene as a structure graph. Our structure graph is different from the previous ones [2], since we associate a relationship set rather than a single relationship with each edge in the structure graph. We establish a matching between the layout subgraphs (Fig. 4(f)) via graph matching, which provides a cue to relate two structure graphs. Based on the matching, we *merge* the scene structures into an *Augmented Graph* (*AG*), which encodes all the layout structure information among the examples. We then utilize the *AG* to guide scene synthesis by using several simple and efficient operations, i.e., *replacing*, *growing* and *transfer*. The *growing* operation is especially efficient for adding a new object. These operations provide a flexible and user-friendly way to synthesize diverse scenes. To evaluate scene quality and avoid

* Corresponding author.
E-mail address: shishenghuang0@gmail.com (S.-S. Huang).

http://dx.doi.org/10.1016/j.gmod.2016.03.004
1524-0703/

low-quality scenes during the synthesis, we introduce a synthesis compatibility value to measure each synthesis operation and the quality of a resulting scene.

Our main contribution lies in the following three points:

(1) We represent a 3D interior scene as a structure graph associated with a relationship set, and introduce a furniture object matching method between scene pairs via graph matching. Our scene matching is general and efficient, which can be used for other applications besides scene synthesis.
(2) We introduce a unified structure, *Augmented Graph*, to encode all the layout information from examples, augmented from the matched structure graphs. Guided by the *AG*, we provide three simple reshuffle-based synthesis operations, i.e., *replacing*, *growing* and *transfer*, to generate diverse new scenes.
(3) We also introduce a synthesis compatibility metric to measure scene quality during the synthesis, making it efficient to filter out poor quality synthesis results.

2. Related work

It has still been a challenging problem for rapidly designing and creating 3D contents, such as shapes and 3D scenes. In recent years, continuous progresses have been made for shape processing (see [3] for more details). Here we only focus on example-based manipulation and analysis for shapes and 3D scenes.

Part-based shape synthesis. Shape synthesis by reusing parts is an efficient way to generate new shapes [4]. Jain et al. [5] provided a method to synthesize man-made objects by blending a small set of segmented shapes via recombining their object parts. Functional structure plays an important role in shape understanding. Recently, a few part-based substructures have been introduced, such as sFARR-s structure [6], Support Substructure [7] and Replaceable Substructure [8]. Based on such part-based substructures, shape synthesis with plausible results can be efficiently achieved with functional computability maintained. Evangelos et al. [9] learned a probability model from a moderate-to-large shapeset to guide shape synthesis. The data-driven part-based algorithms also show us the efficiency in 3D modeling [10,11]. Xu et al. [12] introduced an approach to generate new shapes via set evolution. Recently, Alhashim et al. [13] provided a shape blending method to synthesize new shapes by topology varying. Our reshuffle-based algorithm is inspired by these part-based shape synthesis methods, which can be extended to scene synthesis.

Scene analysis. In general, interior scene understanding is a challenging problem due to the variation in object geometry and functional arrangement. The relationship between objects can be used as a useful cue to guide scene analysis, especially for scene matching and retrieval. Object retrieval can be enhanced using the context information [14]. Fisher et al. [15] introduced an efficient method to compare scene objects using Graph Kernel defined on a relation graph. Learning based algorithms have also been introduced to synthesize scenes. For example, Fisher et al. [1] proposed a learning based method to synthesize 3D scenes from a given scene set. Su et al. [16] proposed a probabilistic scene model using object frames. In contrast, our method is not learning-based and directly synthesizes scenes from examples. Xu et al. [17] provided a method to organize heterogeneous scene collection using focal points. Recently, Liu et al. [18] provided a method to infer consistent grouping information via parsing with a probabilistic grammar learned from examples. Existing works has paid more attention on the similarity between pairs of either single objects or scenes and few works have studied the matching of furniture objects in the layout structures.

Scene reconstruction. Our work is also related to scene reconstruction. Recently there has been a significant progress on scene reconstruction from LiDAR data [19,20] and RGB-D data [21]. Xu et al. [22] presented a novel approach to reconstruct 3D scenes from user-drawn rough sketches. Scenes reconstructed from sensor data always lack semantic labels or tags, which are time-consuming to manually label or tag. Our method thus aims to handle scenes without any category labels.

Our work is closely related to [1], both aiming at synthesizing scenes by example. However, our algorithm is non-learning based, and works well for a small number of examples, reducing the complexity on constructing a large dataset of scenes. Our work also bears close resemblance to [2]: both of them are reshuffle-based scene synthesis. However, our approach is more flexible on scene structures since structures and relations in [2] are very limited.

3. Overview

Inspired by the part-based methods for shape synthesis and scene analysis (see the discussions in the previous section), we provide a structure-guided method for synthesizing 3D interior scenes from a small set of examples. Our input is a small set of exemplar 3D interior scenes. Each interior scene has been segmented into single furniture objects (Fig. 1) and oriented uprightly [23]. As discussed previously our approach does not need the furniture objects to be semantically tagged or labeled. We also assume that the facing direction of each furniture object is available (see Fig. 2). In general, the shape's orientation detection is not an easy problem on its own. We use the prior knowledge of each 3D scene to determine the facing direction of each furniture object (see Section 4 for more details).

As shown in Fig. 1, our method involves three stages. In the first stage, we extract all the relations in each scene according to the five relations to be formally introduced later. Then we represent each scene as a structure graph $G = \{V, E, A\}$ by associating a relationship set $A_e \in A$ with each edge $e \in E$. In the second stage, we perform scene matching between scene pairs based on the layout subgraphs G_L (Fig. 4 (e)). Following the scene matching scheme, we introduce a greedy method to augment all the scene graphs into a unified *Augmented Graph* (*AG*). In the last stage, we perform scene synthesis according to the three scene synthesis operations defined on the *Augmented Graph* (*AG*). During the synthesis, we use the synthesis compatibility measurement to filter out poor synthesis operations.

Next we first introduce how we extract the relations and perform scene matching via graph matching in Section 4. In Section 5, we will discuss how to augment a set of structure graphs into a final *AG* and to perform scene synthesis guided by *AG*. We will present lots of diverse results synthesized by our algorithm in Section 6 and conclude the paper in Section 7.

4. Scene matching

In this section we first show how we determine the facing direction of each furniture object. First, we compute a symmetry plane (if any) (Fig. 3) for each furniture object. The facing direction is always parallel to the symmetry plane. Users can specify the facing direction manually if none or multiple symmetry planes exist. In general, furniture objects which are located in the boundary region of a 3D scene often have facing directions pointing to the scene's center. This motivated us to assign the facing directions of such boundary objects as the directions which are parallel to their symmetry planes and point to the scene's center. Then for the rest of the furniture objects, their facing directions are assigned as the directions parallel with the symmetry planes and pointing to the nearest objects.

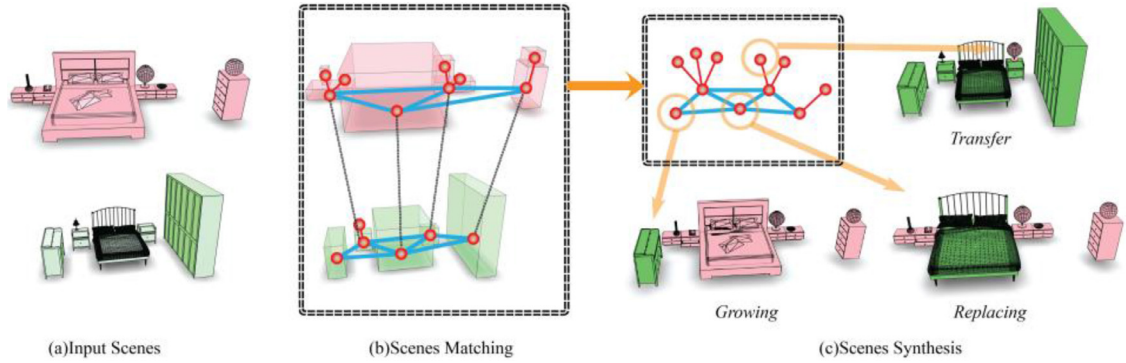

Fig. 1. (a) Given several 3D interior scenes, (b) we represent each scene as a structure graph affiliated with a relationship set and perform scene matching via graph matching on the layout subgraphs (the subgraphs colored in blue). (c) Based on the scene matching, we augment the structure graphs with an *Augmented Graph* to guide the scene synthesis using three synthesis operations, leading to plausible new scenes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

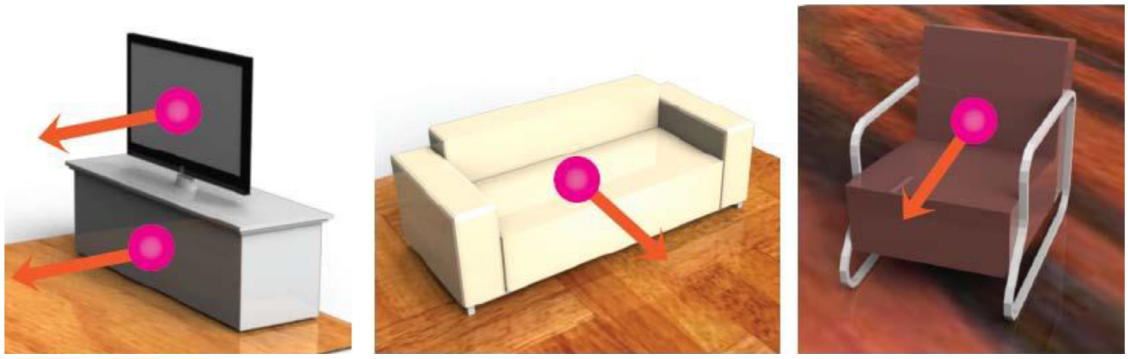

Fig. 2. The facing direction(s) of furniture objects.

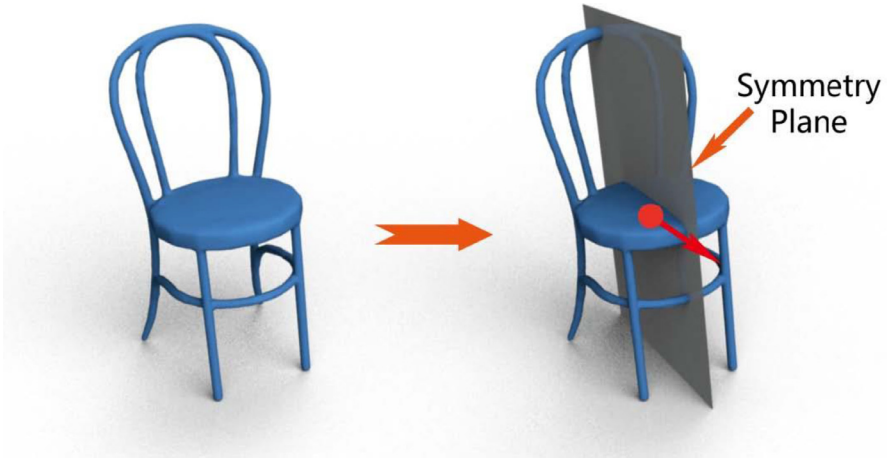

Fig. 3. An illustration for symmetry plane detection.

4.1. Graph building

We represent a given 3D interior scene as a structure graph $G = \{V, E, A\}$ associated with a relationship set, with $v \in V$ representing a furniture object in the scene, and each edge $e_{ij} \in E$ associated with a relationship set $A_{e_{ij}} \in A$ if there exists any relation between nodes v_i and v_j.

Previous methods (e.g., Fisher et al. [15], Merrell et al. [24], Yu et al. [25]) often formulate a scene as a relation graph, with each edge assigned a specific relation. However, we observed that there may exist multiple kinds of relations for one edge. For example, a *TV* is supported by a *cabinet* with one side aligned. In this condition, the *edge* between *TV* and *cabinet* has at least two relations, namely, support relation and aligned relation. This kind of condition widely exists in 3D interior scenes. So instead of formulating a scene as an ordinary relation graph, we formulate it as a structure graph associated with a relationship set. In other words, each edge $e_{ij} \in E$ may have several associated relations, thus forming a relationship set $A_{e_{ij}}$.

In the previous works various relations have been explored, e.g., the alignment relation, emphasis relation, pairwise relation, hierarchical relation [2]. However, since our input is only a small set of scenes, it is hard to apply so many various relations. Instead, we focus on only five simple relations (Fig. 4) to characterize the

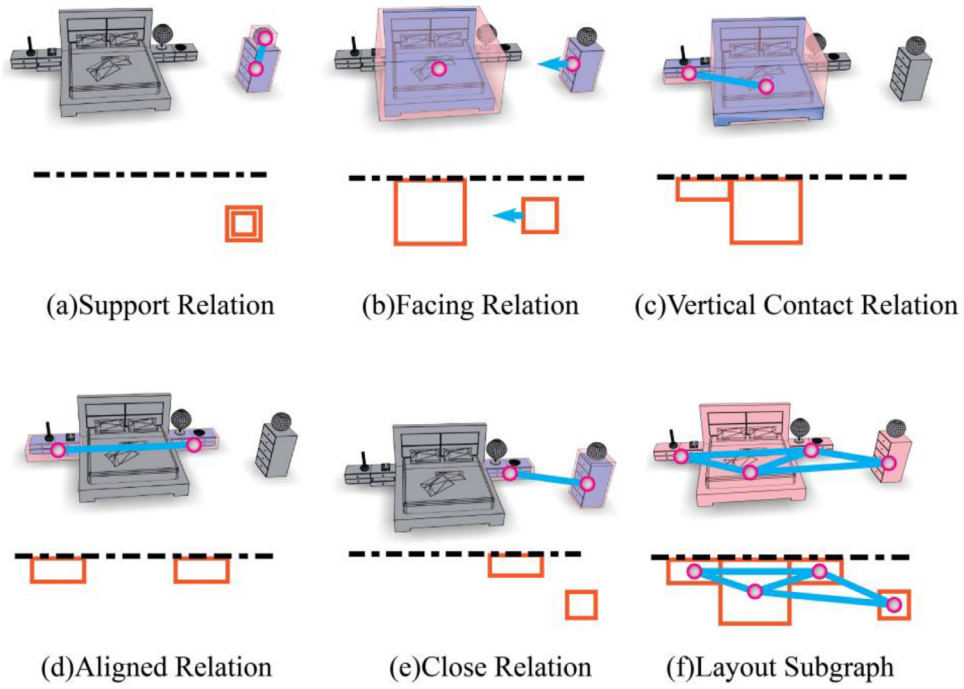

Fig. 4. The relation graph (f) defined based on five types of relation. (a) Support relation. (b) Facing relation. (c) Vertical contact relation. (d) Aligned relation. (e) Close relation. (f) The layout subgraph after removing the supported furniture objects.

placing requirements of these furniture objects, which are still descriptive and efficient. Specifically we define these fine kinds of relation as follows:

Support relation. For a pair of object nodes (v_i, v_j), if node v_i supports node v_j (Fig. 4(a)), we add an edge between v_i and v_j, and add a Support Relation (S) into edge e_{ij}'s relationship set $A_{e_{ij}}$, i.e., $A_{e_{ij}} = A_{e_{ij}} \cup \{S\}$.

Facing relation. If node v_i faces node v_j (Fig. 4(b)) according to the facing direction, we add a Facing Relation (F) to the relationship set $A_{e_{ij}}$, i.e., $A_{e_{ij}} = A_{e_{ij}} \cup \{F\}$. For the Facing Relation, we record the distance between object pair (v_i, v_j) and angle α_f between the two nodes for edge similarity measurement later.

Vertical contact relation. If node v_i vertically contacts node v_j (Fig. 4(c)), we add a Vertical Contact Relation (V) into the relationship set $A_{e_{ij}}$, i.e., $A_{e_{ij}} = A_{e_{ij}} \cup \{V\}$. In practice, the Vertical Contact Relation is extracted if node v_i is in contact with node v_j and the contact area forms a vertical plane surface. Denoting n_{ij} as the normal of contact plane surface, we record the angle α_i between n_{ij} and node v_i's facing direction, and α_j between n_{ij} and node v_j's facing direction. Then we use the average angle $\alpha_v = (\alpha_i + \alpha_j)/2$ as the contact angle for the vertical contact relation. Using the contact angle α_v, we aim at describing the *left* contact or *right* contact for a pair of involved furniture objects.

Aligned relation. If a certain side of the bounding box v_i is aligned with the corresponding side of v_j's bounding box (Fig. 4(d)), we add an Aligned Relation (A) into the relationship set $A_{e_{ij}}$, i.e. $A_{e_{ij}} = A_{e_{ij}} \cup \{A\}$. In practice, we require the object pair (v_i, v_j) to be close enough but not necessarily in contact.

Close relation. If the pair of objects (v_i, v_j) are not in contact with each other but are close to each other (Fig. 4(e)), we also add a Close Relation (C) into the relationship set $A_{e_{ij}}$, i.e. $A_{e_{ij}} = A_{e_{ij}} \cup \{C\}$. We record the distance between the pair of objects (v_i, v_j) for further similarity measurement.

Note that the furniture objects that are supported by any other furniture objects do not influence the scene layout in the 2D layout plane. They can be flexible kinds of object categories with varying geometry. It is the remaining furniture objects (other than the supported furniture objects) that decide the scene layout. Thus we remove the supported objects from the structure graph G, resulting a layout subgraph G_L. Next we will perform scene matching on the layout subgraphs to match two structure graphs mainly considering their layouts.

4.2. Scene matching on layout subgraph G_L

Let $G_L = \{V, E, A\}$ and $G'_L = \{V', E', A'\}$ be two layout subgraphs of two structure graphs G and G', respectively. Our goal is to find an optimized matching between V and V', f: $V \to V'$, that best matches the furniture objects and preserves the relations between edges $e_{ij} \in E$ and $e'_{ij} \in E'$. Mathematically, we would like to find a matching f that maximizes the following energy function:

$$E(f) = \sum_{v_i \in V} \phi_i(f(v_i)) + \lambda \sum_{e_{ij} \in E} \phi(e_{ij}, f(e_{ij})), \tag{1}$$

where $v'_{i'} = f(v_i) \in V'$ is the correspondence node and $e_{i'j'} = f(e_{ij}) \in E'$ is the correspondence edge in G'_L under the matching f, and λ is a parameter to balance the object matching energy $\phi_i(f(v_i))$ and the edge matching energy $\phi(e_{ij}, f(e_{ij}))$.

Object matching energy $\phi_i(f(v_i))$. The optimized matching needs to match similar objects as much as possible, since it is very likely that similar objects serve the same functional role in the scenes. However, since the geometry of the objects can vary a lot, similarity based on detailed shape descriptor like SDF [26] might be too restrictive. Instead, we define an object matching energy at the level of the object bounding box. More specifically, let $BOB(v_i) = \{\theta_1, \theta_2, \theta_3\}$ and $BOB(v'_{i'}) = \{\theta'_1, \theta'_2, \theta'_3\}$ denote the

bounding box scale in the three main axes for nodes v_i are $v'_{i'}$, with $f(v_i) = v'_{i'}$. We have

$$\phi_i(f(v_i)) = \exp\{(1.0 - \prod_i^3 g(\theta_i, \theta'_i))/\sigma_g\}, \quad (2)$$

where $g(x, y) = 1 + |x - y|/|x + y|$ and σ_g is a parameter which will be discussed in Section 6.

Edge matching energy $\phi(e_{ij}, f(e_{ij}))$. We define the edge matching energy based on the relations. For the Aligned Relation (A), we use the number of such relations that both relationship sets share to serve for the similarity of the edge pair $(e_{ij}, e'_{i'j'})$. More specifically, we define

$$d^A(e_{ij}, e'_{i'j'}) = |A_{e_{ij}} \cap A_{e'_{i'j'}} \cap \{A\}|, \quad (3)$$

as the Aligned Relation similarity.

For the Vertical Contact Relation (V), we use the contact angle α_v to measure the similarity, i.e.,

$$d^V(e_{ij}, e'_{i'j'}) = \exp\{-|\alpha_{v_{ij}} - \alpha_{v'_{i'j'}}|/\pi\} I\{A_{e_{ij}} \cap A_{e'_{i'j'}} \cap \{V\}\}, \quad (4)$$

where $I(.)$ is the indicator function with $I(\varnothing) = 0$ and $I(a \neq \varnothing) = 1$.

For the Facing Relation (F), we use the facing relation angle α_f to measure the similarity:

$$d^F(e_{ij}, e'_{i'j'}) = \exp\{-|\alpha_{f_{ij}} - \alpha_{f'_{i'j'}}|/\pi\} I\{A_{e_{ij}} \cap A_{e'_{i'j'}} \cap \{V\}\}. \quad (5)$$

For the Close Relation (C), we use the edge distance $d_{e_{ij}}$ and define

$$d^C(e_{ij}, e'_{i'j'}) = \exp\{-|d_{e_{ij}} - d_{e'_{i'j'}}|/\sigma_c\} I\{A_{e_{ij}} \cap A_{e'_{i'j'}} \cap \{C\}\}, \quad (6)$$

as the Close Relation similarity.

Then we sum them all to measure the Edge Matching Energy,

$$\begin{aligned}\phi(e_{ij}, f(e_{ij})) = {} & d^V(e_{ij}, e'_{i'j'}) + d^A(e_{ij}, e'_{i'j'}) \\ & + d^F(e_{ij}, e'_{i'j'}) + d^C(e_{ij}, e'_{i'j'}), \end{aligned} \quad (7)$$

where $e'_{i'j'} = f(e_{ij})$.

We can use $x_{ik} \in \{0, 1\}$ to denote whether $v_i \in V$ matches $v'_k \in V'$: if $f(v_i) = v'_k$ then $x_{ik} = 1$, otherwise $x_{ik} = 0$. In this way the energy function $E(f)$ can be reformulated as

$$\begin{aligned} E(f) &= \sum_i \sum_k \phi_i(v'_k) x_{ik} + \lambda \sum_{e_{ij} \in E} \sum_{e'_{kl} \in E'} \phi(e_{ij}, e'_{kl}) x_{ik} x_{jl} \\ &= XWX, \end{aligned} \quad (8)$$

with $X = \{x_{11}, x_{21}, ..., x_{m1}, ..., x_{m-1n}, x_{mn}\}$, where $m = |V|, n = |V'|$, and W is the matching matrix.

Graph matching. We would like to find an optimal matching f^* to maximize the matching energy function such that $f^* = argmax_f E(f)$. This is the so-called Graph Matching problem. Matching two graphs is a fundamental problem, and has been widely studied [27–30]. We use the SMAC algorithm [29] to solve this optimization. The SMAC algorithm finds the optimized matching by using an SVD decomposition of the matrix W and forcing the eigenvalues into binary X to obtain the matching result. In our case, we choose the one-to-one attributed matching, meaning that each node can have only one matching node, if any. In practice, we use the maximum eigenvalue as the matching confidence for each matching. If the matching confidence exceeds a threshold γ ($\gamma = 0.8$ in all our experiments), we use the matching result; otherwise, we withdraw it. Fig. 5 shows a toy example illustrating how we perform the scene matching via graph matching.

5. Scene synthesis guided by augmented graph

Once furniture objects are matched using the scene matching method described in the previous section, we can synthesize new scenes by simply reshuffling the matched furniture objects. However, it is quite often that two layout graphs have different numbers of objects. Therefore, there may exist some object nodes without any correspondence. Thus simply reshuffling between corresponding furniture objects cannot always lead to diverse new scenes. Inspired by the work in [13] for blending shapes, we suggest to 'merge' the structure graphs into a so-called *Augmented Graph* (*AG*) based on the matched layout graphs. With the *Augmented Graph*, we define three reshuffle-based synthesis operations, i.e., *replacing*, *growing* and *transfer* (to be described in more detail later), to synthesize diverse and plausible new scenes. Note that our condition is different from [13]: their matching between components is not fully automatic while ours is automatic. Besides, our matching is a one-to-one matching. In contrast theirs can be either one-to-one or one-to-many, since one functional part can be served as multiple components. We thus do not need to define *split node* for node augmentation or the related operations for edge augmentation.

Augment two matched layout graphs. We augment two matched layout graphs $G_L = \{V, E, A\}$ and $G'_L = \{V', E', A'\}$ into an *Augmented Graph* $\hat{G}_L = \{\hat{V}, \hat{E}, \hat{A}\}$, with node augmentation, edge augmentation and attribution augmentation, as illustrated in Fig. 6.

Node augmentation. For a node $v \in G_L$, if v has the corresponding node $v' \in G'_L$, then we merge node pair (v, v') into a node $\hat{v}$ in $\hat{G}_L$, i.e., $\hat{V} = \hat{V} \cup \hat{v}$; otherwise, if v does not have correspondence in G'_L, we just add v into $\hat{G}_L$, i.e., $\hat{V} = \hat{V} \cup v$. We do the same for nodes $v' \in G'_L$ to obtain the augmented nodes for $\hat{G}_L$.

Edge augmentation. For a node pair $(\hat{v}_i, \hat{v}_j) \in \hat{G}_L$, if there exist edges in either graph G_L or G'_L, we add an edge to this node pair, i.e., $\hat{E} = \hat{E} \cup \hat{e}_{ij}$.

Attribution augmentation. For an edge $\hat{e}_{ij} \in \hat{G}_L$, we merge the relationship sets from the corresponding edges $e_{ij} \in G_L$ and $e'_{i'j'} \in G'_L$ into an augmented relationship set, i.e., $\hat{A}_{\hat{e}_{ij}} = \hat{A}_{\hat{e}_{ij}} \cup A_{e_{ij}} \cup A'_{e'_{i'j'}}$.

After the two layout graphs are augmented, the supported object nodes are then added to the Augmented Graph correspondingly to get the final Augmented Graph (*AG*).

Augment a set of structure graphs. Since our input is a small set of 3D scenes, we need to consider the problem of augmenting a set of scene graphs. This problem has not been explored in [13], since their work considers only a pair of shapes. For n scene graphs $\{G_1, G_2, ..., G_n\}$, our goal is to augment them into a single *Augmented Graph* (*AG*), i.e., $\{G_1, G_2, ..., G_n\} \Rightarrow AG$. Our strategy is to first augment their layout subgraphs one by one greedily and then obtain the final *AG* by adding the supported nodes respectively. We define the matching similarity $s(G_i, G_j)$ for a pair of graphs (G_i, G_j) as the number of nodes matched. For the layout subgraphs $\{G_{L_1}, G_{L_2}, ..., G_{L_n}\}$, we first augment graph pair (G_{L_i}, G_{L_j}) with the highest matching similarity, i.e., $(G_{L_i}, G_{L_j}) \Rightarrow \hat{G}^1_L$. Next a new layout subgraph G_{L_k} with the highest matching similarity to $\hat{G}^1_L$ is augmented with $\hat{G}^1_L$ to form a further Augmented Graph $\hat{G}^2_L$, i.e., $(\hat{G}^1_L, G_{L_k}) \Rightarrow \hat{G}^2_L$. When matching $(\hat{G}^1_L, G_{L_k})$ via graph matching, the Object Matching Energy is calculated as the maximizing node pair from the original graphs, i.e., $\phi_{\hat{v}_l}(v'_{l'} \in G_{L_k}) = max\{\phi_{v_l \in G_{L_i}}(v'_{l'} \in G_{L_k}), \phi_{v_l \in G_{L_j}}(v'_{l'} \in G_{L_k})\}$. The Edge Matching Energy is also calculated as the maximizing edge matching energy from the original edge pairs, i.e., $\phi(\hat{e}_{ij} \in \hat{G}^1_L, e'_{kl} \in G_{L_k}) = \max\{\phi(e_{i'j'} \in G_{L_i}, e'_{kl} \in G_{L_k}), \phi(e_{i'j'} \in G_{L_j}, e'_{kl} \in G_{L_k})\}$. The above augmentation steps

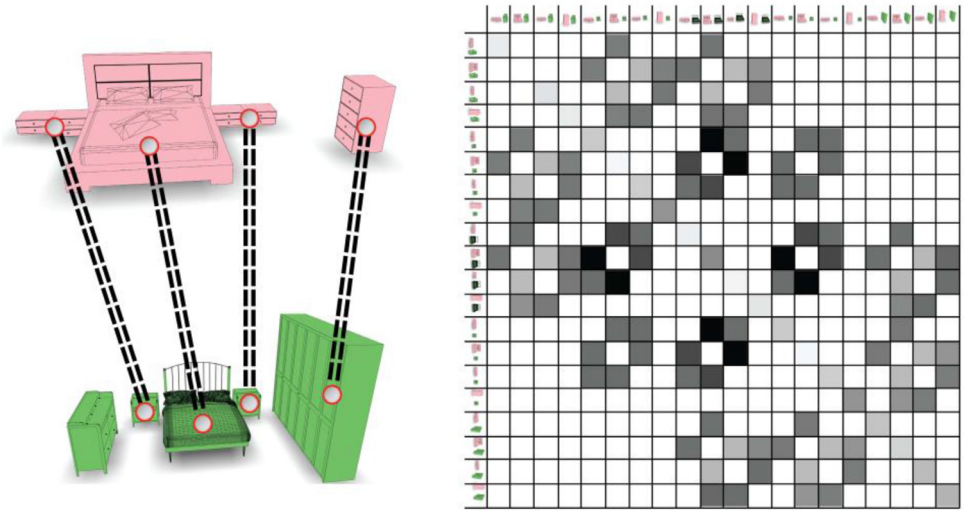

Fig. 5. An illustration of how to perform graph matching between the layouts of two scenes (Fig. 1). Left: the matching results with the corresponding objects linked with dashed lines. Right: the similarity matrix W when $\lambda = 0.5$ with the darker colors meaning the higher similarity.

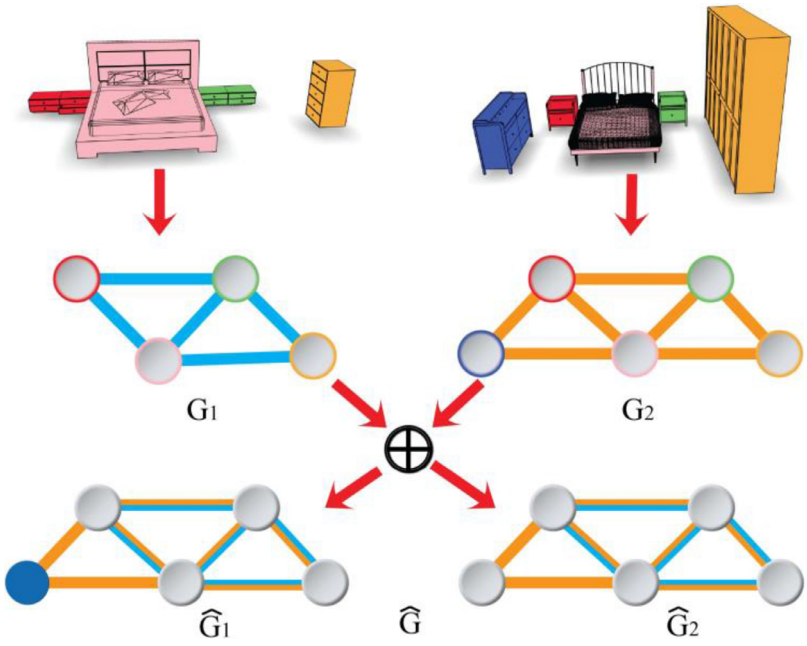

Fig. 6. An example showing how we augment two layout subgraphs, with node augmentation, edge augmentation and attribution augmentation. Note that in each layout subgraph, the nodes are colored the same as the original furniture objects. The blue node in $\hat{G}_1$ is a newly added node after augmentation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

are repeated until all the layout subgraphs are augmented to one graph $\hat{G}_L^{n-1}$. Please refer to the supplementary materials for the detailed augmentation process of the 4 data sets shown in Fig. 9.

In the end the supported object nodes are added to $\hat{G}_L^{n-1}$ according the original structure graphs to get the final Augmented Graph (*AG*). For any structure graph G_i, any furniture object node $v \in G_i$ has one and only one corresponding furniture object node in *AG*. The *AG* encodes all the layout information from the examples, and thus can be used to guide scene synthesis.

5.1. AG guided scene synthesis

After the scenes are augmented to an *Augmented Graph*, we enable AG guided scene synthesis by introducing three reshuffle-based synthesis operations on *AG*, i.e., *replacing*, *growing* and *transfer*. Suppose we augment a set of scenes $\{G_1, G_2, ..., G_n\}$ into one Augmented Graph *AG*. We design the *replacing* and *growing* operations for nodes belonging to the layout subgraphs of *AG*, and design the *transfer* operation for supported nodes in *AG* as shown in Fig. 1(c).

1. Replacing. Furniture objects that correspond to the same nodes in the *AG* can replace with each other. More specifically, if two furniture objects $v \in G_i$, $v' \in G_j$ correspond to the same node $\hat{v} \in AG$, then we can perform the *replacing* operation between v and v'. For example, to replace v with v', we first rotate v' such that v''s facing direction aligns with v's. If A_v has the Aligned Relation, v' is then adjusted to align with the related furniture objects.
2. Growing. For a 3D interior scene G_i, if G_i's layout subgraph G_{L_i} does not have a node corresponding to one node $\hat{v} \in AG$. For example, if $\hat{v} \in AG$ does not have any correspondence in G_{L_i}, we

can consider *growing* a new node in G_{L_i} that corresponds to $\hat{v} \in AG$ to synthesize new scenes from G_i.

If we grow a node $\hat{v} \in AG$ for G_i, we consider a set of neighbors of $\hat{v} \in AG$ and denote it as $\mathcal{N}_{\hat{v}}$. If none of the nodes in $\mathcal{N}_{\hat{v}}$ has any correspondence in G_i, i.e., $\mathcal{N}_{\hat{v}} \cap G_i = \varnothing$, then growing becomes unfeasible, since there is limited layout information to reconstruct $\hat{v}$ in G_i. Otherwise, we use the relations of the edge between $\mathcal{N}_{\hat{v}} \cap G_i$ and $\hat{v}$ to reconstruct $\hat{v}$ in G_i. During reconstruction, we try to preserve the contact angle α_v for the Vertical Contact relation, α_f for the Facing relation, and the related distance for the Close relation to get the best layout position. If there exists any Aligned relation, $\hat{v}$ is adjusted to align the related nodes. Lastly, $\hat{v}$ may contain several furniture objects from the original scenes. Once $\hat{v}$'s position in G_i is determined, each time we can place the original furniture objects to that place, obtaining several diverse new scenes.

3. Transfer. The transfer operation is performed for supported nodes in *AG*. If several supported furniture nodes are supported by one node $\hat{v} \in AG$, since $\hat{v}$ contains several corresponding furniture nodes, the supported furniture nodes can be transferred to be supported by other furniture nodes (corresponding to $\hat{v}$) in other scenes. When placing a supported furniture object in new scenes, the domain supported surface of the supporting furniture object is detected as done in [1]. The supported furniture object is then placed onto the domain supported surface with the same place as in the original scene. If the supporting furniture object does not have such a domain supported surface (e.g., a *sofa*'s top supported surface cannot support a furniture object) we do not perform the *transfer* operation.

We perform scene synthesis starting from each scene one by one. Each time we select an input scene G_i and randomly perform the three synthesis operations to generate new scenes.

Scene synthesis compatibility. During the synthesis, we compute a synthesis compatibility value for each synthesis operation and the entire new scene, to measure the compatibility of the synthesis operation and the quality of the newly generated scene. Our synthesis compatibility metric considers the environment of the newly placed furniture object.

For the *replacing* operation, we use the scale changing value of the bounding box between the newly placed furniture object v and the original object v'. As the object matching energy described in Section 4, let $BOB(v) = \{\theta_1, \theta_2, \theta_3\}$, $BOB(v') = \{\theta_1', \theta_2', \theta_3'\}$ denote the bounding box scale in the three main axes for v and v'. We use

$$\zeta_r(v) = \exp\{(1.0 - \prod_i^3 g(\theta_i, \theta_i'))/\sigma_r\}, \tag{9}$$

as the synthesis compatibility metric for the *replacing* operation.

For the *transfer* operation, we consider the furniture objects v and v' that support the to-be-transferred furniture objects, and use

$$\zeta_t(v) = \exp\{(1.0 - \prod_i^3 g(\theta_i, \theta_i'))/\sigma_t\}, \tag{10}$$

as the synthesis compatibility metric for the *transfer* operation.

For the *growing* operation, we consider the neighbors of the to-be-grown furniture object v. Assume v's original structure graph is G_i and we grow it in G_j (we denote v in G_j as v'). v and v''s respective neighbors are $\mathcal{N}_v$ and $\mathcal{N}_v'$, and $f : \mathcal{N}_v \to \mathcal{N}_v'$ is the matching. We use

$$\zeta_g(v) = \sum_{\bar{v} \in \mathcal{N}_v} \exp\{-(d(v, \bar{v}) - d(v', f(\bar{v})))/\sigma_g\}, \tag{11}$$

as the synthesis compatibility metric for the *growing* operation, where $d(.)$ is the Euclidean distance. If v' intersects with some furniture object in G_j, we set $\zeta_g = 0$.

For a new scene u that is generated by a serials of operations, we use

$$K(u) = \prod_i \zeta_r(v_i) \prod_j \zeta_g(v_j) \prod_k \zeta_t(v_k), \tag{12}$$

as the synthesis compatibility value to evaluate the quality of new scene u. The $K(u) \in [0, 1]$ is higher, the quality is better.

6. Results

Data. We tested our method on example sets of 3D interior scenes coming from Xu et al. [22]'s open dataset. That dataset contains varying categories of interior scenes such as the living room, meeting room etc. Some representative scene synthesis results are shown in Fig. 9. The scenes were segmented into meaningful single furniture objects. We extracted the structure graphs, and matched them via graph matching. Please refer to the details of the graph matching in the supplementary materials. After augmenting them into an *AG*, we synthesized them using the three synthesis operations, and got dozens of new synthesis scenes with diverse geometry and arrangement variation. For each synthesis operation, we only used the operations with synthesis compatibility values above 0.8. The synthesis was terminated if the new scenes' synthesis compatibility is below 0.2.

Parameters. Each time for scene synthesis, we suggested an input scene set containing only $2 \sim 4$ scenes. In all of our experiments, we set $\lambda = 0.5$ for graph matching. When calculating the synthesis compatibility values, we set σ_r, σ_t and σ_g as the diagonal length of the scene's bounding box.

Comparing with Xie's method. The most related method to our algorithm is Xie et al.'s [2]. We compared our algorithm with Xie et al.'s by evaluating the quality of the synthesized scenes by the two approaches. For the four data sets shown in Fig. 9, we obtained the synthesis results by Xie et al.'s method and sorted them according to the assessment scores in the descending order. Our results were also sorted according to the synthesis compatibility values in the descending order. Then we selected the top-40 results from the two synthesis result sets for each input set, thus obtaining in total 320 synthesis results with half coming from Xie et al.'s method and half coming from ours. Later, we asked 20 participants to give a score for each synthesis result in the scale from 1 (low quality) to 5 (high quality). Each synthesis result was randomly shown to each participant, and each participant gave a score by being asked: "Is the new synthesis result's layout consistent with the original one?" and "Is the new synthesis result appropriate in the real world?". We computed a top-K average score for the i-th data set as

$$Avg(i, K)_1 = \frac{1}{K} \sum_j a_{ij}^1, Avg(i, K)_2 = \frac{1}{K} \sum_j a_{ij}^2, \tag{13}$$

where a_{ij}^1 and a_{ij}^2 were the j-th result's score in the i-th data set for the Xie et al.'s method and ours, respectively. Thus we obtained 8 average top-K score curves as shown in Fig. 7. From the curves, our results' scores were always higher in all of the 4 data sets than Xie's method [2]. We performed t-tests for the scores of the 4 data sets, and obtained the p-values: 0.031, 0.009, 0.023 and 0.019 with all the p-values less than 0.05. Thus our results were significantly better than Xie et al.'s.

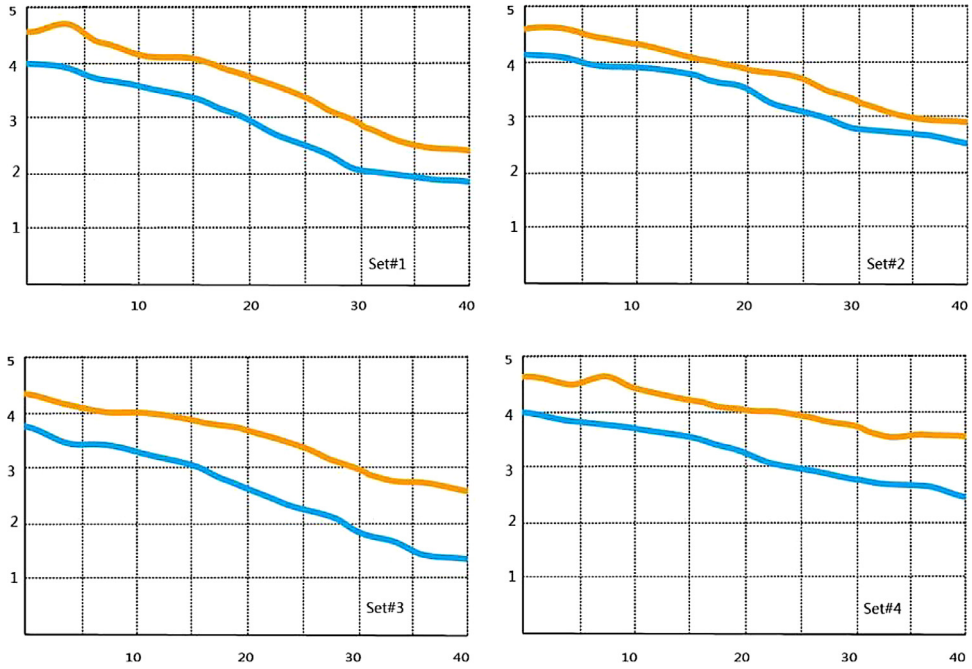

Fig. 7. The average top-K score curves of the 4 data sets by using Xie et al.'s method [2] and ours. In each set, the orange one is ours and the blue one is Xie et al.'s. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

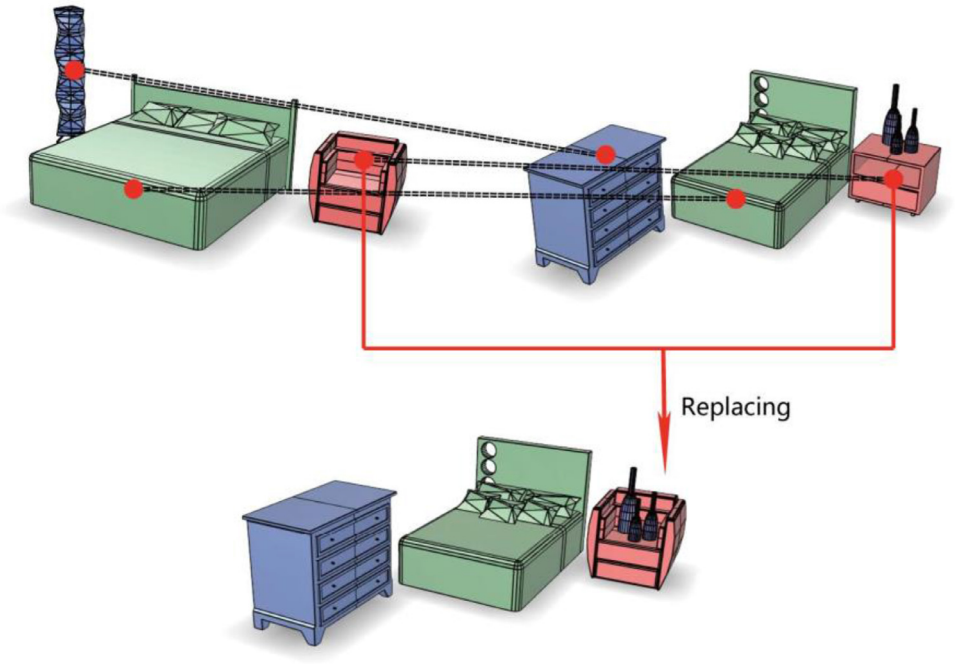

Fig. 8. A failure case.

Time complexity. Given n segmented 3D scenes, we need to perform scene matching for $O(n)$ times to obtain the final *Augmented Graph*. It only took about $1-2$ s to perform each scene matching and less than 1 s for each synthesis operation, measured on a PC with an Intel Core 2 Duo 2.4 GHz CPU and 16G RAM. The facing direction estimation for each furniture object can be time consuming, due to the slow symmetry plane estimation. The average time used for the facing direction estimation of each scene was about 30 s in our experiments. So once the given scenes are segmented and oriented, our algorithm is fast and efficient.

Limitations. Our method suffers from the following limitations. First, due to the lack of prior knowledge which might be learned via learning algorithms, our method cannot guarantee that every synthesis operation is functionally plausible, especially for the *transfer* operation. Second, currently we do not consider some high-level relations, such as symmetry and concentric relations for layout reconstruction, which will be fitted using our current relations. Besides, when performing the scene matching between the un-labeled furniture objects, our bounding-box based Object Matching Energy may not be so descriptive than those using labeled or semantic information. As shown in Fig. 8, due to the

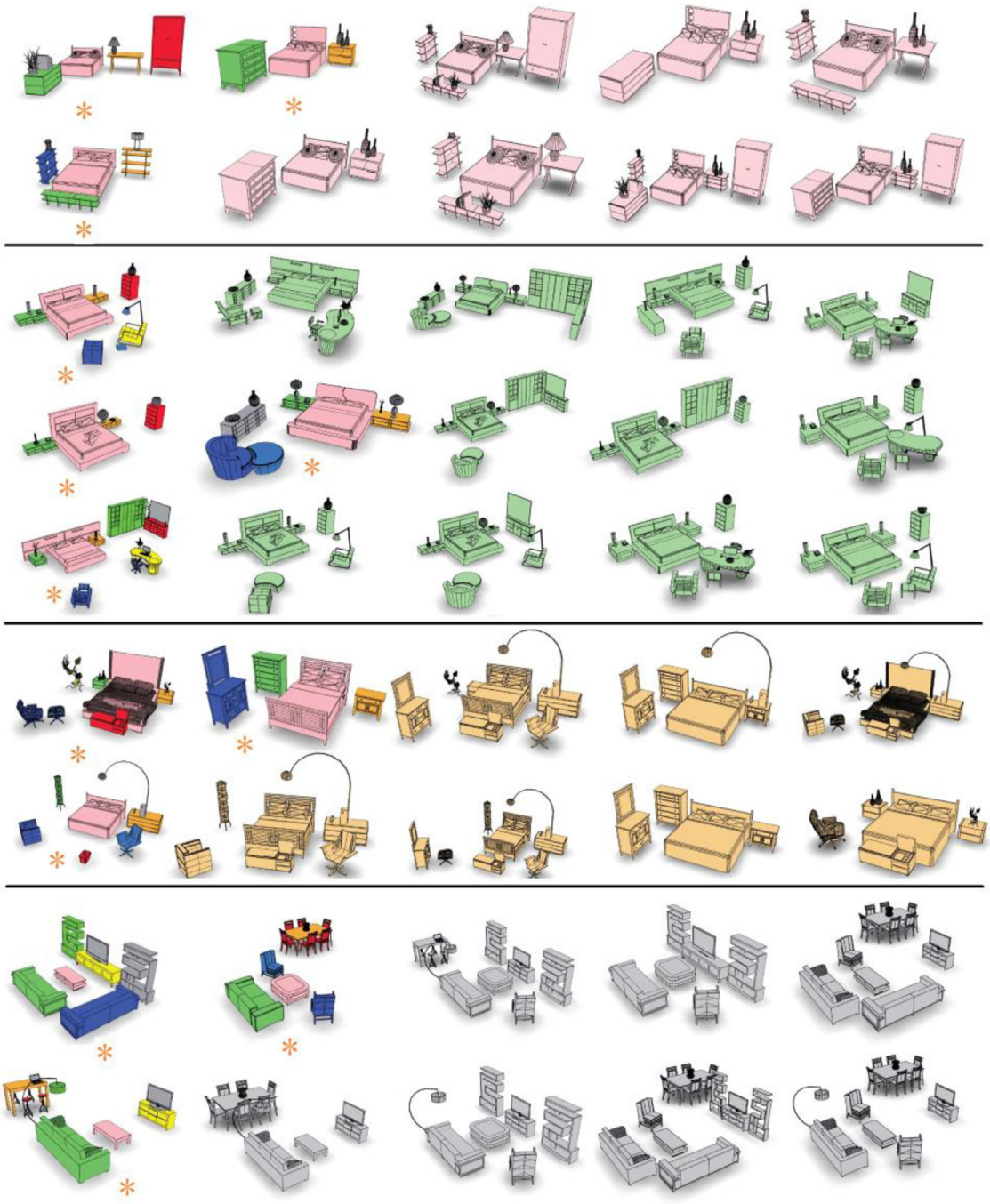

Fig. 9. Gallery of scene synthesis results for some typical sets of scenes. Each scene (marked with $*$) is segmented into meaningful furniture objects, shown in different colors. We get dozens of new scenes with varying geometry and arrangement and randomly place some of them in this figure with the small synthesis compatibility value. Please find more results in the supplementary materials. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

limited layout information of the two scenes, the established matching between the two scenes is not very appropriate. Performing the replacing operation, our method obtained a less successful result. In other words, our scene synthesis results rely on the scene matching quality, which is another limitation of our method.

Discussions. In this paper, we only consider one-to-one matching when performing scene matching. The key observation for this strategy is that we would like to obtain the exact layout matching between scene pairs. However, there may be similar objects between scene pairs, thus leading to one-to-many matching. Considering such a one-to-many constraint can lead to more matching information. However, it will also introduce more complexity for graph augmentation and the subsequent scene synthesis. Another issue is about the room structures such as doors and windows. In this paper, we focus on interior 3D scenes without using any door or window information, which is widely studied by previous works [1,2]. We leave these two considerations as future works.

7. Conclusion

In this paper, we introduced a method to synthesize scenes directly from unlabeled 3D interior scenes. Each scene is formulated

as a structure graph associated with a relationship set. We establish a one-to-one matching between the layout subgraphs of the structure graph pairs via graph matching, and augment them into a unified structure *Augmented Graph*. Based on the *Augmented Graph*, we define three synthesis operations, i.e., *replacing*, *growing*, *transfer*, providing a flexible way to synthesize new, nontrivial scenes. Our scene matching approach is general and might be used to analyze scenes for other applications. In the future, we would like to improve the graph augmentation for example using an advanced graph matching approach [31] to extract the prior knowledge especially the structure information from scene data, and use it to guide scene synthesis and other geometry applications.

Acknowledgments

This work was supported by the Natural Science Foundation of China (Project Number 61521002, 61120106007), Research Grant of Beijing Higher Institution Engineering Research Center, and Tsinghua University Initiative Scientific Research Program. Hongbo Fu was partially supported by grants from the Re425 search Grants Council of HKSAR, China (Project No. 113513, 11204014 and 11300615).

References

[1] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, P. Hanrahan, Example-based synthesis of 3d object arrangements, ACM SIGGRAPH Asia 2012 papers, SIGGRAPH Asia '12, 2012.
[2] H. Xie, W. Xu, B. Wang, Reshuffle-based interior scene synthesis, in: Proceedings of the 12th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry, in: VRCAI '13, ACM, New York, NY, USA, 2013, pp. 191–198.
[3] N. Mitra, M. Wand, H.R. Zhang, D. Cohen-Or, V. Kim, Q.-X. Huang, Structure-aware shape processing, in: SIGGRAPH Asia 2013 Courses, in: SA '13, ACM, New York, NY, USA, 2013, pp. 1:1–1:20.
[4] T.A. Funkhouser, M.M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, D.P. Dobkin, Modeling by example, ACM Trans. Graph. 23 (3) (2004) 652–663.
[5] A. Jain, T. Thormählen, T. Ritschel, H.-P. Seidel, Exploring shape variations by 3d-model decomposition and part-based recombination, Comput. Graph. Forum (Proc. Eurograph. 2012) 31 (2) (2012).
[6] Y. Zheng, D. Cohen-Or, N.J. Mitra, Smart variations: functional substructures for part compatibility, Comput. Graph. Forum (Eurograph.) 32 (2pt2) (2013) 195–204.
[7] S.-S. Huang, H. Fu, L. Wei, S.-M. Hu, Support substructures: support-induced part-level structural representation, IEEE Trans. Vis. Comput. Graph. (2015), doi:10.1109/TVCG.2015.2473845.
[8] H. Liu, U. Vimont, M. Wand, M. Cani, S. Hahmann, D. Rohmer, N.J. Mitra, Replaceable substructures for efficient part-based modeling, Comput. Graph. Forum 34 (2) (2015) 503–513, doi:10.1111/cgf.12579.
[9] E. Kalogerakis, S. Chaudhuri, D. Koller, V. Koltun, A probabilistic model for component-based shape synthesis, ACM Trans. Graph. 31 (4) (2012) 55:1–55:11, doi:10.1145/2185520.2185551.
[10] S. Chaudhuri, E. Kalogerakis, L.J. Guibas, V. Koltun, Probabilistic reasoning for assembly-based 3d modeling, ACM Trans. Graph. 30 (4) (2011) 35, doi:10.1145/2010324.1964930.
[11] S. Chaudhuri, V. Koltun, Data-driven suggestions for creativity support in 3d modeling, in: ACM SIGGRAPH Asia 2010 Papers, in: SIGGRAPH ASIA '10, ACM, New York, NY, USA, 2010, pp. 183:1–183:10, doi:10.1145/1866158.1866205.
[12] K. Xu, H. Zhang, D. Cohen-Or, B. Chen, Fit and diverse: set evolution for inspiring 3d shape galleries, ACM Trans. Graph. 31 (4) (2012) 57:1–57:10, doi:10.1145/2185520.2185553.
[13] I. Alhashim, H. Li, K. Xu, J. Cao, R. Ma, H. Zhang, Topology-varying 3d shape creation via structural blending, ACM Trans. Graph. 33 (4) (2014) 158:1–158:10.
[14] M. Fisher, P. Hanrahan, Context-based search for 3d models, in: ACM Transactions on Graphics (TOG), 29, ACM, 2010, p. 182.
[15] M. Fisher, M. Savva, P. Hanrahan, Characterizing structural relationships in scenes using graph kernels, in: ACM Transactions on Graphics (TOG), 30, ACM, 2011, p. 34.
[16] H. Su, A.W. Yu, Probabilistic modeling of scenes using object frames, Sci. China Inf. Sci. 58 (3) (2015) 1–13, doi:10.1007/s11432-014-5151-3.
[17] K. Xu, R. Ma, H. Zhang, C. Zhu, A. Shamir, D. Cohen-Or, H. Huang, Organizing heterogeneous scene collections through contextual focal points, ACM Trans. Graph. 33 (4) (2014) 35.
[18] T. Liu, S. Chaudhuri, V.G. Kim, Q. Huang, N.J. Mitra, T. Funkhouser, Creating consistent scene graphs using a probabilistic grammar, ACM Trans. Graph. 33 (6) (2014) 211:1–211:12, doi:10.1145/2661229.2661243.
[19] Y.M. Kim, N.J. Mitra, D. Yan, L.J. Guibas, Acquiring 3d indoor environments with variability and repetition, ACM Trans. Graph. 31 (6) (2012) 138.
[20] L. Nan, K. Xie, A. Sharf, A *search-classify* approach for cluttered indoor scene understanding, ACM Trans. Graph. 31 (6) (2012) 137.
[21] K. Chen, Y. Lai, Y. Wu, R. Martin, S. Hu, Automatic semantic modeling of indoor scenes from low-quality RGB-D data using contextual information, ACM Trans. Graph. 33 (6) (2014) 208.
[22] K. Xu, K. Chen, H. Fu, W. Sun, S. Hu, Sketch2scene: sketch-based co-retrieval and co-placement of 3d models, ACM Trans. Graph. 32 (4) (2013) 123.
[23] H. Fu, D. Cohen-Or, G. Dror, A. Sheffer, Upright orientation of man-made objects, ACM Trans. Graph. 27 (3) (2008) 42:1–42:7.
[24] P. Merrell, E. Schkufza, Z. Li, M. Agrawala, V. Koltun, Interactive furniture layout using interior design guidelines, ACM Trans. Graph. 30 (4) (2011) 87.
[25] L. Yu, S.K. Yeung, C. Tang, D. Terzopoulos, T.F. Chan, S. Osher, Make it home: automatic optimization of furniture arrangement, ACM Trans. Graph. 30 (4) (2011) 86.
[26] R. Gal, A. Shamir, D. Cohen-Or, Pose-oblivious shape signature, IEEE Trans. Vis. Comput. Graph. 13 (2) (2007) 261–271.
[27] R. Zass, A. Shashua, Probabilistic graph and hypergraph matching, in: 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA, 2008.
[28] S. Gold, A. Rangarajan, A graduated assignment algorithm for graph matching, IEEE Trans. Pattern Anal. Mach. Intell. 18 (4) (1996) 377–388.
[29] T. Cour, P. Srinivasan, J. Shi, Balanced graph matching, in: Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006, 2006, pp. 313–320.
[30] M. Leordeanu, M. Hebert, A spectral technique for correspondence problems using pairwise constraints, in: 10th IEEE International Conference on Computer Vision (ICCV 2005), 17-20 October 2005, Beijing, China, 2005, pp. 1482–1489.
[31] P. Morrison, J.J. Zou, Inexact graph matching using a hierarchy of matching processes, Comput. Vis. Media 1 (4) (2015) 291–307.