SCENEHGN: Hierarchical Graph Networks for 3D Indoor Scene Generation with Fine-Grained Geometry

Lin Gao, Jia-Mu Sun, Kaichun Mo, Yu-Kun Lai, Leonidas J. Guibas and Jie Yang*

Abstract—3D indoor scenes are widely used in computer graphics, with applications ranging from interior design to gaming to virtual and augmented reality. They also contain rich information, including room layout, as well as furniture type, geometry, and placement. High-quality 3D indoor scenes are highly demanded while it requires expertise and is time-consuming to design high-quality 3D indoor scenes manually. Existing research only addresses partial problems: some works learn to generate room layout, and other works focus on generating detailed structure and geometry of individual furniture objects. However, these partial steps are related and should be addressed together for optimal synthesis. We propose SCENEHGN, a hierarchical graph network for 3D indoor scenes that takes into account the full hierarchy from the room level to the object level, then finally to the object part level. Therefore for the first time, our method is able to directly generate plausible 3D room content, including furniture objects with fine-grained geometry, and their layout. To address the challenge, we introduce functional regions as intermediate proxies between the room and object levels to make learning more manageable. To ensure plausibility, our graph-based representation incorporates both vertical edges connecting child nodes with parent nodes from different levels, and horizontal edges encoding relationships between nodes at the same level. Our generation network is a conditional recursive neural network (RvNN) based variational autoencoder (VAE) that learns to generate detailed content with fine-grained geometry for a room, given the room boundary as the condition. Extensive experiments demonstrate that our method produces superior generation results, even when comparing results of partial steps with alternative methods that can only achieve these. We also demonstrate that our method is effective for various applications such as part-level room editing, room interpolation, and room generation by arbitrary room boundaries.

Index Terms—3D indoor scene synthesis, deep generative model, recursive neural network, variational autoencoder, graph neural network, relationship graphs, fine-grained mesh generation

1 INTRODUCTION

3D indoor scenes are useful for a wide range of applications, such as smart digital houses, virtual reality/argument reality, robotics, virtual room planning, etc. Therefore, high-quality 3D indoor scenes are in high demand. However, they are compositionally complex and individual furniture objects often contain rich geometric details. Creating high-quality 3D indoor scenes is not only time-consuming but also requires expertise for designers. So research that can automate 3D scene generation would be highly valuable.

Although existing research works have considered some of the problems related to 3D indoor scene generation, they usually focus only on partial steps, rather than the whole process. For example, a large body of work addresses indoor scene layout generation, including traditional data-driven models (*e.g.* [1, 2]) and more recent deep generative models (*e.g.* [3, 4]). Although such works can generate diverse and plausible furniture layouts, they do not generate furniture geometry at the same time, and usually only retrieve existing furniture shapes from a repository. However, shape geometry and layout are related and treating these two steps separately may produce suboptimal results. Moreover, as furniture can have a large variety in terms of both geometry and structure, retrieving shapes inevitably restricts the diversity of furniture shapes that may appear in the synthesized scenes. Some other works (*e.g.* [5, 6, 7]) explicitly consider part-aware 3D shape generation, which can be applied to furniture objects to synthesize objects with various structures and/or geometry details, but such works are restricted to individual objects, rather than at the 3D indoor scene level.

While it is possible to apply these individual steps in sequence to synthesize 3D indoor scenes, it is difficult to ensure consistency and compatibility between objects, and if the early stage output is treated as a constraint to the next stage, it may also unnecessarily restricts the diversity of the generated scenes. A key observation is that the layout and geometry of furniture are entangled. For example, when a chair is placed underneath a table, its geometry cannot be arbitrary as the chair must fit in the space there. Hence, the geometry must be compatible with its layouts. To address these challenges, we propose to *jointly* model the layout and

* *Corresponding Author is Jie Yang (yangjie01@ict.ac.cn).*
- *Lin Gao, Jia-Mu Sun, and Jie Yang are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, and also with the University of Chinese Academy of Sciences, Beijing, China. E-mail: {gaolin, sunjiamu21s, yangjie01}@ict.ac.cn*
- *Kaichun Mo and Leonidas Guibas are with Stanford University. E-mail: kaichunm@stanford.edu, guibas@cs.stanford.edu*
- *Yu-Kun Lai is with the School of Computer Science & Informatics, Cardiff University, U.K. E-mail: LaiY4@cardiff.ac.uk*

Manuscript received April 19, 2005; revised August 26, 2015.

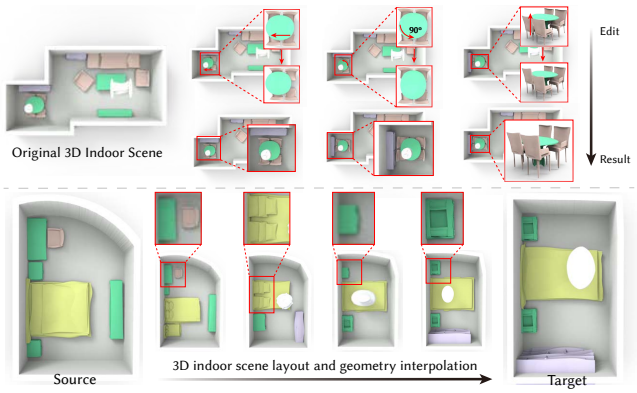


Fig. 1: Our deep generative model SCENEHGN encodes the indoor scene across multiple conceptual levels: the room, functional regions, furniture objects, and even fine-grained object part geometry. We utilize edges, including our proposed hyper-edges to strengthen the relations between objects during decoding. This enables some interesting applications, such as room editing with part-level geometry and scene interpolation. Our approach allows the entire 3D room to be represented and synthesized. Based on this, we can achieve part geometry editing (at different scales) in the scene, such as rigid transformation in a functional region and non-rigid deformation at the part level. Meanwhile, our network is capable of capturing the smooth latent space near similar scenes for plausible scene interpolation.

fine-grained geometry, and synthesize an entire 3D room using a single deep generative model. This has significant advantages: by treating it as a joint optimization problem, our approach is able to generate diverse indoor scenes with rich geometric details, while ensuring object room relationships, contextual relationships between objects, and consistency/compatibility of content and style for furniture objects.

However, there are many challenges due to the complexity of data and the problem. While a room is naturally hierarchical: it contains multiple furniture objects, and each furniture object can also be modeled using a part-based approach [8] for flexible structure and fine-grained geometry, a room may contain a large number of objects (*e.g.* a room in the 3D-FRONT [9] dataset contains up to 188 objects), which makes learning difficult. Considering that for larger rooms, smaller groups of objects are more likely to be related to each other (*e.g.* several chairs surrounding a table), we introduce *functional regions* (*e.g.* dining regions, sofa regions), as intermediate proxies to bridge the gap between the room and objects. When generating indoor scenes, the room shape is usually given. We represent the room shape flexibly by deforming a unit square, and the coding of the deformation is treated as a condition for indoor scene generation. To ensure plausible synthesis, it is also essential to take rich relationships into account. These include vertical relationships in the hierarchy: regions must be within the room boundary, objects should be within the region boundary, etc., and also horizontal relationships constraining objects at the same level: *e.g.* symmetry of objects, adjacency between objects, the symmetry between object parts, etc. To achieve this, our hierarchical graph network is a recursive neural network (RvNN) based variational autoencoder (VAE) that covers 4 levels, namely: room, regions, objects and parts, with carefully designed edges between graph nodes to enforce constraints. Training of such a large network can also be challenging, and we propose a multi-stage training strategy to ensure training is stable and effective.

As no publicly available datasets [9, 10, 11, 12] contain rich 3D indoor scenes at the part level, we use a hybrid dataset combining 3D-FRONT [9] data (object-level 3D indoor scenes), with PartNet [13] data which contains objects with detailed part-level annotation, where each object in the 3D-FRONT dataset is replaced with the most similar PartNet object to make the obtained 3D scene with part-level annotations. Extensive experiments show that our method is superior to baselines (even when comparing the partial steps they are designed for), and allow a range of interesting applications, including reconstruction, generation, completion, and interpolation.

In summary, the main contributions of our paper are:

1) To the best of our knowledge, this is the first deep generative model capable of synthesizing an entire room with plausible furniture, including object layout and fine-grained object geometry.
2) To achieve this, we propose a hierarchical graph network based on an RvNN VAE, that covers 4 levels from a room to objects and object parts. We introduce functional regions and carefully designed both vertical and horizontal edges in the graph, including hyper-edges to represent relationships among multiple objects, to ensure effective learning and plausible generation. We further encode room boundary as a deformed square and incorporate it as the condition for a controlled generation.
3) Extensive experiments show that our method outperforms existing baselines, and supports a range of applications from indoor scene synthesis to editing and completion.

2 Related Work

In this section, we first briefly review the compositional scene representations and approaches in the literature. Then, we summarize existing research efforts on leveraging hierarchical graph representations for learning 3D generative models.

Indoor Scene Representations. Due to the complexity and diversity of realistic 3D scenes, researchers have pursued compositional scene representations, such as scene graphs that explicitly model the entities in the scene (*e.g.* rooms, objects, walls) and the rich relations among them (*e.g.* adjacency, symmetry). Scene graphs have been shown to be powerful to generate 2D images [14, 15]. Recent works [3, 16, 17, 18, 19, 20] have been exploring leveraging scene graphs to guide 3D scene generation. To name a few, Luo *et al.* [17] learned to generate furniture layout in a room given a scene graph as input. House-GAN [19] tackled the floor plan generation problem given room graph user inputs as constraints. In our method, we use a hierarchical graph scene representation that not only exploits the advantages of

scene graph representations for encoding rich relationships among scene entities at the same level, but also leverages hierarchical decomposition to abstract nodes at different levels (*e.g.* regions, objects, object parts).

3D scenes can be hierarchically decomposed into multiple semantic levels of content nodes: regions, objects and object parts. Some parent-child inclusion constraints, such as objects that should lie within the room boundary, should hold. Huang *et al.* [21] used holistic scene grammars to parse scenes as hierarchical structures for reconstruction from a single image. Armeni *et al.* [22] introduced a scene hierarchical representation from the entire building to rooms and objects. GRAINS [4] proposed to represent the objects and their relationships in a four-wall room as a hierarchy and employed recursive neural networks [23, 24] to conduct learning on such a representation. Shi *et al.* [25] explored to predict 3D scene layout using hierarchy denoising recursive autoencoders. Our work introduces a hierarchical graph scene representation with four compositional levels: the room, functional regions, objects, and object parts, augmented with horizontal edges forming smaller graphs among sibling nodes at each level.

Indoor Scene Synthesis. There are many works exploring furniture layout generation; we refer to survey papers [26, 27] for a comprehensive discussion. Next, we give a brief review of this task.

The usual setting of works in this line is to retrieve 3D models from a given database and predict the model positions for the generation of semantically and functionally realistic indoor scenes. Before deep learning gained its popularity, a substantial body of works [1, 28, 29, 30, 31, 32, 33, 34, 35] has explored this problem and constantly pushes the frontier. For real indoor scenes, these works [36, 37, 38] perform dense and realistic reconstruction of the real scene via a scanning approach using robots. More recently, deep learning-based methods further boost performance. PlanIT [3] introduced an image-based generative model reasoning over relation graphs. Ritchie *et al.* [39] and Wang *et al.* [40] proposed to learn image-based deep convolutional generative models. GRAINS [4] leveraged a recursive neural network to layout furniture within a room with four walls. Zhang *et al.* [41] solved the task by training a generative adversarial network to achieve a free-form generation without any floor constraint via a hybrid representation. SceneFormer [42] achieved faster realistic 3D scene generation with self-attention of transformers, which can predict a sequence of object locations conditioned on the room layout or text descriptions. ATISS [43] also uses autoregressive transformers for automatic layout synthesis, scene completion, and object suggestion under some constraints. For learning the location recommendation, Zhou *et al.* [44] used neural message passing to predict the probability of newly added objects for learning the spatial and structural relationships between objects within an incomplete indoor scene. Liu *et al.* [45] introduced a visual context-aware graph generation network to learn global implicit relations on the in-game residential home complex. Furthermore, Sync2Gen [46] uses the learned parametric prior distribution to regularize the unrealistic indoor scenes from feed-forward neural models. There are also other works learning to produce furniture layout under language [47], activity [48], human [49] and action [50] constraints. Different from these works, our approach learns to generate novel 3D furniture shapes instead of retrieving existing models from a database and also extends to a more fine-grained level of object parts.

Hierarchical Graph Networks. Designing neural architectures for processing hierarchical data is highly non-trivial. Recent works have shown promising results using recursive neural networks (RvNN) [23, 24, 51] and Tree-LSTM [52] to encode tree-structured natural language sentences and natural scenes.

Recently, we witnessed a surge of success using RvNNs to model 3D shapes and scenes. GRASS [5] first proposed to represent 3D shapes as a binary hierarchy of parts and employed an RvNN variational autoencoder [53] for a 3D shape generative model. Follow-up works [54, 55] extended GRASS for reconstructing the part hierarchy from a single image or 3D point cloud. Using the PartNet dataset [13], which provides large-scale hierarchical shape part segmentation annotations, StructureNet [56] and its follow-ups [8, 56, 57] extended the GRASS binary hierarchy into more flexible n-ary hierarchies and augmented the representation with adjacent and symmetric part relations forming local graphs among sibling nodes. In this paper, we further explore using hierarchical graph networks, specifically RvNNs in this paper, for 3D scene generation, where we model each 3D scene as an n-ary hierarchy with graphs.

3 Overview

We propose SceneHGN, a hierarchical graph network for 3D indoor scene mesh generation that is end-to-end differentiable across multiple conceptual levels: rooms, functional regions, furniture objects, and even fine-grained object part geometry. Given room boundary layouts as inputs, SceneHGN learns smooth and continuous conditional latent spaces for generating diverse and novel indoor room scenes with semantically plausible furniture layouts and shape geometry with part-level details. Figure 2 presents a high-level architecture overview of our hierarchical conditional variational autoencoder (VAE) for 3D scene generation.

Hierarchical Graph Representation. We represent every 3D indoor scene as a hierarchical tree with multiple levels: the room⇒functional region⇒objects⇒object parts. We find that modeling the complicated space of highly compositional 3D indoor scenes at reasonable levels is necessary for producing high-quality results.

Aside from the naturally defined object and part levels, we additionally introduce an abstract level of functional regions (*e.g.* dining region, sleeping region) to better organize the furniture objects within big rooms. Figure 3 presents example regions commonly seen in 3D indoor rooms. The functional region level serves as an intermediate proxy for the hierarchy and helps group smaller sets of furniture shapes within a big room. All the objects within a room are thus more consistently organized by an explicit hierarchy: room⇒region⇒objects.

Objects in the scenes also have their own compositional part structures: their constituent parts and part relationships. Hence, in our approach, we follow [6, 8] to organize an object as an n-ary hierarchical graph tree with its

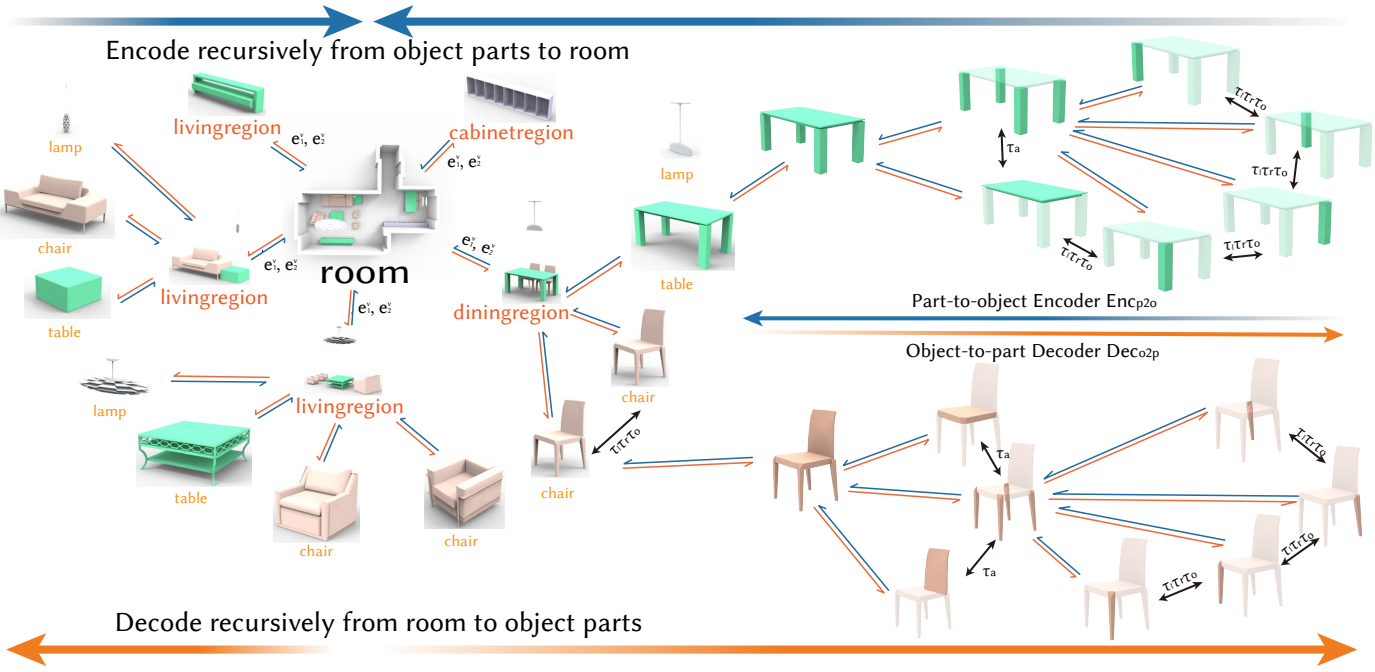


Fig. 2: **Hierarchical Scene Representation.** Our scene hierarchy has four conceptual levels: the room root node, functional regions, objects, and object parts. To train the recursive autoencoder, we use an encoder network to summarize the features in a bottom-up fashion and a decoder network that reconstructs the scene hierarchy from the room root node to regions to objects and finally to object parts in a top-down manner. We also model the rich edge relationships at different levels in this process to enforce the validity of the generated scene structures.

part geometry and edge relations. Note that the object nodes in the entire scene hierarchy not only encode the n-ary hierarchical graph tree to represent the geometry and structure of the object, but also encode the spatial parameters in the context of the room furniture layout.

In the scene hierarchy, we also encode rich relationships among nodes at different levels. Besides the parent-child vertical relationships naturally defined in the hierarchical representation, the nodes at the same level also have horizontal edge constraints, such as between shapes and walls (*e.g.* a bed that is well-aligned with the wall), between two objects (*e.g.* two symmetric nightstands), and even among multiple objects (*e.g.* a dining table and four surrounding chairs, two nightstands co-aligned with the bed). Not only do we adopt the binary edge relationships that are commonly used in previous works [4, 6], we also propose to enforce hyper-edge constraints among multiple objects.

Hierarchical Graph Network. We train a conditional recursive variational autoencoder to learn a smooth latent space for scene generation. Our framework consists of a room layout encoder, a scene hierarchy encoder, and a scene hierarchy decoder. The room layout encoder takes the deformation gradients of the floor boundary as input and extracts a vector that is used as the floor boundary condition for the decoder. The scene hierarchy encoder maps the indoor scene hierarchies from the room level to the functional region level, the object level, and finally down to the object part geometry level into a common latent space hierarchically and recursively. In contrast, the scene hierarchy decoder performs an inverse mapping which decodes a latent vector and floor boundary condition vector into the furniture layouts and shapes with detailed geometry in a top-down manner.

During the encoding process, furniture objects are firstly encoded as object features with a pre-trained DSG-Net [8]. Then, combining the object features and their spatial location information in the regions, the region-level features are extracted. Finally, a single room-level root feature summarizes all region information together with the region-level spatial arrangement. The scene hierarchy decoder inversely reconstructs the indoor scene hierarchy from a room-level latent code to more fine-grained levels in a top-down manner. During the encoding and decoding of the training, several graph message passing operations are performed to capture rich edge relations and constraints between object and object, between room boundary and object, and among multiple objects. Figure 4 visualizes the adopted binary edge relationships for scene generation and our proposed hyper-edge constraints among multiple objects.

Paper Organization. In the following sections, we first describe the detailed definitions for our indoor scene hierarchical representation in Sec. 4. We will introduce our concrete node level designs and rich edge relationships, including our proposed hyper-edges constraining n-ary part relationships beyond $n = 2$, which are important for indoor scene generation. Then, we present our hierarchical architecture designs for learning hierarchical 3D indoor scene generation in Sec. 5. Learning to represent the input room boundary layouts as conditions to our hierarchical framework, we introduce FloorNet which learns to encode and decode input room boundary layouts represented as 2D deformation representation in vertex neighborhoods. We also discuss the key roles of the introduced functional regions and how to model the rich relationships among

objects and room boundaries. Thanks to our learned latent space and decoder, in Sec. 6, we show that our framework enables some interesting applications, such as indoor scene editing, completion, and conditional generation with some constraints (scene generation from 3D box layouts).

4 Hierarchical Graph Representation

Below, we detail our node and edge compositional and relational designs of our proposed hierarchical graph representation.

4.1 Hierarchical Node Decomposition

Given a 3D indoor scene, we represent it as a hierarchical graph structure. There are multiple conceptual levels in the hierarchy: the room, functional regions, furniture objects, and object part geometry. Besides the natural concepts of objects and object parts defined in PartNet [13], we additionally propose a new conceptual level of functional regions to further divide the objects in big rooms into smaller clusters of objects for producing 3D scenes with higher quality via the parameter-free method. We describe the detailed node definitions as follows.

Regions. According to the functionality of furniture shapes, functional regions divide the whole scene into smaller groups of objects for a more consistent and learning-friendly scene hierarchical representation. Different types of rooms usually have very disparate functional regions. For example, one may have entertainment and living regions in a living room, and have sleeping and cabinet regions in a bedroom. Explicitly modeling the region semantics not only provides many meaningful semantic labels as parts of the scene generation results, but it is also beneficial for modeling a large number of objects within each room. Due to the capacity limit of the RvNNs, we cannot practically model a big graph of a huge number of objects in a single room as the children nodes of a single parent node (*e.g.* close to hundreds of objects for the biggest room in the 3D-FRONT dataset [9]).

In our implementation, we divide each scene into smaller functional regions by running a spatial clustering algorithm DBSCAN [58], which is a density-based and non-parametric clustering algorithm, where the number of clusters is self-adaptive. All objects can be grouped and organized according to spatial closeness by the algorithm, which is often correlated to their functionality. The functionality of the region is determined by the object with the largest area in the cluster. Within each detected functional region, many functionally related objects are often clustered together (*e.g.* dining table and its surrounding dining chairs), as illustrated in Figure 3. We also detect and correct automatic labeling errors manually for some unreasonable divisions of indoor scene space.

In summary, for a 3D scene S, we have $S = \{\langle g_1, g_2, \cdots, g_K \rangle, \mathbf{R}_{region}\}$, where g_i means the i^{th} clustered group (a functional region), with $\mathbf{R}_{region}$ means the horizontal relations between these functional regions. We will describe the region relationships in Sec. 4.2.

Objects. After we divide the scene space into functional regions, there are many furniture objects within each region.

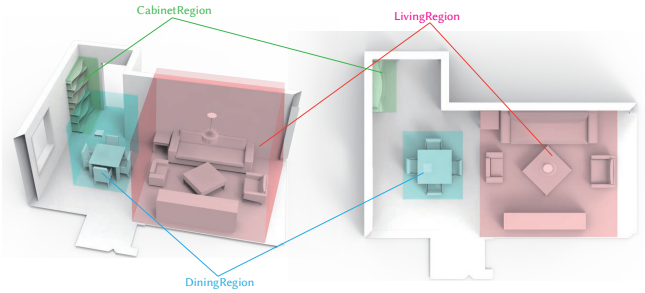


Fig. 3: **Functional Region Visualization.** In the figure, a whole scene is divided into three functional regions including a Cabinet Region, a Dining Region, and a Living Region, which are highlighted in different colors. The separation is conducted by a spatial clustering algorithm DBSCAN [58], which is a density-based and non-parametric clustering algorithm, where the number of clusters is self-adaptive. We can see that an indoor scene can be divided reasonably.

Each shape $O_i \in g_k, 1 \leq k \leq K$ is described by a mesh geometry, along with its semantic object category and its spatial location within the region.

Formally, we define $g_i = \{\langle O_1, O_2, \cdots, O_M \rangle, \mathbf{R}_{object}\}$, where O_i means the i^{th} object in the functional region, with $\mathbf{R}_{object}$ denotes the horizontal relations among objects that belong to one functional region. We will describe the region relationships in Sec. 4.2.

Object Parts. We use the PartNet [13] shape part hierarchy to decompose every 3D shape into the semantically consistent part hierarchy, organized as an n-ary hierarchical graph tree structure that covers different levels of part instances ranging from coarse-grained to fine-grained parts.

Namely, each object O_i is decomposed into some parts $\mathbf{P}^j_{O_i}$ organized by a hierarchy structure $\mathbf{H}_{O_i}$ and binary/n-ary relations $\mathbf{R}_{O_i}$ between parts. Each part $\mathbf{P}^j_{O_i}$ has a pre-defined semantic label and detailed mesh geometry $\mathbf{G}^j_{O_i}$. We follow the exact same set of edge relationships for the part nodes within an object as in StructureNet [6].

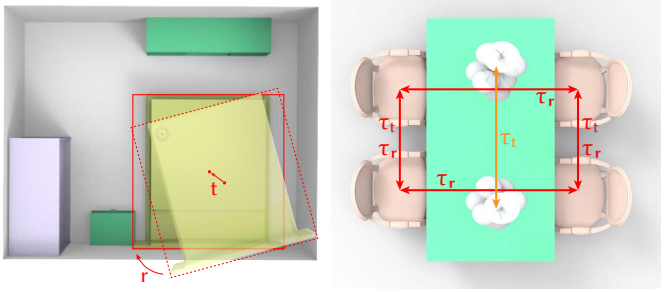
4.2 Edge Relationships

We consider two kinds of node relationships in a scene hierarchy.

- vertical edges for the relationships between parent and child nodes;
- horizontal edges among the nodes at the same level.

Extending previous works that only considered binary edges [4, 6] for modeling relationships, we propose hyper-edges to model the n-ary relationships among $n > 2$ parts. In many indoor scenes, multiple objects may hold one n-ary constraint, such as parallel collinearity and n-fold rotational symmetry (*e.g.* dining chairs surround a round table). We experimentally find that introducing hyper-edges makes our generated indoor scenes more reasonable and realistic.

A notably related work to us is GRAINS [4]. In the GRAINS scene representation, they described the relations between objects and walls with the three relations: supporting, surrounding, and co-occurrence. Different from



(a) room-object edges (b) object-object edges

Fig. 4: **Two Types of Binary Edges between Objects.** We illustrate the two types of binary edges at the object level of our hierarchy. In (a), we show a binary edge example of the first kind which is defined between the room wall and an object. It encourages the object to locate within the boundary of the room and align with the room boundary. In (b), another type of binary edge describes the spatial relationship between two objects. For example, any pair of the four chairs have rich symmetry relationships of different kinds.

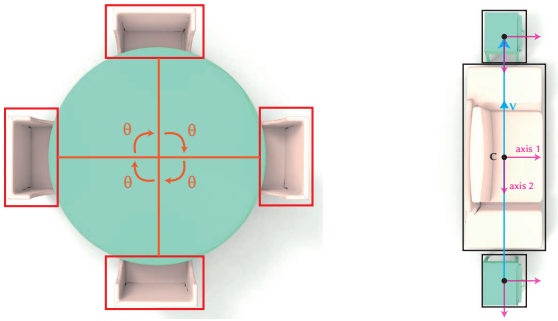


Fig. 5: **Illustration of Hyper-edges.** We define two types of hyper-edges that exist across multiple objects: rotation and parallel. A rotation hyper-edge indicates that objects are rotated around a center, and a parallel hyper-edge indicates objects are placed collinearly.

GRAINS, we use more accurate relations to describe how to organize these furniture objects, including edges between two objects: adjacency (τ_a), translational symmetry (τ_t), reflective symmetry (τ_r), and rotational symmetry (τ_o); edges among multiple objects: parallel collinearity (τ_{np}) and n-fold rotational symmetry (τ_{nr}). Experiments in Sec. 6 show that our proposed set of relationships works better than GRAINS.

4.2.1 *Vertical Edges*

The vertical edges represent the relationships between parent nodes and children nodes, which are naturally described by the multiple levels of node concepts in the hierarchy:

- the room root node comprises of many functional regions;
- every region contains many furniture objects;
- each object is further composed of object parts at different granularity.

In order to synthesize a realistic and reasonable indoor scene, we must make the generated objects compatible with the room boundary. So, we propose two additional types of vertical edges connecting the room root node to the object level:

- $\mathbf{e}_1^v$: the oriented bounding box of an object may have to align with the room boundary in some scenes;
- $\mathbf{e}_2^v$: the generated object in a scene must be located within the room boundary walls.

We find that such skip-linked vertical edges help regularize the validity of the generated scene meshes.

4.2.2 *Binary Horizontal Edges*

The horizontal edges are defined on the nodes at the same level, to describe the rich relations and constraints among sibling nodes of a parent node. In this work, we consider binary horizontal edges between two objects and between two object parts. Inspired by previous works [6, 8] on shape generation, we define four types of binary edges, including adjacency (τ_a), translational symmetry (τ_t), reflective symmetry (τ_r), and rotational symmetry (τ_o). For the adjacency relationship, we define two parts as adjacent if their smallest distance is below $0.05 \times \bar{r}$, where $\bar{r}$ is the average bounding sphere radius of the two parts. For the symmetry relationships, we follow the method from [59] to detect τ_t, τ_r, τ_o. Such binary edges are automatically detected from the input training scene data at the object level. We directly adopt the object part relationships in the previous works [6, 8]. Note that there may exist multiple relationships between two nodes. For example, in Figure 4 (b), the two chairs on one side of the table has both the translational symmetry (τ_t) and the reflective symmetry (τ_r).

4.2.3 *N-ary Hyper-edges*

We find that binary relations are not enough to describe the complex object layouts, since some realistic scenes may have more complicated relationships that happen among more than two nodes. For example, in Figure 5 (a), besides the illustrated extensive set of binary relationships, it would be beneficial to consider a 5-ary hyper-edge relationship constraining that the four chairs surround the central dining table. Similar hyper-edge relationships may also be helpful, such as the example in Figure 5 (b) where the two nightstands and the bed should have their oriented bounding boxes in parallel to each other. Thus, in this work, we introduce two types of hyper-edge relationships:

- n-fold rotational symmetry $\mathbf{e}_1^{hyper}$: *e.g.* the dining table is surrounded by some dining chairs;
- parallel collinearity $\mathbf{e}_2^{hyper}$: *e.g.* multiple box objects are in parallel to each other and their centers may be collinearly aligned.

Our hyper-edges are detected at object level and within a functional region, where the number of objects is larger than 2. The relationships between two objects are represented by binary edges. For another reason, if we only consider the relations between any two nodes, a dense graph will be constructed for the network learning, which is very hard for training and conducting message-passing operations.

N-fold Rotational Symmetry. Consider N objects $\mathbf{O} = \{O_1, \cdots, O_N\}$. They are in an *N-fold rotational symmetry hyper-edge* if and only if they satisfy:

$$\begin{aligned} &\exists p,\ \forall O_i \in \mathbf{O} \text{ where } i \leq N-1, \\ &CD\left(O_{i+1}, Rot\left(p, \frac{2\pi}{N}\right) \times O_i\right) \leq \epsilon_R \end{aligned} \tag{1}$$

and

$$CD\left(O_1, Rot\left(p, \frac{2\pi}{N}\right) \times O_N\right) \leq \epsilon_R \tag{2}$$

Here, CD indicates the Chamfer Distance between two objects. $Rot\,(p, \theta)$ denotes a rotation matrix that rotates an object around an axis that is parallel to the world up-axis and passes through the point p by the angle θ (in radians). ϵ_R is a constant threshold. Note that the point p in Equation 1 and Equation 2 must be the same. We show an example of multiple objects satisfying an N-fold rotational hyper-edge in Figure 5 (a).

Parallel Collinearity. Given N objects $\mathbf{O} = \{O_1, \cdots, O_N\}$, they satisfy a *collinearly parallel hyper-edge* if and only if

- the two main axes of the 2D oriented bounding boxes of all objects are parallel to each other (we exclude the world up-axis here as all objects are placed on the ground floor);
- they satisfy the following equation:

$$\begin{aligned} &\exists \mathbf{v},\ \forall O_i \in \mathbf{O} \text{ where } i \leq N-1, \exists d \in \mathbb{R}, \\ &||C_{i+1} - (d\mathbf{v} + C_i)||^2 \leq \epsilon_T \end{aligned} \tag{3}$$

where $\mathbf{v}$ is an arbitrary vector, d is an arbitrary non-negative real number, C_i is the center of the oriented bounding box of O_i, and ϵ_T is a constant threshold. Figure 5 (b) illustrates an example of collinearly parallel objects.

Different from superstructures (Hub and Spokes, Chains) in PlanIT [3], since our representation takes the part into consideration, the hyperedges are detected by calculating the object part bounding box instead of the bounding box of the whole shape. Further, the orientation of the object is involved in the detection: 1. the orientation of all objects that satisfy the *Parallel Collinearity* must be the same; 2. the orientation of all objects that satisfy the *N-fold Rotational Symmetry* must point to the same center object.

5 Hierarchical Graph Network

Our network is a conditional RvNN [23, 24] VAE [53] on the hierarchical scene graph representation defined in the previous section, with many edge losses to encourage more realistic and plausible structural relations in the generated scene.

Our SceneHGN takes the scene hierarchy S from the room root node down to object part geometry as inputs, with an additional input room boundary b_i as the condition. The room boundary b_i is mapped into a conditional feature f_{b_i} by a floor encoder Enc_{floor}, while the scene hierarchy S is also mapped into a latent vector f_S by the encoder of SceneHGN Enc_{SceneHGN}. Then, the conditional room boundary feature f_{b_i} and the encoded scene hierarchy feature f_S are concatenated together, which is subsequently fed into the decoder of SceneHGN Dec_{SceneHGN} to reconstruct

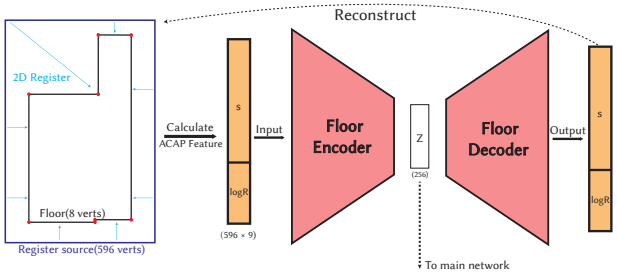


Fig. 6: **Floor VAE.** We train a separate Variational AutoEncoder for encoding floor boundaries. Specifically, we register a 2D ring of vertices onto the input floor boundary map, and then calculate the ACAP feature [60] on the registered ring structure. Finally, the VAE maps the ACAP feature into a latent vector which will serve as a condition for scene generation.

the input scene hierarchy S. To train the VAE generative model, we add a regularization, using the KL-Divergence, on the latent space to map all scenes onto a standard Gaussian distribution, from which we can smoothly sample novel scenes and interpolate between given scenes.

Since we adopt the object part hierarchy from DSG-Net [8], the part geometry encoder Enc_{PG} and part geometry decoder Dec_{PG} are following DSG-Net. The Enc_{PG} takes the part geometry information $G_{P_i} = (X_{P_i}, c_{P_i})$, including deformation gradients X_{P_i} of each part of object O_j, its center c_{P_i}, and its structural information S_{P_i}, and maps to a latent embedding feature $f_{P_i} = Enc_{PG}(G_{P_i}, S_{P_i})$. Inversely, the Dec_{PG} maps the latent feature f_{P_i} and its structural information S_{P_i} back-into deformation gradient space $\hat{G}_{P_i} = (\hat{X}_{P_i}, \hat{c}_{P_i}) = Dec_{PG}(f_{P_i}, S_{P_i})$.

Below, we focus on introducing our room boundary layout and scene hierarchy VAEs, along with the training strategy and loss terms.

5.1 Room Boundary Layout VAE

Our goal is room generation given an arbitrary room boundary which is topologically isomorphic with a ring-shaped boundary. For generating reasonable indoor rooms and modeling the relationships between the objects and room boundaries, the detailed geometry of the room boundaries needs to be effectively represented. For achieving this goal, we train a FloorNet VAE, consisting of an encoder Enc_{floor} and a decoder Dec_{floor}, to map the floor boundary of a room to a latent space. We propose to use the deformation features of a 2D ring of vertices for the representation of the floor boundary geometry. Any closed floor boundary can be represented as a deformed 2D unit square boundary. In our paper, the unit square boundary consists of 596 vertices and 596 edges. The architecture of the proposed FloorNet is shown in Figure 6.

The floor boundary in our dataset is a 2D ring represented by 596 vertices and 596 edges. Just for the illustration purpose, for example, the floor in Figure 6 (left) consists of 8 vertices and 8 edges. To efficiently and accurately encode the floor boundary (especially for sharp corners), we apply the 2D non-rigid registration technique [61] rather than vertex

coordinates to deform a source 2D mesh with 596 vertices to fit the shape of the floor boundary. Then, we calculate the ACAP deformation gradients [60] on the registered 2D mesh. For every vertex, we finally extract a 6-dimensional feature $s \in \mathbb{R}^6$ and a 3-dimensional feature $\log R \in \mathbb{R}^3$, which represent scaling/shear and rotation respectively.

The 596×9 feature matrix is then fed into the encoder of our FloorNet. The key component of the FloorNet is a Graph Convolutional Network, where we treat the whole registered floor boundary as a cycle graph and perform convolution operation on it to extract 2D mesh features. After two convolutional operations, the features pass through an MLP which outputs the latent vectors encoding the room floor boundary. We perform 2 iterations for this message to pass to neighboring vertices to learn the angle between adjacent edges for sharp corners. The decoder is basically the inverse of the encoder to map the latent code back into the ACAP deformation gradients of room boundary. We use another Graph Convolutional Network to iteratively decode the vertex locations of the 2D ring structure.

5.2 Scene Hierarchy Encoder

Our encoder consists of two parts: a recursive encoder Enc_{p2o} from the part geometry to object and another recursive encoder Enc_{o2r} from the object to the whole room level. We directly follow the design of Enc_{p2o} in DSG-Net [8]. Below, we focus on describing our Enc_{o2r} design that learns to map objects, through functional regions, and finally to the room root node.

For each object node, there are three types of information stored in it: the structural and geometric information f_{O_i} extracted using Enc_{p2o} for the objects, the placement parameters $Pos_{O_i} \in \mathbb{R}^7$ and the one-hot vector of semantic category label l_{O_i}. The Pos_{O_i} parameters include the center position $c_{O_i} \in \mathbb{R}^3$, the scales $s_{O_i} \in \mathbb{R}^3$, and the orientation $r_{O_i} \in \mathbb{R}$ around the world up-axis. The object feature encoder Enc_{obj} encodes the above information together into an object latent code $f_{O_i}^{obj}$.

$$f_{O_i}^{obj} = Enc_{obj}([f_{O_i}; Pos_{O_i}; l_{O_i}]) \tag{4}$$

where the Enc_{obj} is a full-connected layer and [;] means the concatenation operator.

For other non-object node N_i, the recursive encoder Enc_{o2r} is used to gather the features of all children and perform message passing along the part relation edges among nodes. For the hyper-edges among multiple nodes, we first only aggregate the type of hyper-edges into the corresponding nodes to update the node features by an MLP Enc_{hyper}, which consists of two fully-connected (FC) layers and a Leaky ReLU activation. Then, we perform two message passing operations within the sub-graph and aggregate all features of child nodes by an FC layer with LeakyReLU activation. Finally, the feature f_{N_i} gathers information from the features of all children nodes.

$$\begin{aligned} \bar{f}_{O_i}^{obj} &= Enc_{hyper}(f_{O_i}^{obj}, l_{O_i}^{hyper}), s.t.\ O_i \in N_{hyper} \\ f_{N_i} &= Enc_{o2r}\left(\left\{\bar{f}_{O_j}^{obj}\right\}_{(N_i,O_j)\in\mathbf{H}}, l_{O_j}\right) \end{aligned} \tag{5}$$

where $(N_i, O_j) \in \mathbf{H}$ denotes that node O_j is a child of N_i, l_{O_i} is the one-hot vector of the semantic label, $l_{O_i}^{hyper}$ is the one-hot vector of the hyper edge type for object O_i, and N_{hyper} is the set of nodes with the attribute of hyper edges. If the node does not have any hyperedges associated with it, $l_{O_i}^{hyper}$ is empty (filled with zero).

The whole process Enc_{p2o} and Enc_{o2r} are repeated until to the root node. A fully connected layer maps the final feature of the root node into the latent space. We add a regularization term, namely the KL-divergence, to the latent space for encouraging the latent space to be close to the standard Gaussian distribution.

5.3 Scene Hierarchy Decoder

The decoding process is conditioned on the feature of floor boundary extracted by the floor encoder Enc_{floor}. The scene hierarchy decoder takes the floor condition and the root node feature outputted by the scene hierarchy encoder as inputs and learns to decode the scene hierarchy down to the part geometry in a recursive manner. It also includes two parts: one recursive decoder Dec_{o2p} that predicts part geometry from an object feature, and another recursive decoder Dec_{r2o} that consumes the parent node feature $\hat{f}_{N_i}$, along with the conditioned feature f_{floor}, and decodes the object root node features. For the object decoding, the Dec_{o2p} decoder follows the design in DSG-Net. We refer the readers to DSG-Net [8] for more details. Below, we focus on introducing the network design for Dec_{r2o}.

The recursive decoder Dec_{r2o} takes the room node feature as input and infers the node features of its children functional regions until reaching the object level. For each decoding step, we assume there are 10 children nodes at most for each parent and learn to predict the probability of node existence likelihood scores using a binary classification network (implemented by an MLP with a final Sigmoid activation function). We also predict the children's node semantic information as outputs. For the object nodes, we train an MLP to predict the placement parameters, categorically semantic labels, and object features, which will then be fed to the Dec_{o2p} for decoding the part hierarchy. The placement parameter prediction network consists of two fully-connected layers, a Leaky ReLU activation, and a skip-link. The network predicts the center $c_{O_i} \in \mathbb{R}^3$, scales $s_{O_i} \in \mathbb{R}^3$, and the orientation $r_{O_i} \in \mathbb{R}^1$ around the world up-axis by three individual fully-connected layers respectively. Hence, according to the extracted node features, we firstly predict node existence, semantics and geometry. For object nodes, the placement parameters are also predicted. Then we predict the edges between existing nodes according to their geometry and node features, and the edge type will be compared to the ground truth (GT) edge types for optimizing the hyper-parameters of the network during the recursive decoding for training.

For the binary edge predictions, we draw all pairs of existing nodes and predict the edge existence for every edge. For the hyper-edges, we predict a mask $\mathbf{M}_i$ by an attention mechanism to obtain which nodes share a hyper-edge attribute. Leveraging the predicted edge connections among nodes, we perform two iterations of message-passing for updating the node features. Finally, we obtain

the node features $\left\{f_{N_{j_1}}, f_{N_{j_2}}, \cdots, f_{N_{k_i}}\right\}$, where k_i denotes the number of existing nodes for parent node N_i. Above all, we have

$$\left\{\hat{f}_{N_{j_1}}, \hat{f}_{N_{j_2}}, \cdots, \hat{f}_{N_{j_{k_i}}}, \hat{\mathbf{R}}_i, \mathbf{M}_i\right\} = Dec_{r2o}(\hat{f}_i) \quad (6)$$

where $M_i \in \mathbb{R}^{N\times 3}$ is a matrix. Here, N is the number of nodes of the sub-graph for the parent node, and 3 is the number of hyper-edge types. For each row, it predicts the probabilities of different hyper-edge types, and the predicted type is the one with the highest probability (selected by 'argmax' operation). According to our defined hyper-edges, each object only has one type of hyper-edges, such as none, N-fold Rotation Symmetry, and parallel collinearity.

The recursive decode process of Dec_{r2o} is repeated until it reaches the object level, which is then the job of Dec_{o2p} to further decode it into object parts.

5.4 Training and Losses

We describe our training strategy and loss terms as follows.

5.4.1 Training Strategy

Since our scene hierarchy is a very deep tree from the room root node to functional regions, objects, and finally to object parts, it is very difficult to train it effectively together from scratch. We thus choose to train the network in two stages. We first train the recursive network from object to part geometry and then train the whole network while fine-tuning the pre-trained object-to-part network. For the floor boundary VAE, we train it separately from our backbone network. We conduct an ablation study in Sec. 6.4 for evaluating the benefit of such a training strategy.

5.4.2 Loss Terms

We define the total training loss $\mathcal{L}$ as the following:

$$\mathcal{L} = \mathbb{E}_{S\sim\mathfrak{S}}\left[\mathcal{L}_{recon} + \mathcal{L}_{struc} + \gamma\mathcal{L}_{KL}\right] \quad (7)$$

where $\mathfrak{S}$ is the distribution of scenes in the whole dataset, and the reconstruction loss $\mathcal{L}_{recon}$ includes leaf loss, semantic loss, edges/node existence loss, geometry loss, placement loss, and some edge losses (*e.g.* room-object edge, object-object edge, hyper-edge, and part-part edge). Except for placement loss and edge losses for the room-object edges and the proposed hyper-edges, the other loss terms are following the StructureNet [13]. We refer the readers to StructureNet [13] for these losses. For the structure consistency loss, it is used in StructureNet to ensure the generated structures of objects are reasonable and realistic. However, different from StructureNet, we only add the loss on the object hierarchy instead of the whole scene hierarchy. And the regularization $\mathcal{L}_{KL}$ aims to make the latent space smoother and easier for downstream applications (scene generation and interpolation). We set $\gamma = 0.01$ empirically for our experiments.

We now define placement parameter reconstruction loss and edge losses.

Placement Parameter Reconstruction Loss. The center $c_{O_i} \in \mathbb{R}^3$, scales $s_{O_i} \in \mathbb{R}^3$, and the orientation $r_{O_i} \in \mathbb{R}$ around the world up-axis are used to represent each object's location in the indoor scene. We apply the L2-Loss to the center and scale for encouraging the perfect reconstruction by $\mathcal{L}_{locate} = d_{center} + d_{scale} + d_{orient}$, where d is L2 distance metric, $d_{center} = \|c_{obj} - \hat{c}_{obj}\|_2^2$, $d_{scale} = \|s_{obj} - \hat{s}_{obj}\|_2^2$. $\hat{c}_{obj}$ and $\hat{s}_{obj}$ denote the center and scale of object location. Besides, we observe that furniture is typically located on the floor or ceiling, and the furniture shapes in the room are usually in 8 orientations with $45°$ intervals ($Angle = [0°, 45°, 90°, 135°, 180°, -45°, -90°, -135°]$) in most cases. So, we use a discrete representation to encode the orientation of furniture for the prediction of coarse orientation from the candidates above, along with a residual offset to fit the ground truth orientation of the object. In summary, we have $d_{orient} = \|Angle[\arg\max_k(\rho_k)] + b - o_{obj}\|_2^2$, where $\rho = (\rho_1, \rho_2, \cdots, \rho_8)$ is a predicted 8-d vector whose entity is the probability of the furniture at every orientation, b is the predicted offset in the range of $[-22.5°, 22.5°]$, and o_{obj} is the ground truth orientation of the furniture.

Room-object Binary Edge Loss. The main purpose of this loss term is to align the position of a predicted object with the boundary walls of the room as much as possible. In the original indoor scene data, the room boundary is mostly well aligned with the world x-axis and z-axis, if we denote the world up-axis as the y-axis. Under this assumption, we encourage the oriented bounding box of object to align with the x-axis and z-axis. The loss term is only applied to the objects with the attribute (room-objects edge) detected during decoding. We add a loss to approximate the distance between normals of room box and object box: $\mathcal{L}_{Ro} = \sum_{\forall O_i\in\mathbf{S}} d_{chs}(T(q_i)\mathbf{N}, \mathbf{N})$, where $\mathbf{S}$ is a set of all predicted objects, $d_{chs} = \frac{1}{|A_i|}\sum_{x_i\in A_i}\min_{x_j\in A_j}\|x_i - x_j\|_2^2 + \frac{1}{|A_j|}\sum_{x_j\in A_j}\min_{x_i\in A_i}\|x_j - x_i\|_2^2$ is Chamfer Distance [62, 63], and $\mathbf{N}$ is the six unit normal vectors of a unit box, $T(q_i)$ is a transformation matrix rotating the normals to align with orientation q_i of the object box.

Hyper-edge Loss. These loss terms encourage multiple objects to preserve their n-ary hyper-edge relationships. We include two types of losses here corresponding to the two types of hyper-edges.

For objects $\mathbf{O} = \{O_1, \cdots, O_N\}$ satisfying the n-fold rotational symmetry hyper-edge, we can define the loss:

$$\mathcal{L}_{\mathbf{e}_1^{hyper}} = \sum_{i=1}^{N}\left\{\min_{j=1,\cdots,N, i\neq j} d_{chs}\left[O_i, Rot\left(p, \frac{2\pi}{N}\right)\times O_j\right]\right\} \quad (8)$$

where $p = \frac{1}{N}\sum_{i=1}^{N} C_i$ is the barycenter of all object centers, $Rot(p, \theta)$ indicates a rotation matrix that rotates an object around an axis that is parallel to the y-axis and passes through point p by θ in radians. Because the n-fold symmetry in Equation 1 is ordered while our graph decoder is not, we must traverse through all decoded objects and find the one with the minimal Chamfer Distance.

For the collinearly parallel hyper-edges, we define two loss functions. The first one is

$$\mathcal{L}_{hpara_1} = \sum_{i=1}^{N}\left\{\sum_{j=i+1}^{N}\left[d_{chs}\left(T(q_i)\mathbf{N}, T(q_j)\mathbf{N}\right)\right]\right\} \quad (9)$$

where $\mathbf{N}$ is the six unit normals of the unit box, $T(q_i)$ is a transformation matrix that rotates the normals to align with orientation q_i of the object box. This loss corresponds to the first condition of a parallel hyper-edge. And the second loss is

$$\mathcal{L}_{hpara_2} = \sum_{i=1}^{N} (dist(C_i, \mathbf{v}, p)) \tag{10}$$

where C_i is the center of the OBB of O_i, $p = \frac{1}{N}\sum_{i=1}^{N} C_i$ is the barycenter of all centers, $\mathbf{v} = norm\left(\sum_{i=1}^{N}\sum_{j=i+1}^{N} C_j - C_i\right)$ is the average of relative position for all pairs of object centers in the hyper-edge, $norm$ indicates normalizing a certain vector. Note that when calculating $\mathbf{v}$, we sort the object centers according to their x-coordinates in an ascending order and then using their y-coordinates in an ascending order to avoid adding vectors pointing at opposite directions. Finally, we have the total loss function

$$\mathcal{L}_{\mathbf{e}_2^{hyper}} = \mathcal{L}_{hpara_1} + \mathcal{L}_{hpara_2} \tag{11}$$

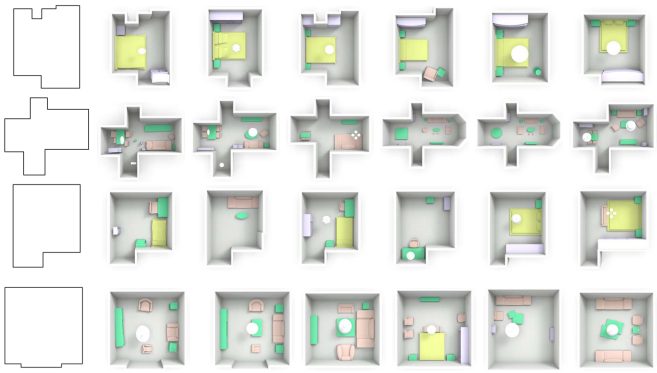
6 Experiments and Applications

We perform extensive experiments evaluating our SCENE-HGN for 3D scene reconstruction, generation, and interpolation, as well as many other applications, such as scene editing, conditional generation from 3D Box layout, and room completion. Since our framework is a generative model, the scene reconstruction evaluations are presented in our supplementary material. We compare GRAINS [4] and Deep Priors [40] as two state-of-the-art methods in terms of many quantitative metrics and a perceptual study, demonstrating our superior performance. Ablation studies further validate some of our key module designs. We use 3D-FRONT dataset [9] for our training and evaluation. 3D-FRONT is a newly released dataset of 3D indoor scenes which contains 6,815 houses and 51,708 rooms. The room designs are directly sourced from professional creations. In the dataset, each house is divided into several rooms with a room type associated. Among all the rooms, 18,797 rooms are furnished with objects from 3D-FUTURE [64], a dataset of textured 3D furniture models, and each model is labeled with a furniture category. We describe more details of dataset preparation in the supplementary material.

6.1 Scene Generation

The generation of 3D indoor scenes is the first and most straightforward application of our network. Our method can do free generation, but in reality, most of the generation tasks require some sort of condition as input (*e.g.* the boundary of a room). So we decide to take a floor boundary (which completely decides the walls) as a condition and input. The generated results are rooms filled with furniture. We show the quantitative and qualitative comparison with GRAINS [4], Deep Priors [40] and ATISS [43]. Also, the perceptual study is performed for the generative models.

Comparison with GRAINS. The settings of GRAINS are slightly different from ours. Firstly, GRAINS does not take any condition as input. Secondly, GRAINS uses four walls as 'anchors', which it encodes in its hierarchical representation. All the objects in the room need the walls to locate themselves. So GRAINS can only generate rooms with the same shape (in particular, rectangular rooms). But in our settings, the room shape can be arbitrary.



(a) Input (b) Ours (c) Top-5

Fig. 7: **The gallery of shape generation results given the room boundary and top-retrieved rooms in the training set.** To demonstrate the novelty of room generation, we show the top-5 nearest neighbors in the training set by shape retrieval according to the CD (Chamfer Distance) on the sampled point clouds (with 100,000 points). Given the room boundary, we can see that our generated shapes are different from the top-5 retrieved rooms on the object layout and geometric details, which demonstrates the novelty of generated rooms.

To reasonably compare the performance of GRAINS with ours, we first use GRAINS to generate 1,000 results for each room type. For an input room boundary and room type, we render the boundary into an image and find its inscribed rectangle with maximum area. With this rectangle, we can find the most similar room shape in the generated results and place the generated furniture into the corresponding rectangle in the input room boundary. This produces the 'conditional' generation result.

GRAINS can only predict the bounding boxes of the objects. With those boxes, it retrieves models from a database (*e.g.* 3D-FRONT, SUNCG) that match the bounding boxes the best. But our method can generate the geometry along with the room layout. In order to compare, we first create a model database using part VAE and graph encoder/decoder in our network. This portion of our network is actually a VAE itself and can perform free generation. We generate 200 objects for all categories for GRAINS to retrieve.

As for training, the original GRAINS VAE uses the SUNCG dataset [65]. Unfortunately, the SUNCG dataset is unavailable when we do the experiments. So we process the 3D-FRONT dataset and export it in the format of the SUNCG dataset, and train GRAINS on this dataset.

Comparison with Deep Priors. Deep Priors is a generation pipeline based on top-down view images of the room. Taking a room boundary as input, it iteratively inserts objects into the scene. For each object, it uses multiple networks to decide object location, orientation, and dimension.

Comparison with Deep Priors is straightforward because their settings are almost the same as ours. But there are still some differences. Just like GRAINS, Deep Priors retrieves

models from a database. We also use the generated model database for Deep Priors to retrieve models.

The original Deep Priors is trained on the SUNCG dataset which is unavailable, so again the exported 3D-FRONT dataset mentioned above is used instead.

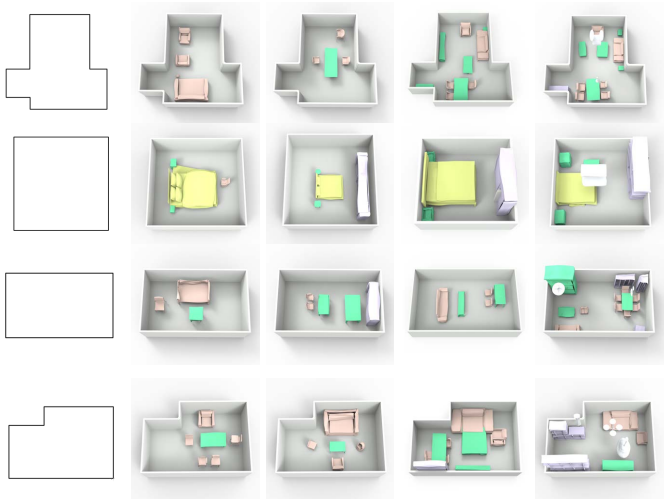
Comparison with ATISS. ATISS is a transformer-based generative model for a given room boundary. It also iteratively generates the object layouts and retrieves furniture to fill the empty scene.

ATISS is the state-of-the-art generative model in indoor scene synthesis, which is a very strong baseline in the 3D-Front and the setting is almost the same as ours. But there is also a major difference that is the same as Deep Priors, namely the furniture is retrieved from the original dataset in their method. So for comparison, we use the same generated model database for shape retrieval.

From the results in Figure 8, our approach can be able to capture the functional regions' variation (or a local gathering of furniture) better. For example, our method can successfully predict four chairs with the same geometry surrounding a table, while the other baselines cannot. Besides, more generated indoor scenes are present in Figure 9. Given a room boundary, our network can generate the object layouts and the fine-grained geometry of furniture. The figure shows 12 generated rooms, including 4 living rooms, 4 bedrooms, and 4 libraries, which indicates the generated plausible part geometries and reasonable object layout can fit the given room boundaries. Furthermore, to demonstrate the novelty of room generation, we show the top-5 nearest neighbors (Fig. 7) in the training set by shape retrieval according to the CD on the sampled points (100,000). The presented results reveal that our generated rooms are different from the top-5 retrieved rooms on the object layouts and geometry details.

Metrics for generation results. We have five metrics for generation results:

- FID stands for Fréchet Inception Distance [66] between the generation results and the ground truth. The results and the ground truth are rendered into a top-down view similar to the input of [39, 40].
- o_1 is obtained by first calculating the **distribution of furniture categories** across the generation results and the ground truth, and calculating the *Earth Mover's* Distance (EMD) [67] between them.
- o_2 is obtained by first calculating the **distribution of furniture categories for every room type** (*e.g.* bedroom, living room, etc.), and taking the average EMD.
- o_3 is obtained by first calculating the distribution of the **co-occurrence of every two types of furniture for every room type**, and taking the average EMD.
- o_4 measures the distribution of the correlation of object pairs from generated rooms, which is obtained following two steps: firstly we calculate the offset of the x-z positions of every possible object pair in each type of room, such as Table-Chair offset and Sofa-Table offset in living rooms. Secondly, we take out a square space around the origin of the 2-D plane, the edge length of which is 3.5m. This space is divided into a 1000×1000 grid. Every offset that lies in this square is counted, and a gray-scale image can be drawn from this grid and offsets.
- Orientation measures the radian distance between the rotation angle around the y-axis with the set of $\Theta = \{\theta|\theta = \frac{i\pi}{4}, i = -3, -2, -1, 0, 1, 2, 3, 4\}$. In the 3D-Front dataset, we observe that the orientation of almost all objects are aligned to the x, z-axis



(a) Input (b) GRAINS (c)DeepPriors (d) ATISS (e) Ours

Fig. 8: **Comparison on Room Generation.** We show the comparison of generation results of our method, Deep Priors [40], GRAINS [4], and ATISS [43]. From the results we can see that our method captures the functional regions (or local gathering of furniture) better. For example, our method can successfully predict four same chairs surrounding a dining table, while the three baseline methods cannot.

Fig. 9: **Room Generation results.** Given the room boundary, we can utilize our trained decoder to generate new rooms. Our network is able to take arbitrary room boundaries as input to generate object layouts and geometric details in a recursive manner. The figure shows 12 generated rooms (4 living rooms, 4 bedrooms, and 4 libraries). From the results, our network learns the continuous latent space successfully, which can capture the plausible part geometries and reasonable object layout that fits the room boundary simultaneously.

or diagonal direction of them, meaning that the rotation angle around y-axis θ is in the set of Θ. The function $\cos^2(x)$ is suitable since in our formulation $s = \cos^2(2\theta)$, it has peak value when x is among these eight angles. For the orientation of an object, we take its rotation angle θ around y-axis as input, and calculate s, then report the average of s for every object in our method and three baselines.

We believe that FID is a global metric that measures the similarity between the ground truth and the generation results. On the other hand, o_1, o_2, o_3, o_4 are 'structural' metrics that can prove whether the furniture arrangement patterns are learned. For example, in bedrooms, beds and nightstands often appear, and they often appear together. o_2 captures the appearance of beds and nightstands, while o_3 captures the co-occurrence of them. Here, o_1 encourages the distribution of the overall furniture category to be close to the training set. o_4 measures the distribution of the correlation of object pairs from the generated rooms. For this metric, we use the histogram to visualize the correlation of object pairs in Figure 10, *e.g.* Bed-Cabinet, Bed-Nightstands, Sofa-Table, Table-Chair. From the results in Figure 10, we can see that our generated results can successfully capture the distribution of objects related to other objects. All the results are reported in Table 1. From the numerical evaluations, we can see that our method outperform all baselines on FID, o_1, o_2, o_3 and has a similar performance on orientation metric with state-of-the-art work ATISS.

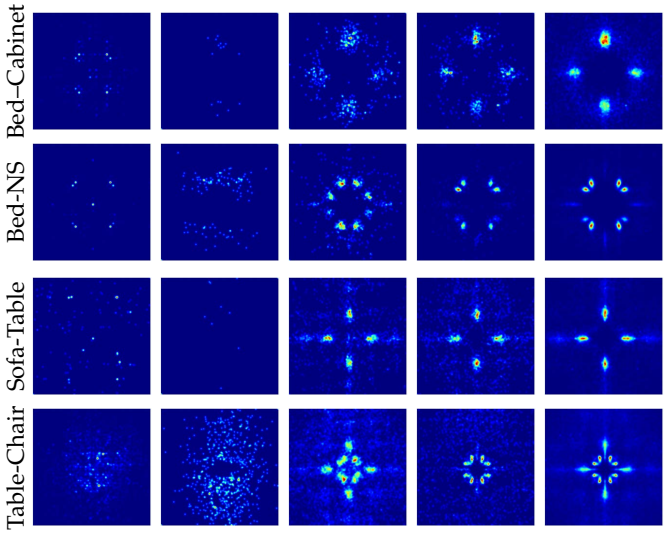


Fig. 10: **Comparison on Room Generation for correlation metric o_4.** We show the comparison of generation results on correlation metric o_4 of our method, Deep Priors (DP) [40], GRAINS [4], ATISS [43], and GT (training data). We display some selected histograms on Bed-Cabinet, Bed-NS (Nightstands), Table-Chair and Sofa-Table. From the results we can see that our generated results can captures distribution of objects related to other objects.

Perceptual study on generation results. In addition to quantitative results, we conduct a perceptual study on the generation results of our method and two baseline methods(GRAINS and Deep Priors). To fairly compare the results, we randomly select 100-floor boundaries from the dataset and generate rooms for all three methods using these boundaries as conditions. For each participant, we prepare 20 questions. For every question, we ask the user to rank the results under three criteria: a) The layout of the furniture, a.k.a locations, and categories of the placed furniture. b)The coordination of the furniture (*e.g.* a dining table and a beach chair do not coordinate with each other). c) The overall performance of the generated rooms. All the participants were local volunteers known to be reliable. The results of the perceptual study are shown in Table 2.

TABLE 1: **Generation comparison metrics between methods.** We compute Fréchet Inception Distance and four additional metrics o_1, o_2, o_3 and orientation that can measure the distribution of furniture in the generated rooms of our method, Deep Priors [40], GRAINS [4], and ATISS [43]. We can see that our results outperforms the all baselines on FID, o_1, o_2, o_3 and has a similar performance on orientation metric with SoTA work ATISS.

Methods	FID	o_1	o_2	o_3	orientation (degree)
Ours	**139.4508**	**0.05**	**0.13**	**0.47**	0.9572 (5.98°)
GRAINS [4]	181.9106	0.30	0.52	0.79	0.9931 (2.39°)
Deep Priors [40]	158.0010	0.40	0.41	1.04	0.9505 (6.43°)
ATISS [43]	141.7889	0.47	0.19	0.87	0.9578 (5.92°)

We can see that our method is the most preferred among the three methods. In the two baselines, Deep Priors performs better in layout, and GRAINS performs better in furniture coordination. We infer the reason for this result is Deep Priors tend to capture the whole furniture layout via top-down images and CNN, while GRAINS models structural relationships better with their hierarchical representation and RvNN-VAE.

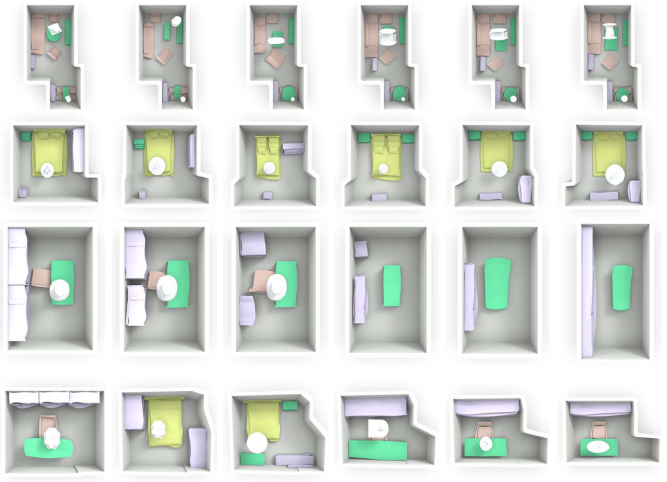
6.2 Scene Interpolation

Interpolation is another direct application of our trained RvNN-VAE. To show the smoothness of the latent space our VAE has learned, we show some examples of interpolation in Figure 11. When interpolating, we first encode the floor boundary and the scene layout of source and target scenes into feature vectors and perform linear interpolation between source and target on the two features simultaneously before feeding them into our conditional VAE decoder. Thanks to our generative FloorNet to ensure the meaningful latent space, FloorNet can achieve a reasonable interpolated room boundary between source and target. Note that the interpolation between 3D indoor scenes is not as straightforward as 3D shapes, because our representation not only encodes the layout of objects, but also contains geometry information. We find that interpolation between two similar scenes has the best performance. For example, The source and target of the second row in Figure 11 have similar floor boundaries and furniture layout.

As is shown in Figure 11, every step in the interpolation process is a valid 3D scene layout. The floor boundary gradually deforms from the source to the target, but in every step the layout changes according to the boundary rather than moving itself, which is what we expect. We also witness

TABLE 2: **Perceptual study results on 3D scene generation.** We show the average ranking scores (from 2 (the best) to 0 (the worst)) and the frequency of the method being ranked 1^{st}, 2^{nd} and 3^{rd}: Deep Priors [40], GRAINS [4], and ours. The results are calculated based on 433 trials. We see that our method achieves the best on all metrics.

Method	Deep Priors[40]/Rank($1^{\text{st}}, 2^{\text{nd}}, 3^{\text{rd}}$)	GRAINS[4]/Rank($1^{\text{st}}, 2^{\text{nd}}, 3^{\text{rd}}$)	Ours/Rank($1^{\text{st}}, 2^{\text{nd}}, 3^{\text{rd}}$)
Layout	0.7611/(17.5%, 41.2%, 41.4%)	0.6482/(11.1%, 42.7%, 46.2%)	**1.5907**/(71.4%, 16.2%, 12.4%)
Furniture Coordination	0.7212/(15.2%, 41.6%, 43.1%)	0.7389/(19.7%, 34.5%, 45.8%)	**1.5398**/(65.0%, 23.9%, 11.0%)
Overall	0.6925/(14.8%, 39.6%, 45.6%)	0.6748/(12.2%, 43.1%, 44.7%)	**1.6327**/(73.0%, 17.2%, 9.8%)



(a) source (b) target

Fig. 11: **Room Interpolation.** We simultaneously interpolate on boundary and scene layout and feed them into the VAE decoder. Every interpolation step is a valid 3D scene layout, with the boundary deforming continuously and the layout changing according to the boundary. Note that the interpolation is only reasonable when the source and target are similar.

the deformation of object geometry in interpolation. For example, look at row 2 in Figure 11: the small square-shaped shelf in the bottom of the source scene image stretches and becomes a longer one in the target scene.

The interpolation results show that our latent space is smooth near any point representing a valid 3D indoor scene layout. This is a key feature for more applications like room editing or room completion.

6.3 Applications

The hierarchical graph representation of 3D indoor scenes and locally smooth latent space created by the RvNN-VAE enable us to perform multiple interesting applications. In this section, we demonstrate three: a) room editing at multiple levels; b) room generation conditioned on 3D box layouts; and c) scene completion from a partial input room.

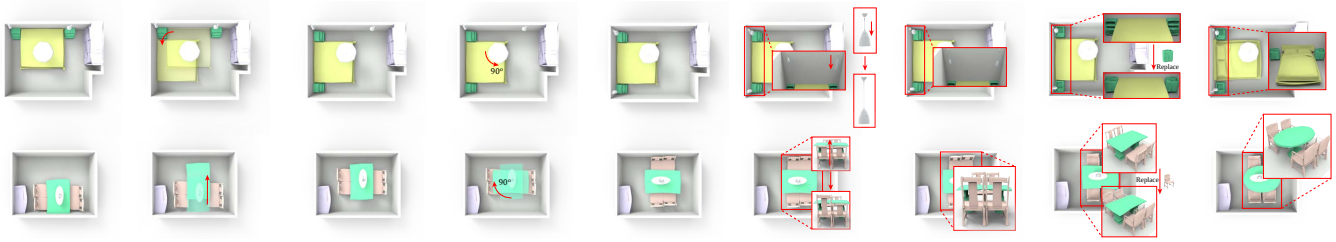
Room Editing. Editing 3D indoor scenes is not an easy task, because all the elements from functional regions to object parts are in relation to each other. Thus only editing one of them often makes the scene seem inharmonious (*e.g.* Only editing one of four chairs surrounding a table makes the scene looks weird). But our RvNN-VAE learns the whole scene including the object part geometry and scene layout. With an edit to the scene and the latent space, we can find a latent code in the space that both satisfies the edit and decodes to a valid scene layout.

Because of the local smoothness of our latent space, similar scenes are close to each other in the space. After editing a scene, we can search for a latent code in the space which is close to the original scene and satisfies the editing. So we apply the gradient descent (using the Adam [68] optimizer) to minimize the objective function: $\|z - z_*\|_2^2 + q_{chs}(T(B_e^z)\mathbf{U}, T(B_e^t)\mathbf{U}) + \mathcal{L}_{struc}(d(z))$, where z is the latent vector we need to optimize on, z_* is latent vector of the unedited original scene, B_e^t is the edited box, and B_e^z is the corresponding box in the decoded scene of z. (Note that both the edit of the object and the edit of a part are actually edited on boxes but at different levels.) $\mathbf{U}$ is a pre-computed set of samples on the unit cube. This loss function encourages our edited scene to be as close as the original scene while preserving the edited features. Figure 12 shows some results of our room editing application.

In Figure 12, each row shows a scene that has been edited four times. The first column shows the original scene, the second column shows the first edit on a certain object, and the third column shows the result of the editing. There are three groups of edits and results. The first two edits are rigid, meaning we only edit the locations or orientations of the objects. In the third edit, we deform object parts to demonstrate our network can process fine-grained features down to object parts. For the final edit (replacing the geometry), it can lead to the other object changes happening in the scene, which demonstrates that our method is able to learn the correlations of objects from the data.

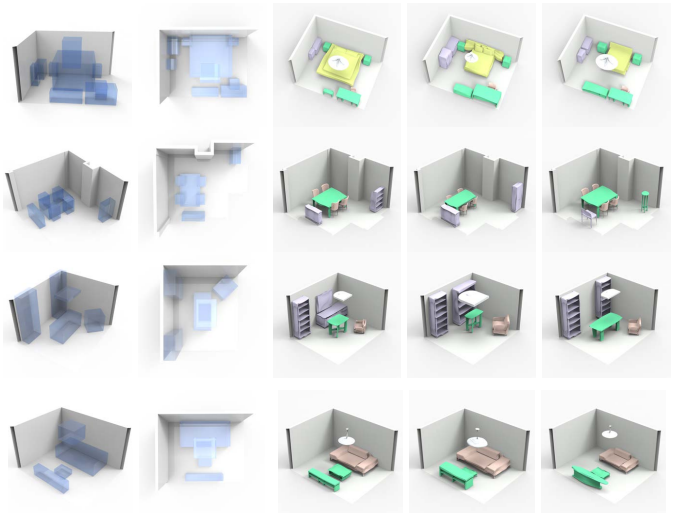
For example, The second row shows three edits on a living room. Firstly, we move the table to the left. The result is that the chairs surrounding it move with it, and all other objects remain the same. Secondly, we rotate the table counterclockwise by 90 degrees, and the chairs also rotate around the center of the table. Thirdly, we stretch the back of one chair to make it taller, and all the other chairs become taller too.

Room Generation from 3D Box Layout. Sometimes more conditions than floor boundaries are given when generating 3D indoor scenes. A widely used condition in interior design is the 2D floor plan, which basically contains the information on floor boundaries and the 2D bounding boxes of the furniture. We can extend 2D floor plans to 3D box layouts. In 3D box layouts, we provide room boundary and 3D bounding boxes of objects, but the semantic types and part geometry of objects are unknown. We are required



(a) Origin (b) Op1. (c) Edited (d) Op2. (e) Edited (f) Op3. (g) Edited (h) Op4. (i) Edited

Fig. 12: **Room Editing.** The first column shows the original scene, followed by pairs of columns demonstrating the edits and their results. There are four edits to each of the two scenes. The first and second edits only alter the locations and orientations of the objects, the third edit deforms object parts, and fourth edit replaces the geometry of objects. From the results, we can observe that every object related to the edited object moves or deforms according to the edit.



(a)3D box-layout(2 Views) (b) Generated Room

Fig. 13: **3D scene generation from 3D box layout.** We input a hierarchy consisting of 3D boxes into our RvNN-VAE with geometry information sampled from random distribution, then we encode the hierarchy, sample a latent vector and decode it into a complete 3D scene. We can see the positions of objects in the generated results are similar to the box layout, and the detailed geometry of the scenes looks harmonious.

to generate a room, where the furniture in it needs to coordinate with the input boxes.

To complete the task, we can build a hierarchy with no structural edges and part-level nodes. In the object-level nodes, we only include the box features, and fill the geometry features with random numbers. We then construct a hierarchy with these boxes (note there are no structural relations in this hierarchy) and feed this hierarchy into the encoder of our VAE, then we sample from the Gaussian distribution our encoder outputs. The sampled latent vector is then mapped back into a hierarchy-graph representation of our generated 3D scene. Some results of this procedure are displayed in Figure 13.

In Figure 13 we include two views of input box layouts for clarity. For each input box layout, we generate three results. We can see clearly that all the objects in the results are placed roughly in the position of the input boxes, but the geometry of the objects is different. The part geometry is completely generated by our network, and the generated geometry looks harmonious (*e.g.* in row 2 the size of the chairs corresponds to the size of the table).

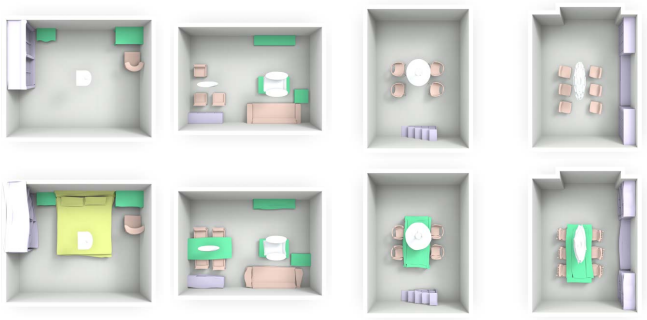


Fig. 14: **Room Completion.** Given a partial room, our method can complete the room layout. Each of the top four images shows one partial scene, and the bottom four images show the results of our room completion. We can see not only the deleted object is added, its children are added too.

Room Completion. Room completion is a somehow more challenging task than room generation conditioned on 3D box layout. With a partial scene, our method needs to predict all of the missing objects in the room. To solve this problem, we make use of the latent space learned by our VAE. As the reconstruction loss is used to train the VAE, in theory every point in the latent space can be decoded into a complete scene. So the room completion pipeline consists of two parts: using the encoder to map a partial scene to a point in the latent space, and using the decoder on the point to get the complete scene. To test the quality of room completion, we can start by deleting some of the nodes in our hierarchical graph representation and its children, and feed this partial tree into our VAE, then decode the sampled latent vector into a complete scene. The results are shown in Figure 14.

The top four images and the bottom four images of Figure 14 show the partial and completed scenes. The first, third, and fourth columns show that our method can complete scenes missing one key object. The second column

TABLE 3: **Quantitative Scene Reconstruction Performance of the Ablation Studies.** We show the CD($\times 10^{-5}$) and EMD($\times 10^{-4}$) metrics of the reconstruction results and FID, o_1, o_2, o_3 metrics of the generation results from our full method and the ablated versions, each of which is trained without a certain element of our key scene hierarchy designs. We can see from the table that the performance of our full method is the best.

Methods	Reconstruction		Generation			
	CD	EMD	FID	o_1	o_2	o_3
separate training	315.2	423.5	139.450	0.065	0.126	1.348
end-to-end training	473.9	676.8	170.079	0.119	0.140	1.344
w/o object-object edge	321.3	440.8	145.398	0.119	0.137	1.625
w/o room-object edge	344.9	470.9	150.478	0.069	0.131	1.357
w/o hyper-edge	315.8	443.5	130.276	0.080	0.126	1.553
w/o functional region	398.1	563.3	135.940	0.096	0.140	1.595
Ours (Full)	310.7	389.0	106.005	0.050	0.130	0.470

shows that the children of the missing object can also be added back, while the other parts of the room remain unchanged.

6.4 Ablation Study

We introduce functional regions and many node edges, including the proposed hyper-edges, in our method. To show that these elements are indeed beneficial, we perform a set of ablation studies. For each ablation study, we remove a certain element from our method, use this version to reconstruct 3D indoor scenes, and compare its performance to our full method. We also validate our training strategy. Again, we use Chamfer Distance and Earth Mover's Distance to measure the quality of reconstruction, and use FID, o_1, o_2, o_3 to measure the ability of generation. We show the quantitative evaluations in Table 3, where we can see clearly that all of the ablated versions perform worse than our full method, especially for the distribution o_3 of co-occurrence of two furniture for each room, there is a very large margin compared to other ablated versions.

Structures in Hierarchical Graph Representation. We show the reconstruction results of our method without a certain structure in the hierarchical graph representation in Figure 15. We consider some key structures in the hierarchy: functional regions, room-object edges, object-object edges, and hyper-edges. We can see that removing each of them introduces some specific flaws in the reconstruction results.

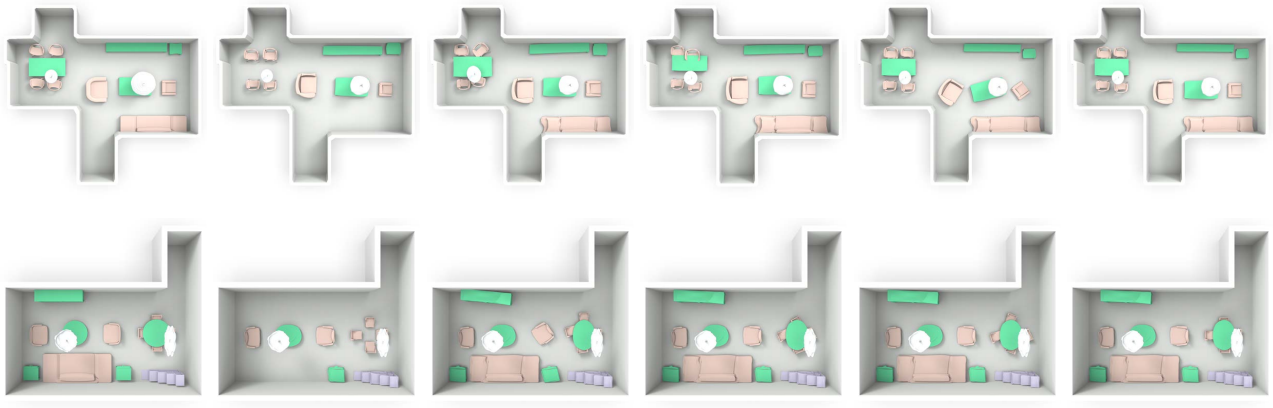
As we can observe from Figure 15, without functional regions, the network fails to predict some objects because the hierarchy is missing a whole level of nodes, and is not organized tightly. Without room-object edges to keep the objects aligned, we see that the chairs and the tables are often misaligned. Without object-object edges to preserve the symmetrical relation between objects, the chairs around the table (which are in rotational or reflective symmetry) fail to preserve the symmetrical relation. Without hyper-edges, we also find that the rotational symmetry fails to preserve, and in addition we observe that the parallel relationship breaks in the top image of the fifth column of Figure 15. Our full method helps address the above issues.

Training Strategy. When training our network, we use two-stage training. More precisely, we first train the networks that encode and decode the part-to-object Enc_{p2o}, Dec_{o2p}. Then, we start training the two other networks Enc_{o2r}, Dec_{r2o} while fine-tuning the Enc_{p2o}, Dec_{o2p} networks. We find this training strategy gives better results than training the four networks from scratch jointly or separately training the room-object and object-part networks. This is also intuitively reasonable as the object-part networks and the room-object networks are relatively entangled to learn the consistency between the objects within a functional region and performing a joint training from scratch makes the network too deep to be effectively trained. Figure 16 shows some qualitative result comparisons, the end-to-end training means that we train the whole networks without fine-tuning object-to-part networks, the separate training means that we first train the object-part network and then train the room-object network. The losses of both networks in Figure 16 have converged, and we can conclude from the images that the layouts and the geometry of the two-stage training reconstruction results are considerably better and more realistic than the others.

7 Conclusion, Limitations and Future Work

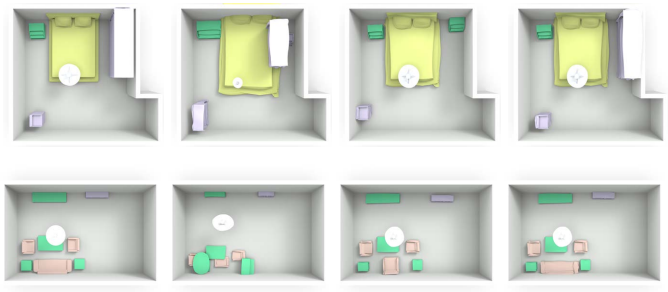
We propose a hierarchical graph network SceneHGN on 3D indoor scene generation. Our method conducts learning over a structural scene hierarchy containing multiple conceptual levels of entities: the room, functional regions, objects, and object parts. We train a recursive variational autoencoder that learns to map 3D scene hierarchies to a latent manifold, on which we can generate diverse 3D scene meshes by randomly sampling over the learned distribution, interpolate smoothly between input 3D scene data, and perform many downstream applications, such as scene editing, conditional generation, and completion tasks. At the core of our innovations, we propose a level of functional regions between the room root node and its constituent objects for more effective and efficient learning and devise a rich set of binary edges and n-ary hyper-edges among the nodes in the hierarchy for better modeling the structural and relational constraints for a valid 3D scene generation. We conducted extensive evaluations and comparisons to strong state-of-the-art methods, demonstrating our superior performance.

Limitations and Future Works. There are some limitations of our SceneHGN networks: a) our networks are basically VAEs, and naturally need a large amount of data of high consistency and quality for training; b) the construction of our hierarchical graph representation heavily relies on annotated data such as object part hierarchy, especially for some small objects (*e.g.* cups, vases, books), more labeling is required to allow our approach to be easily extended to more object types. Also, our framework is not designed for the vertical edges within object-level, *e.g.* supporting, etc., which is limited for representing complex structures on the z-axis. But our model is easily generalizable to this design via adding an extra edge, which can be our future feature as deep exploration along the direction; c) our introduced edges and functional regions can successfully model most of the layouts in the 3D-FRONT dataset, but we cannot guarantee the structures are applicable to all scene data; d) we use non-rigid registration and ACAP feature to encode the detailed geometry for the room boundary layout and the



(a) Input (b) w/o FR (c) w/o ROE (d) w/o OOE (e) w/o HE (f) Ours

Fig. 15: **Ablation Study on Key Structural Designs of the Scene Hierarchy.** We perform ablation studies that respectively remove the introduced functional region level and each type of relation edges. It is clear that: in (b), the version without functional regions (FR) drops some objects; in (c), the version without room-object edges (ROE) predicts misaligned objects; in (d), the version without object-object edges (OOE) struggles when objects need to be placed symmetrically; and (e), the version without hyper-edges (HE) fails to place parallel objects. In contrast, our full method does not have any of these problems.



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

Fig. 16: **Training Strategy.** We compare 3 training methods: end-to-end, separate and two-stage training on reconstruction results. (a) is the input scene, (b) is reconstruction via end-to-end training of total networks, (c) is the reconstruction via separate training (i.e., first train object-part network, and then train room-layout network), (d) is reconstruction via two-stage training with fine-tuning the object-to-part networks. All networks have converged, and the results of two-stage training are much better than the others. Compared to separate training (c), our full model can achieve reasonable and realistic results due to fine-tuning the object-part networks, which optimizes the relation and geometry between objects.

object part geometry, which do not always provide realistic model generation in our results; e) doors and windows are often an important part of a complete room. We use the deformation to represent the room boundary. Since doors and windows are located on the wall, they are not compatible with our framework. Our proposed framework needs to be extended to handle such relationships for windows and doors, which should be achievable from a technical point of view. f) our network is not able to generate photorealistic rooms since the texture is not considered. So it would be interesting as future work to align the PartNet hierarchy with 3D-FRONT models, using the former as structure supervision and the latter for detailed geometry and texture.

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

Fig. 17: **Failure Cases.** We present exemplar failure cases of our scene generation results. We may see some problematic prediction results, such as the invalid intersection between objects (a), some duplicate or missing objects in the scene ((b), (c)), prediction errors for the semantic categories of objects (*e.g.* a chair is predicted to be a lamp and attached to the ceiling in (d)), or unaligned object orientations with the room boundary walls (e).

We show the several failure cases of our method in Figure 17. Since we do not explicitly discourage collision between objects, the current approach may generate objects with intersection among them in some cases, *e.g.* Figure 17 (a). Some points in our latent space may be mapped to generate imperfect scenes with missing/duplicate objects or incorrect semantic labels. See Figure 17 (b, c, d) and the figure caption for more detailed explanations. Lastly, although we introduce room-object edges to align objects with rooms, there would still be some occasional failure cases, as presented in Figure 17 (e). Future works may address these issues and thus further improve the 3D scene generation performance of our framework.

ACKNOWLEDGMENT

This work was supported by the Beijing Municipal Natural Science Foundation for Distinguished Young Scholars (No. JQ21013), the National Natural Science Foundation of China

(No. 62061136007) and the Youth Innovation Promotion Association CAS. Kaichun Mo and Leonidas J. Guibas were supported by the ARL grant W911NF-21-2-0104, a Vannevar Bush Faculty Fellowship, and a gift from the Adobe Corporation.

REFERENCES

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

[2] P. Henderson and V. Ferrari, "A generative model of 3d object layouts in apartments," 2017.

[3] K. Wang, Y.-A. Lin, B. Weissmann, M. Savva, A. X. Chang, and D. Ritchie, "Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–15, 2019.

[4] M. Li, A. G. Patil, K. Xu, S. Chaudhuri, O. Khan, A. Shamir, C. Tu, B. Chen, D. Cohen-Or, and H. Zhang, "Grains: Generative recursive autoencoders for indoor scenes," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 2, pp. 1–16, 2019.

[5] J. Li, K. Xu, S. Chaudhuri, E. Yumer, H. Zhang, and L. Guibas, "Grass: Generative recursive autoencoders for shape structures," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, pp. 1–14, 2017.

[6] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. J. Mitra, and L. J. Guibas, "Structurenet: hierarchical graph networks for 3d shape generation," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–19, 2019.

[7] L. Gao, J. Yang, T. Wu, Y.-J. Yuan, H. Fu, Y.-K. Lai, and H. Zhang, "Sdm-net: Deep generative network for structured deformable mesh," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–15, 2019.

[8] J. Yang, K. Mo, Y.-K. Lai, L. J. Guibas, and L. Gao, "Dsm-net: Disentangled structured mesh net for controllable generation of fine geometry," *arXiv preprint arXiv:2008.05440*, 2020.

[9] H. Fu, B. Cai, L. Gao, L.-X. Zhang, J. Wang, C. Li, Q. Zeng, C. Sun, R. Jia, B. Zhao *et al.*, "3d-front: 3d furnished rooms with layouts and semantics," in *International Conference on Computer Vision (ICCV)*, 2021, pp. 10 933–10 942.

[10] J. Zheng, J. Zhang, J. Li, R. Tang, S. Gao, and Z. Zhou, "Structured3d: A large photo-realistic dataset for structured 3d modeling," in *European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 519–535.

[11] Z. Li, T.-W. Yu, S. Sang, S. Wang, M. Song, Y. Liu, Y.-Y. Yeh, R. Zhu, N. Gundavarapu, J. Shi *et al.*, "Openrooms: An open framework for photorealistic indoor scene datasets," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 7190–7199.

[12] M. Roberts, J. Ramapuram, A. Ranjan, A. Kumar, M. A. Bautista, N. Paczan, R. Webb, and J. M. Susskind, "Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding," in *International Conference on Computer Vision (ICCV)*, 2021.

[13] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 909–918.

[14] J. Johnson, A. Gupta, and L. Fei-Fei, "Image generation from scene graphs," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 1219–1228.

[15] O. Ashual and L. Wolf, "Specifying object attributes and relations in interactive scene generation," in *International Conference on Computer Vision (ICCV)*, 2019, pp. 4561–4569.

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

[17] A. Luo, Z. Zhang, J. Wu, and J. B. Tenenbaum, "End-to-end optimization of scene layout," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 3754–3763.

[18] R. Hu, Z. Huang, Y. Tang, O. V. Kaick, H. Zhang, and H. Huang, "Graph2plan: Learning floorplan generation from layout graphs," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 118:1–118:14, 2020.

[19] N. Nauata, K. Chang, C.-Y. Cheng, G. Mori, and Y. Furukawa, "House-gan: Relational generative adversarial networks for graph-constrained house layout generation," in *European Conference on Computer Vision (ECCV)*, 2020.

[20] W. Para, P. Guerrero, T. Kelly, L. Guibas, and P. Wonka, "Generative layout modeling using constraint graphs," *arXiv preprint arXiv:2011.13417*, 2020.

[21] S. Huang, S. Qi, Y. Zhu, Y. Xiao, Y. Xu, and S.-C. Zhu, "Holistic 3d scene parsing and reconstruction from a single rgb image," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 187–203.

[22] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3d scene graph: A structure for unified semantics, 3d space, and camera," in *International Conference on Computer Vision (ICCV)*, 2019, pp. 5664–5673.

[23] R. Socher, C. C.-Y. Lin, A. Y. Ng, and C. D. Manning, "Parsing natural scenes and natural language with recursive neural networks," in *ICML*, 2011.

[24] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Ng, "Convolutional-recursive deep learning for 3d object classification," *Advances in neural information processing systems*, vol. 25, pp. 656–664, 2012.

[25] Y. Shi, A. X. Chang, Z. Wu, M. Savva, and K. Xu, "Hierarchy denoising recursive autoencoders for 3d scene layout prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1771–1780.

[26] S.-H. Zhang, S.-K. Zhang, Y. Liang, and P. Hall, "A survey of 3d indoor scene synthesis," *Journal of Computer Science and Technology*, vol. 34, no. 3, pp. 594–608, 2019.

[27] G. Pintore, C. Mura, F. Ganovelli, L. Fuentes-Perez, R. Pajarola, and E. Gobbetti, "State-of-the-art in automatic 3d reconstruction of structured indoor environments," in *Computer Graphics Forum*, vol. 39, no. 2. Wiley Online Library, 2020, pp. 667–699.

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

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

[30] Y.-T. Yeh, L. Yang, M. Watson, N. D. Goodman, and P. Hanrahan, "Synthesizing open worlds with constraints using locally annealed reversible jump mcmc," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–11, 2012.

[31] W. Xu, B. Wang, and D.-M. Yan, "Wall grid structure for interior scene synthesis," *Computers & Graphics*, vol. 46, pp. 231–243, 2015.

[32] Z. S. Kermani, Z. Liao, P. Tan, and H. Zhang, "Learning 3d scene synthesis from annotated rgb-d images," in *Computer Graphics Forum*, vol. 35, no. 5. Wiley Online Library, 2016, pp. 197–206.

[33] P. Henderson, K. Subr, and V. Ferrari, "Automatic generation of constrained furniture layouts," *arXiv preprint arXiv:1711.10939*, 2017.

[34] Y. Liang, S.-H. Zhang, and R. R. Martin, "Automatic data-driven room design generation," in *International Workshop on Next Generation Computer Animation Techniques*. Springer, 2017, pp. 133–148.

[35] S.-H. Zhang, S.-K. Zhang, W.-Y. Xie, C.-Y. Luo, Y. Yang, and H. Fu, "Fast 3d indoor scene synthesis by learning spatial relation priors of objects," *IEEE Transactions on Visualization and Computer Graphics*, 2021.

[36] K. Xu, L. Zheng, Z. Yan, G. Yan, E. Zhang, M. Niessner, O. Deussen, D. Cohen-Or, and H. Huang, "Autonomous reconstruction of unknown indoor scenes guided by time-varying tensor fields," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–15, 2017.

[37] S. Dong, K. Xu, Q. Zhou, A. Tagliasacchi, S. Xin, M. Nießner, and B. Chen, "Multi-robot collaborative dense scene reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–16, 2019.

[38] K. Xu, H. Huang, Y. Shi, H. Li, P. Long, J. Caichen, W. Sun, and B. Chen, "Autoscanning for coupled scene reconstruction and proactive object analysis," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–14, 2015.

[39] D. Ritchie, K. Wang, and Y.-a. Lin, "Fast and flexible indoor scene synthesis via deep convolutional generative models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6182–6190.

[40] K. Wang, M. Savva, A. X. Chang, and D. Ritchie, "Deep convolutional priors for indoor scene synthesis," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[41] Z. Zhang, Z. Yang, C. Ma, L. Luo, A. Huth, E. Vouga, and Q. Huang, "Deep generative modeling for scene synthesis via hybrid representations," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 2, pp. 1–21, 2020.

[42] X. Wang, C. Yeshwanth, and M. Nießner, "Sceneformer: Indoor scene generation with transformers," in *International Conference on 3D Vision (3DV)*. IEEE, 2021, pp. 106–115.

[43] D. Paschalidou, A. Kar, M. Shugrina, K. Kreis, A. Geiger, and S. Fidler, "Atiss: Autoregressive transformers for indoor scene synthesis," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[44] Y. Zhou, Z. While, and E. Kalogerakis, "Scenegraphnet: Neural message passing for 3d indoor scene augmentation," in *International Conference on Computer Vision (ICCV)*, 2019, pp. 7384–7392.

[45] L. Liu, Y. Yang, Y. Yuan, T. Shao, H. Wang, and K. Zhou, "In-game residential home planning via visual context-aware global relation learning," in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 336–343.

[46] H. Yang, Z. Zhang, S. Yan, H. Huang, C. Ma, Y. Zheng, C. Bajaj, and Q. Huang, "Scene synthesis via uncertainty-driven attribute synchronization," in *International Conference on Computer Vision (ICCV)*, 2021, pp. 5630–5640.

[47] R. Ma, A. G. Patil, M. Fisher, M. Li, S. Pirk, B.-S. Hua, S.-K. Yeung, X. Tong, L. Guibas, and H. Zhang, "Language-driven synthesis of 3d scenes from scene databases," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–16, 2018.

[48] Q. Fu, X. Chen, X. Wang, S. Wen, B. Zhou, and H. Fu, "Adaptive synthesis of indoor scenes via activity-associated object relation graphs," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–13, 2017.

[49] S. Qi, Y. Zhu, S. Huang, C. Jiang, and S.-C. Zhu, "Human-centric indoor scene synthesis using stochastic grammar," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5899–5908.

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

[51] C. Goller and A. Kuchler, "Learning task-dependent distributed representations by backpropagation through structure," in *International Conference on Neural Networks (ICNN'96)*, vol. 1. IEEE, 1996, pp. 347–352.

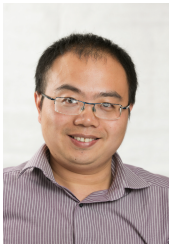
[52] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint*

arXiv:1503.00075, 2015.
[53] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[54] C. Niu, J. Li, and K. Xu, "Im2struct: Recovering 3d shape structure from a single rgb image," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4521–4529.
[55] F. Yu, K. Liu, Y. Zhang, C. Zhu, and K. Xu, "Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9491–9500.
[56] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. J. Mitra, and L. J. Guibas, "Structedit: Learning structural shape variations," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 8859–8868.
[57] K. Mo, H. Wang, X. Yan, and L. Guibas, "PT2PC: Learning to generate 3d point cloud shapes from part tree conditions," *European Conference on Computer Vision (ECCV)*, 2020.
[58] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.
[59] Y. Wang, K. Xu, J. Li, H. Zhang, A. Shamir, L. Liu, Z. Cheng, and Y. Xiong, "Symmetry hierarchy of man-made objects," in *Computer graphics forum*, vol. 30, no. 2. Wiley Online Library, 2011, pp. 287–296.
[60] L. Gao, Y. Lai, J. Yang, L.-X. Zhang, L. Kobbelt, and S. hong Xia, "Sparse data driven mesh deformation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, pp. 2085–2100, 2021.
[61] S. Bouaziz, A. Tagliasacchi, and M. Pauly, "Dynamic 2d/3d registration." in *Eurographics (Tutorials)*. Citeseer, 2014, p. 7.
[62] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 605–613.
[63] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *Proceedings: Image Understanding Workshop*, 1977, pp. 21–27.
[64] H. Fu, R. Jia, L. Gao, M. Gong, B. Zhao, S. Maybank, and D. Tao, "3d-future: 3d furniture shape with texture," *International Journal of Computer Vision*, pp. 1–25, 2021.
[65] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 190–198, 2017.
[66] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017.
[67] Y. Rubner, C. Tomasi, and L. Guibas, "A metric for distributions with applications to image databases," *International Conference on Computer Vision (ICCV)*, pp. 59–66, 1998.
[68] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

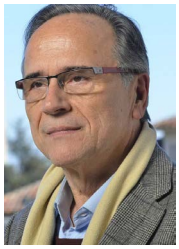
Lin Gao received his PhD degree in computer science from Tsinghua University. He is currently an Associate Professor at the Institute of Computing Technology, Chinese Academy of Sciences. He has been awarded the Newton Advanced Fellowship from the Royal Society and the AG young researcher award. His research interests include computer graphics and geometric processing.

Jia-Mu Sun is the Master student in Computer Science at Institute of Computing Technology, Chinese Academy of Sciences. He received his bachelor's degree from Huazhong University of Science and Technology. His research interests include computer graphics.

Kaichun Mo is a sixth-year and final-year Ph.D. Student in Computer Science at Stanford University, advised by Prof. Leonidas Guibas. Before that, he received his BS.E. degree from the ACM Honored Class at Shanghai Jiao Tong University. His research interests focus on understanding 3D shape structure and semantics for various applications in 3D vision, graphics, and robotic manipulation. He has interned at Adobe Research, Autodesk Research (AI Lab), and Facebook AI Research. He has published papers at CVPR, ICCV, ECCV, NeurIPS, ICLR, Siggraph Asia, AAAI, and CoRL.



Yu-Kun Lai received his bachelor's degree and Ph.D. degree in computer science from Tsinghua University in 2003 and 2008, respectively. He is currently a Professor in the School of Computer Science & Informatics, Cardiff University. His research interests include computer graphics, geometry processing, image processing and computer vision. He is on the editorial boards of *Computer Graphics Forum* and *The Visual Computer*.



Leonidas Guibas is a professor in Stanford University. He obtained his Ph.D. from Stanford in 1976 under the supervision of Donald Knuth. He has been at Stanford since 1984 as Professor of Computer Science. He is a member of the US National Academy of Engineering and the American Academy of Arts and Sciences, an ACM Fellow, an IEEE Fellow and winner of the ACM Allen Newell Award, the ICCV Helmholtz prize, and a DoD Vennevar Bush Faculty Fellowship.

Jie Yang received a bachelor's degree in mathematics from Sichuan University and a Ph.D. degree in computer science from Institute of Computing Technology, Chinese Academy of Sciences. He is currently an Assistant Professor at the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include computer graphics and geometric processing.