Deep Generative Modeling for Scene Synthesis via Hybrid Representations

ZAIWEI ZHANG, ZHENPEI YANG, The University of Texas at Austin, USA
CHONGYANG MA, Kuaishou Technology, China
LINJIE LUO, ByteDance Inc.
ALEXANDER HUTH, ETIENNE VOUGA, QIXING HUANG, The University of Texas at Austin, USA

Fig. 1. For each block, we show a randomly generated scene (top) and its closest scene (bottom) in the training set according to the matrix encoding. Our approach can generate realistic and diverse 3D scenes that are not present in the training dataset.

We present a deep generative scene modeling technique for indoor environments. Our goal is to train a generative model using a feed-forward neural network that maps a prior distribution (e.g., a normal distribution) to the distribution of primary objects in indoor scenes. We introduce a 3D object arrangement representation that models the locations and orientations of objects, based on their size and shape attributes. Moreover, our scene representation is applicable for 3D objects with different multiplicities (repetition counts), selected from a database. We show a principled way to train this model by combining discriminative losses for both a 3D object arrangement representation and a 2D image-based representation. We demonstrate the effectiveness of our scene representation and the network training method on benchmark datasets. We also show the applications of this generative model in scene interpolation and scene completion.

CCS Concepts: • **Computing methodologies → Mesh geometry models**;

Additional Key Words and Phrases: scene synthesis; generative modeling; generative adversarial network; hybrid 3D representations

ACM Reference Format:
Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, and Alexander Huth, Etienne Vouga, Qixing Huang. 2020. Deep Generative Modeling for Scene Synthesis via Hybrid Representations. *ACM Trans. Graph.* 37, 6, Article 1 (July 2020), 21 pages. https://doi.org/0000001.0000001_2

1 INTRODUCTION

Developing automatic tools to generate 3D scenes is a long-standing problem in computer graphics. This problem is challenging because 3D scenes show great variabilities in the appearing objects and their shapes, locations and orientations. Existing 3D scene generation approaches [Chaudhuri et al. 2013, 2011; Fisher et al. 2012; Ha and Eck 2017; Izadinia et al. 2016; Kermani et al. 2016; Li et al. 2017; Liu et al. 2014; Ritchie et al. 2016; Sharma et al. 2017; Wang et al. 2018a;

Authors' addresses: Zaiwei Zhang, Zhenpei Yang, The University of Texas at Austin, 2317 Speedway, Austin, TX, 78712, USA; Chongyang Ma, Kuaishou Technology, Beijing, China; Linjie Luo, ByteDance Inc. Alexander Huth, Etienne Vouga, Qixing Huang, The University of Texas at Austin, 2317 Speedway, Austin, TX, 78712, USA.

0730-0301/2020/7-ART1 $15.00
https://doi.org/0000001.0000001_2

Zou et al. 2017] have predominantly focused on using recursive schemes, i.e., they start from a root object, and iteratively insert new objects into the scene while preserving spatial relations among the objects. Those methods include the early works on query-based data-driven scene synthesis [Chaudhuri et al. 2013, 2011; Fisher et al. 2012] to more recent works that use neural networks to model the recurrent synthesis procedure [Sharma et al. 2017; Wang et al. 2018a; Zou et al. 2017]. However, such recurrent schemes are inherently sub-optimal because existing objects in the scene are not modified after inserting new models or otherwise one has to run costly global optimization to jointly optimize the scene layout (c.f. [Fisher et al. 2012; Yu et al. 2011]). In addition, these approaches do not explicitly establish a mapping, e.g., from a latent parameter space to the space of 3D scenes, making them not suited for applications such as scene interpolation and scene extrapolation.

In this paper, we introduce an approach that generates a 3D scene using a feed-forward neural network. This network takes a random sample from a prior distribution in the latent space as input and outputs a 3D scene described as an arrangement of objects. This setting is applicable to typical internet 3D scenes that are given by flat organizations of objects and do not possess hierarchical decompositions of objects. In addition, a feed-forward parameterization of 3D scenes enjoys many direct applications such as 3D scene completion and 3D scene reconstruction via constrained optimization. However, training a feed-forward 3D scene generator poses several great challenges, including parameterizing 3D scenes to encode both the continuous and discrete variabilities of 3D scenes, developing suitable neural networks and training procedures to capture geometric correlations among multiple objects, and training from unorganized 3D scenes that do not obviously possess consistent orientations and scalings.

Our approach addresses the challenges of learning a feed-forward 3D scene generator by combining three key ideas, ranging from representations of 3D scenes, network design and training, and joint learning of scene generators and poses of the scenes used for training the generator. Specifically, to parameterize a 3D scene, we maintain a superset of abstract objects. Each scene is represented by selecting a subset of objects, and then determining the geometric shape, location, size and orientation of each object. This allows us to parameterize a 3D scene as a matrix, where each column specifies whether the corresponding object appears in the scene or not, and if so, its geometric attributes. In particular, we introduce latent permutation variables to factor out the invariance of this representation when shuffling the columns of this matrix. Given this matrix encoding, we introduce a sparse-dense generative network for generating 3D scenes. This network design effectively addresses the overfitting issue that exists in fully connected networks while simultaneously preserves its expressive power. To further enhance the quality of the resulting 3D generator, we train the network by combining two loss terms, each of which is based on a particular geometric representation. The first loss term employs a standard VAE-GAN [Larsen et al. 2016] loss under the matrix encoding. The second loss term projects the generated scenes onto an image domain and uses discriminators with convolution layers to capture geometric relations among adjacent objects. Moreover, instead of fixing the scenes for training the 3D generator, we introduce latent pose variables for the training instances and optimize them together with the 3D generator, making our approach applicable when training from unorganized 3D scene collections. We show how to integrate these ideas into a single objective function for learning the scene generator and perform alternating minimization to solve the induced optimization problem.

We have applied our approach on synthesizing living rooms and bedrooms using the SUNCG dataset [Song et al. 2017]. The living room and bedroom categories consist of 6401 and 8054 scenes respectively. For each category, we use 6000 and 7000 scenes for training respectively and leave the remaining scenes as testing. Our approach trains 3D scene generators in 1,001 minutes and 1,198 minutes, respectively, using a desktop with 3.2GHZ CPU, 64GB main memory and a GTX 1080 GPU. The trained generators can synthesize diverse and novel 3D scenes that are not present in the training sets (See Figure 1). Synthesizing one scene takes 30 ms. We present quantitative evaluations to justify our design choices. We also show the usefulness of the approach on applications of scene interpolation/extrapolation and scene completion.

In summary, we present the following main contributions:

- We show that it is possible to train a feed-forward parametric generative model that maps a latent parameter to a 3D indoor scene. The 3D scene is represented as an arrangement of 3D objects, and each category of objects may repeat multiple times.
- We introduce a methodology for 3D scene synthesis using hybrid representations, which combines a 3D object arrangement representation for capturing coarse object interactions and an image-based representation for capturing local object interactions.

2 RELATED WORKS

Hand-crafted generative models. Early work in parametric shape modeling consists of shapes designed by domain experts. Examples include work for trees [Weber and Penn 1995] and Greek doric temples [Teboul 2011]. It can be prohibitively difficult, however, for humans to model complicated object classes that exhibit significant geometric and/or topological variability. For this reason, parametric models (or procedural models) for many model classes (e.g., furniture shapes and scenes) do not exist.

Learning generative models. To faithfully capture shape variability in geometric data, a recent focus in visual computing is to learn parametric models from data. This trend is aligned with the significant growth of visual data available from online resources such as ImageNet [Deng et al. 2009] and ShapeNet [Chang et al. 2015]. Methods for learning parametric models differ from each other in terms of the representation of the visual data as well as the mapping function. Early works on learning parametric models focus on Faces and Human bodies [Allen et al. 2003; Anguelov et al. 2005; Blanz and Vetter 1999], which can be parametrized by deforming a template model. The parametric models are given by linearly blending exemplar models in a model collection. Such a method is only applicable to object classes with small geometric variability and

no topological variability. They are not suitable for indoor scenes that can exhibit significant topological and geometrical variability.

Motivated from the tremendous success of deep neural networks, a recent focus has been on encoding the mapping function using neural networks. In the 2D image domain, people have developed successful methods for deep generative models such as generative adversarial networks (GANs) [Arjovsky et al. 2017; Goodfellow et al. 2014; Salimans et al. 2016; Zhao et al. 2016], variational autoencoders [Kingma et al. 2016; Kingma and Welling 2013], and autoregression [Van Den Oord et al. 2016]. Although these approaches work well on 2D images, extending them to 3D data is highly non-trivial. A particular challenge is to develop a suitable representation for 3D data.

3D representations. Unlike other modalities that naturally admit vectorized representations (e.g., images and videos), there exists great flexibility when encoding 3D geometry in their vectorized forms. In the literature, people have developed neural networks for multi-view representations [Qi et al. 2016; Su et al. 2015; Tatarchenko et al. 2016], volumetric representations [Häne et al. 2017; Klokov and Lempitsky 2017; Riegler et al. 2017; Tulsiani et al. 2017a; Wang et al. 2017; Wu et al. 2016, 2015], point-based representations [Qi et al. 2017], part-based representations [Li et al. 2017; Tulsiani et al. 2017b], graph/mesh representations [Henaff et al. 2015; Masci et al. 2015; Monti et al. 2017; Yi et al. 2016] and spherical representations [Cao et al. 2017; Cohen et al. 2018; Esteves et al. 2017].

Existing methods for building parametric 3D models have mostly focused on 3D shapes. [Wu et al. 2016] describe a 3D generative network under the volumetric representation. Extending this approach to 3D scenes faces the fundamental challenge of limited resolution. In addition, its output is not an arrangement of objects. [Tulsiani et al. 2017b] proposed a part-based model for synthesizing 3D shapes that are described as an arrangement of parts. [Nash and Williams 2017] proposed ShapeVAE for synthesizing 3D shapes that are described as a semantically labeled point cloud. Both approaches are specifically tailored for 3D shapes, and it is challenging to extend them to 3D scenes. For example, both approaches require that shapes are consistently oriented, and such orientations are not available for 3D scenes. In our approach, we jointly optimize both the generators and the orientations of the input scenes. In addition, we found that variations in 3D scenes are more significant than 3D shapes, and approaches which work well on shapes generally lead to sub-optimal results on 3D scenes, e.g., spatial relations between correlated objects are not captured well. This motivates us to develop new representations and training methods for 3D scenes.

The difference between our approach and existing 3D synthesis approaches is that we combine training losses under two representations, i.e., an object arrangement representation and an image-based representation. This innovative design allows us to obtain globally meaningful and locally compatible synthesis results.

Assembly-based geometric synthesis. Currently the dominant 3D scene synthesis method is assembly-based. [Funkhouser et al. 2004] introduced the first system that generates new 3D models by assembling parts from existing models. People have also applied this concept for various applications such as interactive modeling [Kreavoy et al. 2007], design [Chaudhuri and Koltun 2010], reconstruction [Huang et al. 2015; Shen et al. 2012], and synthesis [Xu et al. 2012]. The advantage of these methods is that they can handle datasets with significant structural variability. The downside is that these methods require complicated systems and careful parameter tuning. To improve system performance, a recent line of works utilize probabilistic graphical models (e.g., Bayesian networks) for assembly-based modeling and synthesis [Chaudhuri et al. 2013, 2011; Chen et al. 2014; Fisher et al. 2012; Izadinia et al. 2016; Kalogerakis et al. 2012; Kermani et al. 2016; Liu et al. 2014; Merrell et al. 2010; Sung et al. 2017; Xu et al. 2013]. Along this line, several works [Fisher et al. 2015; Jiang et al. 2012; Ma et al. 2016, 2018; Qi et al. 2018; Savva et al. 2016; Shao et al. 2012; Wang et al. 2018b, 2019; Zhu et al. 2018] focus on using human interactions with objects and/or human actions to guide the synthesis process. These methods significantly stabilize the modeling and synthesis process. The nodes and edges in the graphical models, however, are usually pre-defined, which necessitates significant domain knowledge.

Another recent line of works [Ha and Eck 2017; Li et al. 2017, 2019; Ritchie et al. 2016, 2019; Sharma et al. 2017; Wang et al. 2018a; Zou et al. 2017] reformulate assembly-based synthesis as recursive procedures. Starting from a root part, these methods recursively insert new parts conditioned on existing parts. This conditional probability is described as a neural network. In contrast, our approach proposes to learn 3D synthesis using a feed-forward network. In particular, our approach does not require hierarchical labels (either provided by users or generated computationally) that are required for training such recursive procedures.

Priors learned from training data can be used for rectifying 3D scenes as well. [Yu et al. 2011] present an optimization framework for turning a coarse object arrangement into significantly improved object arrangements. Our image-based discriminator loss is conceptually similar to this approach, yet we automatically learn this loss term from data.

Image-based representation for 3D synthesis. Several recent works leverage image-based representations for 3D synthesis. In [Zou et al. 2018], the authors leverage the image-based representation to predict the locations of key objects in a scene. In [Wang et al. 2018a], the authors use an image-based representation to predict locations and other attributes of the object to be inserted. In contrast, our approach learns a parametric 3D generator for synthesis. The image-based representation, which serves as a regularizer for the 3D generator, is only used in the training process.

3 OVERVIEW

In this section, we give an overview of the 3D scene synthesis problem (Section 3.1) and of the proposed approach for solving it (Section 3.2).

3.1 Problem Statement

Our goal is to train a neural network that takes a random variable $\mathbf{z} \in \mathbb{R}^d$ as input and generates a 3D scene, i.e., the parameter $\mathbf{z}$ represents a low-dimensional encoding of the scene. In this paper, we represent a scene as a collection of rigid objects arranged in space

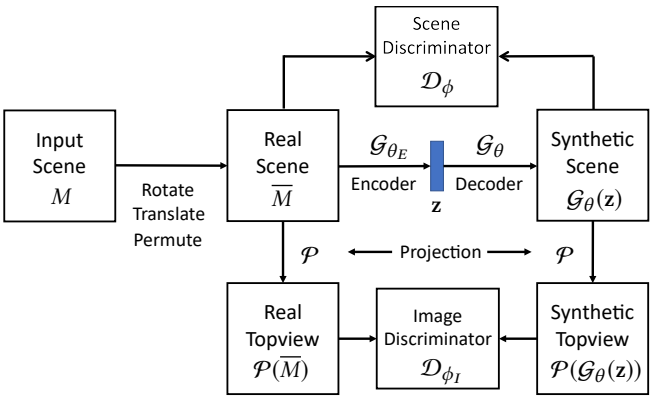

Fig. 2. Illustration of different modules of our design. Encoder ($\mathcal{G}_{\theta_E}$) encodes the input scene to latent embedding, Decoder ($\mathcal{G}_\theta$) generates scene data matrices based on latent embedding, Scene Discriminator ($\mathcal{D}_\phi$) classifies if the input scenes are real or not, $\mathcal{P}$ means the projection layer which projects 3D scenes to 2D top-view images, and Image Discriminator ($\mathcal{D}_{\phi_I}$) classifies if the input top-view images are real or not.

in a semantically meaningful way, and free of interpenetration. We assume each object belongs to one of a predefined set of object classes, and that objects within each class can be parameterized by a shape descriptor (this descriptor is then used to retrieve the object's 3D geometry from a shape database). We further assume that objects rest on the ground, and are correctly oriented with respect to the vertical direction, so that each object's placement in the scene can be specified by an orientation and position in the xy plane (the *top view*), as well as a nonuniform scaling. We formalize the scene representation in Section 4.1.

To train our networks, we use N different 3D scenes $S_1, \cdots, S_N$ gathered from SUNCG [Song et al. 2017]. We do not require that the scenes are globally oriented in a consistent way, that objects are specified in any particular order, etc; our training formulation is robust to such variations. In addition, our approach does not require additional local or global supervision such as a hierarchical grouping of objects in a scene.

3.2 Approach Overview

There are three key factors in training a high-quality 3D generator described above: (1) How to parameterize 3D scenes; (2) How to design and train the 3D scene generator; and (3) How to train the generator from unorganized scene collections. Our approach parameterizes a 3D scene by selecting objects from an over-complete set of objects, allowing us to parameterize a 3D scene as a matrix, whose elements specify which objects appear and their locations, orientations, sizes and geometric attributes. The key contribution of this paper is on how to design and train the 3D generator. Under the matrix representation, a typical strategy is to use fully connected layers to design the generator network. Such a network architecture, however, easily leads to overfitted 3D generators with poor generalization behavior. We present two key innovations. The first one is to use a sparse-dense network to encode the 3D generator, which greatly improves the generalization behavior. The second one is to project a generated 3D scene onto an image domain, i.e., on the ground-plane along the vertical direction, and use a discriminator with convolution layers to capture fine-grained spatial correlations among the input objects. This discriminator can improve the quality of the 3D generator considerably. In other words, we train the 3D scene generator under a hybrid representation, which allows us to add the strength of the learned features under each representation together (see Figure 2). Finally, to handle unorganized scene collections we introduce a pose variable for each scene in the training set and perform alternating minimization to optimize the scene poses and the 3D scene generator and affiliated variables (e.g, the discriminators).

The remainder of this section summarizes the design and motivation of each component of our design, and Section 4 will spell out the technical details.

Object arrangement scene representation. We represent a 3D arrangement using a matrix whose columns correspond to the objects in the scene. In other words, each 3D scene is specified by selecting some number of objects of each object class and then arranging/fixing them in 3D space. Each column of the matrix representation describes the status of the corresponding object, namely, whether it appears in the scene or not, its location, size, orientation and shape. Notice that while each matrix completely specifies a 3D scene, this representation is redundant. To handle the technical challenge of non-uniqueness of this encoding (i.e., shuffling columns of the same category leads to the same scenes), we introduce latent permutation variables which effectively factor out such permutation variability.

Scene generator. We design the scene generator as a feed-forward network with interleaved sparsely and fully connected layers operating on the matrix representation of scenes. The motivation for this architecture comes from the observations that (1) correlations among objects in an indoor scene are usually of low-order, and (2) sparsely connected layers have significantly reduced model size compared to fully connected networks, which reduces generalization error.

Image-based module. We leverage a CNN-based discriminator loss, which captures the object correlations based on local object geometry that cannot be effectively captured by the matrix-encoding-based discriminator loss. Specifically, we encode each 3D scene as a 2D projection of the objects onto the xy plane. We impose a CNN-based image discriminator loss on this 2D image, which is back propagated to the scene generator, forcing it more accurately learn local correlation patterns.

Joint scene alignment. Despite our fairly intuitive network design, training the network from unorganized scene collections is difficult. One challenge is that the training scenes are not necessarily globally oriented in a consistent way, nor do objects have consistent absolute locations. Moreover, although objects can be grouped by class, there is no canonical ordering of objects within the same class. To address these issues, we solve a global optimization problem to jointly align the input scenes in a training preprocessing step. We found that first aligning the input scenes significantly improves the resulting 3D scene generator.

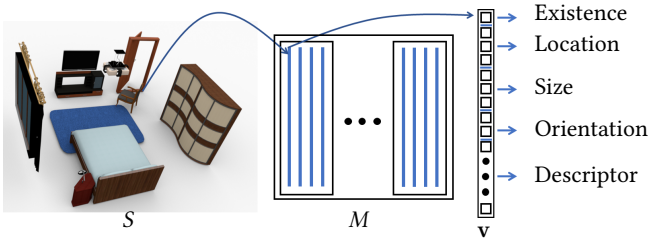

Fig. 3. Illustration of our scene encoding scheme. Existence determines if the object is in the scene or not. Location, Size and Orientation represent the 3D location, Size and Orientation of the object. Descriptor represents the shape geometry of the object.

Network training. Given roughly aligned input scenes, we learn the generator by optimizing an objective function that combines an autoencoder loss and the two discriminator losses described above. The variables include the generator network, two discriminators, the pose of each scene, and the orderings of the objects in each 3D scene. To facilitate the optimization, we introduce a latent variable for the scene that characterizes its underlying configuration (i.e., after transformation and object re-ordering). Both the autoencoder loss and discriminator losses are defined on this latent variable. In addition, we penalize the difference between this latent scene and its corresponding input scene (after transformation and object re-ordering). In doing so, the optimization variables are nicely decoupled, allowing efficient optimization via alternating minimization.

4 APPROACH

In this section, we present technical details of our approach. In Section 4.1, we present a matrix representation of 3D object arrangement. In Section 4.2 and Section 4.3, we describe the 3D object arrangement module and the image-based module, respectively. In Section 4.4, we introduce how to jointly align the input scenes. Finally, we describe network training procedure in Section 4.5.

4.1 Scene Representation

To parameterize 3D scenes, we assume each object belongs to one of n_c object categories, and that scenes can contain up to m_k objects of each class k. Each scene therefore contains a maximum of $n_o = \sum_{k=1}^{n_c} m_k$ objects O. In our experiments, we use $n_c = 30$ and $m_k = 4, 1 \le k \le n_c$ (see Section 5 for details). Note that another alternative encoding is to allow n_o total number of arbitrary objects (c.f. [Fan et al. 2017; Tulsiani et al. 2017b]). However, we found that explicitly encoding the class label of each object is more efficient than synthesizing the class label of each object, particularly when the number of distinctive classes is large. In fact, we otherwise have to introduce one additional element for each class and a large permutation matrix for each scene whose dimension is equal to the total number of objects.

We assume that objects in each class can be uniquely identified with a d-dimensional shape descriptor, with d constant across all classes. We can thus encode each object $o \in O$ using a status vector $\mathbf{v}^o \in \mathbb{R}^{d+9}$:

- v_0^o is a tag that specifies whether o appears in the scene ($v_0^o \ge 0.5$) or not ($v_0^o < 0.5$).
- $(v_1^o, v_2^o, v_3^o)^T$ specifies the center of the bounding box of o in a world coordinate system; we assume the up orientation of each object is always along the z-axis of this world coordinate system.
- $(v_4^o, v_5^o)^T$ specifies the front-facing orientation of the bounding box of o in the top view (xy plane).
- v_6^o, v_7^o, and v_8^o specify the size of the bounding box of o in the front, side, and up directions, respectively.
- $(v_9^o, \cdots, v_{d+8}^o)^T$ is the aforementioned descriptor that characterizes the geometric shape of c. In this paper, we use as our descriptor the second-to-last layer of the volumetric module of Qi et al. [Qi et al. 2016] pre-trained on ShapeNetCore [Chang et al. 2015].

Given this object representation, a 3D scene can be parameterized by a matrix $M \in \mathbb{R}^{(d+9)\times n_o}$, with blocks of columns $M_k \in R^{(d+9)\times m_k}$ containing the status vectors of the objects of the k-th category.

One technical challenge of this intuitive encoding of 3D scenes is that it is invariant to permutations of columns of the M_k. In addition, the location and orientations of each object are dependent on the global pose of each scene. In light of this observation, we introduce two operators on the matrix encoding M.

The first operator applies column permutations σ_k to objects of each class:

$$\mathcal{S}(M; \sigma_1, \ldots, \sigma_{n_k}) :$$
$$\mathbb{R}^{(d+9)\times n_o} \times \prod_{k=1}^{n_c} S_{m_k} \to \mathbb{R}^{(d+9)\times n_o}$$
$$\begin{bmatrix} M_1 & \cdots & M_{n_c} \end{bmatrix} \mapsto \begin{bmatrix} M_1\sigma_1 & \cdots & M_{n_c}\sigma_{n_c} \end{bmatrix}$$

To avoid clutter, in what follows we elide the explicit dependence of $\mathcal{S}$ on the σ. Note that $\mathcal{S}$ applies permutations to objects of each class independently.

The second operator $\mathcal{T}(M; R, \mathbf{t})$ applies a rotation $R \in SO(2)$ about the z axis, and an arbitrary translation $\mathbf{t} \in \mathbb{R}^3$, to the bounding box position encoded within each column of M, and likewise applies R to each object orientation. Again, we will elide R and $\mathbf{t}$ for convenience.

We associated a set of latent variables $\{\sigma_k, R, \mathbf{t}\}$ to each scene i; we will use the notation $\mathcal{S}_i$, $\mathcal{T}_i$, etc, to denote the permutation latent variables and the transformation latent variables.

We factor out permutations of objects and the global pose of each input scene by introducing a latent matrix encoding $\overline{M}_i \in \mathbb{R}^{(d+9)\times n_o}$ for each scene i. The antoencoder and discriminator losses described below will be imposed on $\overline{M}_i$, and we enforce consistency of $\overline{M}_i$ and M_i by minimizing the loss term below:

$$f_d\left(\overline{M}_i, M_i\right) = \min_{\mathcal{T}_i, \mathcal{S}_i} \left\| \overline{M}_i - (\mathcal{T}_i \circ \mathcal{S}_i)(M_i) \right\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ denotes the matrix Frobenius norm. Initialization and optimization of the latent permutation and rigid transformation variables are described below in section 4.4.

4.2 3D Object Arrangement Module

At the heart of our design are two networks:

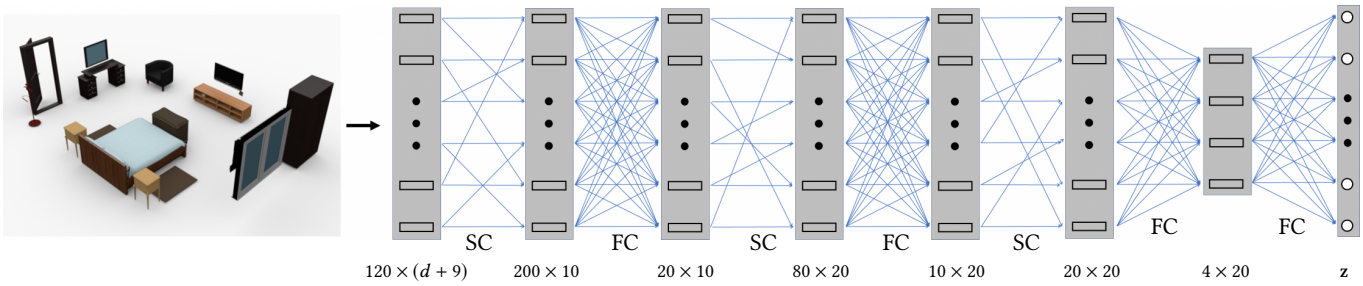

Fig. 4. This figures illustrates the network module that is used for the encoder $\mathcal{G}_{\theta_E}$. The decoder $\mathcal{G}_\theta$ is reversed from the encoder $\mathcal{G}_{\theta_E}$. The arrangement discriminator $\mathcal{D}_\phi$ shares the same network architecture but replaces the latent vector by a value. This network module interweaves between sparsely connected (or SC) layers and fully connected (or FC) layers.

Fig. 5. Visual comparisons between synthesized scenes using different generators. Left (5a): training a fully connected generator network; Left (5b): training a fully convolutional generator network; Right (5c): training a sparsely connected generator network; Right (5d): training a combination of sparsely connected generator network and an image-based discriminator. Unreasonable object placements in 5c are marked with red circles. (The fourth column was generated by directly optimizing the scenes in the second column so that the trained image-based discriminator predicts them to be real.)

- the encoder network $\mathcal{G}_{\theta_E} : \mathbb{R}^{(d+9)\times n_o} \to \mathbb{R}^k$, and
- the decoder network $\mathcal{G}_\theta : \mathbb{R}^k \to \mathbb{R}^{(d+9)\times n_o}$.

Since our matrix encoding of 3D scenes is essentially a vectorized representation (in contrast to an image-based representation), it is natural to use fully connected (FC)-type layers for both the generator network and the encoder network. However, we observed that the naive approach of connecting all pairs of nodes between consecutive layers does not work. Our experiments indicated that this approach easily overfits the training data, so that the generated scenes are of poor quality (See Figure 5(a)). In addition, we have used convolution layers instead of the FC-type layers, by considering matrix encoding as gray scale image representation. Our experiments indicated that this approach does not learn pairwise object relations (See Figure 5(b), Figure 12 and Figure 13).

To address this overfitting issue, we propose to use sparsely connected layers. Each node of one layer is only connected to h nodes of the previous layers. In our implementation, we set $h = 4$ and randomize the connections, i.e., each node independently connects with a node in the previous layer with probability h/L, where L is the number of nodes in the previous layer. As illustrated in Figure 4, our network interleaves between sparsely connected layers and fully connected layers. We still keep some fully connected layers to give the network sufficient expressiveness for network fitting. Note that the connections between nodes remain fixed during the training process.

There are two motivations for using sparsely-connected layers for $\mathcal{G}_\theta$. First, patterns in 3D scenes usually involve small groups of objects [Fisher and Hanrahan 2010; Fisher et al. 2012, 2011], e.g. chairs and tables, or nightstands and beds, so that sparse relationships between object classes are expected. Second, from the perspective of optimization, sparsely-connected networks have significantly reduced model size, and thus tend to avoid overfitting and have improved generalization. In the broader picture, neural networks exhibit exponential expressiveness (c.f. [Poole et al. 2016]), and training compressed networks yield comparable and sometimes better generalization performance [Han et al. 2015, 2016].

Following DCGAN [Radford et al. 2015], we set the architecture of $\mathcal{G}_\theta$ to the reverse of that of $\mathcal{G}_{\theta_E}$. We use VAE-GAN [Larsen et al. 2016] for training both the encoder and decoder networks:

$$f_o = \frac{1}{N}\sum_{i=1}^{N}\left\|\mathcal{G}_\theta\mathcal{G}_{\theta_E}(\overline{M}_i) - \overline{M}_i\right\|_F^2 + KL\left(\{\mathcal{G}_{\theta_E}\left(\overline{M}_i\right)\}, p\right) + \lambda\left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{D}_\phi\left(\overline{M}_i\right) - E_{\mathbf{z}\sim p}\mathcal{D}_\phi\left[G_\theta(\mathbf{z})\right]\right). \quad (2)$$

where the latent distribution p is the standard normal distribution, and the discriminator $\mathcal{D}_\phi$ has the same network architecture as $\mathcal{G}_{\theta_E}$, except that we replace the latent vector by one node.

4.3 Image-Based Module

As discussed in the overview, we introduce a second, image-based discriminator to better capture local arrangement of objects based on geometric detail, such as the spatial relations between chairs and tables and those between nightstands and beds in generated scenes. In our experiments, we found that it is hard to precisely

capture such patterns by merely using FC-type discriminators $\mathcal{D}_\theta$. As illustrated in Figure 5(c), without the image-based module, the learned object arrangement generator $\mathcal{G}_\theta$ exhibits various local compatibility issues (e.g., objects intersect with each other).

Motivated by the fact that CNN-based discriminators [Radford et al. 2015] can nicely capture local interaction patterns among adjacent objects, we propose to convert the 3D object arrangement into a 2D image by projecting the 3D scene onto the top view, and to then impose a CNN-based discriminator on this 2D image. Specifically, let $\mathcal{P} : \mathbb{R}^{(d+9)\times n_o} \to \mathbb{R}^{r\times r}$ be the projection operator onto an $r \times r$ image ($r = 128$, and details of the projection operator are described in detail below). Denote $\mathcal{D}_{\phi_I}$ as the discriminator for the image-based representation, where ϕ_I represents the network parameter. In our experiments, we used ResNet-18 [He et al. 2016], an established CNN network capable of capturing multi-scale patterns of an image, as our discriminator.

We then use the following discriminator loss for learning the object arrangement generator:

$$f_I = \frac{1}{N}\sum_{i=1}^{N} \mathcal{D}_{\phi_I}\left(\mathcal{P}\left[\overline{M}_i\right]\right) - E_{\mathbf{z}\sim p}\mathcal{D}_{\phi_I}\left(\mathcal{P}\left[\mathcal{G}_\theta(\mathbf{z})\right]\right). \tag{3}$$

Rather than projecting the 3D scene to the top view, another option is to convert the scene to a volumetric grid and impose 3D CNN-based discriminator. However, this approach has severe limits on tractable grid resolution (e.g. 64^3) and cannot accurately resolve local geometric detail. On the other hand, most local correlations are revealed in the top view [Wang et al. 2018a; Zou et al. 2018], which provides sufficient supervision for learning the generator.

Although it is possible to use a rendering operator for the projection $\mathcal{P}$, as as described by Wang et al. [Wang et al. 2018a], we want the image-based discriminator $\mathcal{D}_{\phi_I}$ to provide smooth gradients for the generator $\mathcal{G}_\theta$, and such gradients are hard to compute even when using very simple rendering operations. We therefore instead define a fuzzy projection operator $\mathcal{P}$ in terms of summed truncated signed distance fields of objects projected into the top view. Specifically, for each object o, let $E_o(M)$ denote the set of points in the plane computed by (1) embedding object o in 3D as encoded by the parameters in M, and (2) orthogonally projecting that object onto the xy plane. Denote the truncated signed distance function of object o by

$$d_{o,\delta} : \mathbb{R}^2 \to \mathbb{R}$$
$$\mathbf{p} \mapsto \begin{cases} d[\mathbf{p}, \partial E_o(M)], & d[\mathbf{p}, \partial E_o(M)] \le \delta, \mathbf{p} \notin E_o(M) \\ -d[\mathbf{p}, \partial E_o(M)], & d[\mathbf{p}, \partial E_o(M)] \le \delta, \mathbf{p} \in E_o(M) \\ 0, & d[\mathbf{p}, \partial E_o(M)] > \delta. \end{cases}$$

Let $d^I_{o,\delta} \in \mathbb{R}^{r\times r}$ be the rasterization of $d_{o,\delta}$ onto an $r \times r$ image. We then define the projection operator as

$$\mathcal{P}(M) := \sum_{o\in M} c_o d^I_{o,M}, \tag{4}$$

where c_o is a class-specific constant associated with object o. In our implementation, we simply use the index of the category label of o (See Appendix D). For fixed M, the gradient of $\mathcal{P}(M)$ with respect to M can be computed; see Appendix C for details.

4.4 Joint Scene Alignment

As a preprocessing step, we align all input training scenes, by assigning each scene a rigid transformation and set of permutations, as described in section 4.1. We follow the common two-step procedure for establishing consistent transformations (maps) across a collection of objects [Huang et al. 2006; Huang and Guibas 2013; Huber 2002; Kim et al. 2012], namely, we first perform pairwise matching, and then aggregate these pairwise matches into a consistent global alignment of all scenes. A common feature of such two-step approaches is that the second step can effectively remove noisy pairwise matches computed in the first step [Huang and Guibas 2013], leading to high-quality alignments. In our case, simultaneously optimizing for each scene's optimal rigid transformation and permutations is intractable for large-scale data (i.e. several thousands of scenes). We therefore propose to align the input scenes in a sequential manner, by first optimizing rotations, then translations and finally permutations.

Pairwise matching. Given a pair of scenes M^i and M^j, we solve the following optimization problem to determine the optimal transformation $\mathcal{T}^{in}_{ij} = \left(R^{in}_{ij}, \mathbf{t}^{in}_{ij}\right)$ aligning M_i to M_j, as well as permutations $\mathcal{S}^{in}_{ij}$ mapping objects of each class in M^i to their closest match in M^j:

$$\mathcal{T}^{in}_{ij}, \mathcal{S}^{in}_{ij} = \underset{\mathcal{T},\mathcal{S}}{\operatorname{argmin}} \left\|(\mathcal{T}\circ\mathcal{S})\left(M^i\right) - M^j\right\|_{2,1}, \tag{5}$$

where $\|A\|_{2,1} = \sum_{j=1}^{m} \|\mathbf{a}_j\|_\star, A := (\mathbf{a}_1, \cdots, \mathbf{a}_m)$ is a robust norm used to handle continuous and discrete variations between M_i and M_j.

We solve Equation (5) by combining the method of reweighted least squares [Daubechies et al. 2010] and alternating minimization. Since this step is not the major contribution of the paper, we defer the technical details, as well as the precise definition of $\|\|_\star$, to Appendix B.

Computing pairwise alignments of all pairs of scenes is infeasible. We follow the procedure of Heath et al. [Heath et al. 2010] by connecting each scene with $k = 64$ neighboring scenes in the training set. To compute these nearest neighbors, we assign to each scene an n_c-dimensional vector that counts how many objects of each class appear in the scene, and compute nearest neighbors via L2-distance between these vectors.

We expect some pairwise alignments to be noisy (see for instance Figure 6). We address this issue by a second, global alignment refinement step (map synchronization).

Consistent scene alignment. We employ state-of-the-art map synchronization techniques for joint optimization of orientations, translations, and permutations associated with the input scenes. For efficiency, we optimize orientations, translations, and permutations in a sequential manner. For rotation synchronization, we employ the method of Chatterjee and Govindu [Chatterjee and Govindu 2013], which optimizes the orientation R_i associated with each scene S_i via Huber loss. For translation synchronization, we use the method of Huang et al. [Huang et al. 2017], which applies truncated least squares to optimize the translation of $\mathbf{t}_i$ associated with each scene S_i. Finally, we employ normalized spectral permutation synchronization [Shen et al. 2016] to optimize the permutation $\sigma_{i,k}$ associated with each category c_k of each scene S_i. Since our approach directly

applies these techniques, we refer to these respective papers for technical details.

Note that all of these approaches can tolerate a significant amount of noise in the pairwise alignments. As a result, alignments substantially improve after the map synchronization step (See Figure 6). The bottleneck for the scene alignment step is performing pair-wise scene alignments, whose complexity is $O(m)$, where m is the number of pair-wise alignments. In our experiments, the alignment process takes around 6 hours for 7000 bedroom scenes, and the alignment process takes around 5 hours for 6000 living room scenes.

4.5 Network Training

Finally, given the consistently aligned scenes, we proceed to learn the object arrangement generator G_θ, the object arrangement encoder G_{θ_E}, and the two discriminators $\mathcal{D}_\phi$ and $\mathcal{D}_{\phi_I}$. Combining equations (1)), (2), and (3), we arrive at the following objective function:

$$\begin{aligned}\max_{\phi,\phi_I}\min_{\theta,\theta_E}\ & \frac{1}{N}\sum_{i=1}^{N}\left\|\mathcal{G}_\theta\mathcal{G}_{\theta_E}\left(\overline{M}_i\right)-\overline{M}_i\right\|_F^2 \\ &+\frac{\gamma}{N}\sum_{i=1}^{N}\min_{\mathcal{T}_i,\mathcal{S}_i}\left\|\overline{M}_i-\left(\mathcal{T}_i\circ\mathcal{S}_i\right)\left(M_i\right)\right\|_F^2 \\ &+\lambda\left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{D}_\phi\left(\overline{M}_i\right)-E_{\mathbf{z}\sim p}\mathcal{D}_\phi\left[\mathcal{G}_\theta(\mathbf{z})\right]\right) \\ &+KL\left(\left\{\mathcal{G}_{\theta_E}\left(\overline{M}_i\right)\right\},p\right) \\ &+\mu\left(\frac{1}{N}\sum_{i=1}^{N}\mathcal{D}_{\phi_I}\left(\mathcal{P}(\overline{M}_i)\right)-E_{\mathbf{z}\sim p}\mathcal{D}_{\phi_I}\left(\mathcal{P}\left[\mathcal{G}_\theta(\mathbf{z})\right]\right)\right). \end{aligned} \quad (6)$$

In this paper, we set $\lambda=1$, $\mu=1$, and $\gamma=100$. The large value in γ ensures that $\overline{M}_i$ and M_i encode approximately the same scene.

Equation (6) is challenging to solve since the objective function is highly non-convex (even when the discriminators are fixed). We again apply alternating minimization for optimization, so that each step solves an easier optimization sub-problem.

4.5.1 Alternating Minimization. We perform two levels of alternating minimization. The first level alternates between optimizing $\left\{\overline{M}_i,\mathcal{S}_i,\mathcal{T}_i,\theta,\theta_E\right\}$ and the discriminators $\{\phi,\phi_I\}$. In the former case, a second level of alternation switches between optimizing θ,θ_E, the $\overline{M}_i$, the $\mathcal{S}_i$ and the $\mathcal{T}_i$.

Generator optimization. When ϕ, ϕ_I, the $\mathcal{S}_i$, the $\mathcal{T}_i$ and the $\mathcal{M}_i$ are fixed, Equation (6) reduces to

$$\begin{aligned}\min_{\theta,\theta_E}\ & \frac{1}{N}\sum_{i=1}^{N}\left\|\mathcal{G}_\theta\mathcal{G}_{\theta_E}\left(\overline{M}_i\right)-\overline{M}_i\right\|_F^2 \\ &-E_{\mathbf{z}\sim p}\mathcal{D}_\phi\left(\mathcal{G}_\theta\left[\mathbf{z}\right]\right)-\mu E_{\mathbf{z}\sim p}\mathcal{D}_{\phi_I}\left(\mathcal{P}\left[\mathcal{G}_\theta(\mathbf{z})\right]\right). \end{aligned} \quad (7)$$

We apply ADAM [Kingma and Ba 2014] for optimization. In all of our experiments, we trained θ and θ_E for two epochs and then moved to optimize other variables.

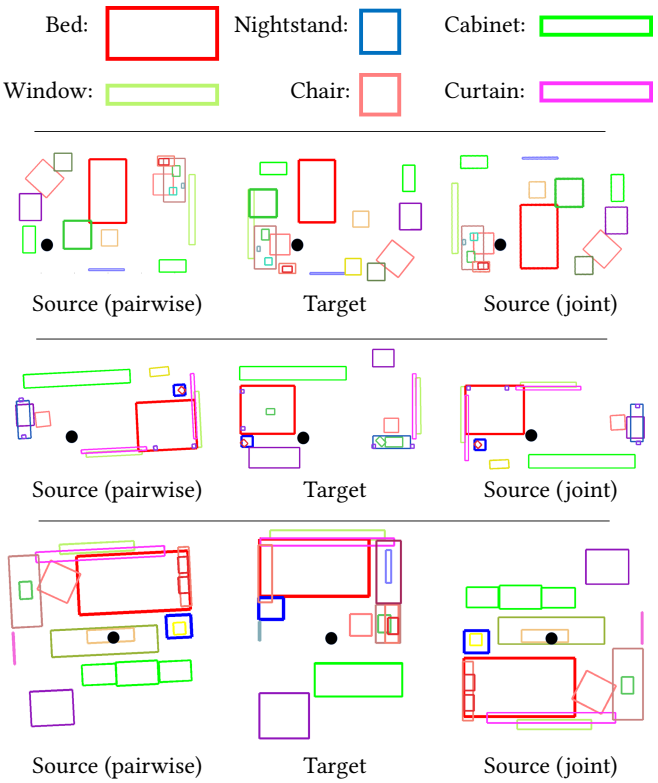

Fig. 6. This figure is best visualized in color. Each row presents the alignment of a pair of scans using pairwise alignment (Left) and joint alignment among 7000 bedroom scenes from SUNCG [Song et al. 2017] (Right). The black dot indicates the relative translations. For simplicity, we show 2D layouts of each scene from the top view. Different categories of objects possess different colors. We can see that joint alignment, which utilize information from the entire collection to determine the pose of each scene, can rectify pairwise misalignments induced from patterns of uncorrelated object groups. In addition, merely aligning the bed objects is sub-optimal, as the locations of bed exhibit significant variations (See the third row).

Latent variable optimization. When θ, θ_E, ϕ, ϕ_I, the $\mathcal{S}_i$ and the $\mathcal{T}_i$ are fixed, we can optimize $\overline{M}_i$ for each scene in isolation:

$$\begin{aligned}\min_{\overline{M}_i}\ & \left\|\mathcal{G}_\theta\mathcal{G}_{\theta_E}\left(\overline{M}_i\right)-\overline{M}_i\right\|_F^2+\gamma\left\|\overline{M}_i-\mathcal{T}_i\left(\mathcal{S}_i(M_i)\right)\right\|_F^2 \\ &+\lambda\mathcal{D}_\phi\left(\overline{M}_i\right)+\mu\mathcal{D}_{\phi_I}\left(\mathcal{P}(\overline{M}_i)\right)\end{aligned}$$

We apply ADAM [Kingma and Ba 2014] for optimization. Since the value of γ is large, the convex potential $\gamma\left\|\overline{M}_i-\mathcal{S}_i(M_i)\right\|_F^2$ strongly dominates the other terms. In our experiments, we found this step usually converges in 8-12 iterations.

Permutation optimization. When the $\overline{M}_i$, the $\mathcal{T}_i$, θ, θ_E, ϕ and ϕ_I are fixed, we can optimize $\sigma_{i,k}$ in each $\mathcal{S}_i$ in isolation:

$$\sigma_{i,k}^\star=\underset{\sigma_{i,k}\in S_{m_k}}{\operatorname{argmin}}\left\|\mathcal{T}_i^{-1}(\overline{M}_{i,k})-M_{i,k}\sigma_{i,k}\right\|_F^2, \quad (8)$$

for $1 \leq k \leq n_c, 1 \leq i \leq N$. It is easy to see that Equation (8) is equivalent to

$$\sigma_{i,k}^{\star} = \underset{\sigma_{i,k} \in S_{m_k}}{\operatorname{argmax}} \left\langle \sigma_{i,k}, \mathcal{T}_i^{-1}\left(\overline{M}_{i,k}\right) M_{i,k}^T \right\rangle$$

which is a linear assignment problem, and can be solved exactly using the Hungarian algorithm.

Transformation optimization. When the $\overline{M}_i$, the $\mathcal{S}_i$, θ, θ_E, ϕ and ϕ_I are fixed, we can optimize each $\mathcal{T}_i$ in isolation:

$$\mathcal{T}_i^{\star} = \underset{\mathcal{T}_i}{\operatorname{argmin}} \left\| \overline{M}_i - \mathcal{T}_i\left(\mathcal{S}_i(M_i)\right) \right\|_F^2. \tag{9}$$

Equation (9) can be formulated as rigid point cloud alignment with known correspondences (orthogonal Procrustes), and we use the closed-form solution described by Horn [Horn 1987].

Discriminator optimization. Finally, when the $\overline{M}_i$, the S_i, θ and θ_E are fixed, the discriminators can be optimized independently as follows:

$$\min_{\phi} \frac{1}{N} \sum_{i=1}^{N} D_{\phi}\left(\overline{M}_i\right) - E_{\mathbf{z}\sim p} \mathcal{D}_{\phi}\left(G_{\theta}(\mathbf{z})\right)$$

$$\min_{\phi_I} \frac{1}{N} \sum_{i=1}^{N} \mathcal{D}_{\phi_I}\left[\mathcal{P}\left(\overline{M}_i\right)\right] - E_{\mathbf{z}\sim p} \mathcal{D}_{\phi_I}\left(\mathcal{P}\left[\mathcal{G}_{\theta}(\mathbf{z})\right]\right).$$

In our experiments, we trained both discriminators for 10 epochs in each alternating minimization.

Termination of alternating minimization. In all of our experiments, we use $t_{max}^{\text{inner}} = 10$ iterations for the inner alternating minimization (i.e., of θ,θ_E, the $\overline{M}_i$, the $\mathcal{S}_i$, and the $\mathcal{T}_i$) and $t_{\max}^{\text{outer}} = 10$ iterations of the outer alternating minimization (between the set of preceding variables and $\{\phi, \phi_I\}$).

5 EXPERIMENTAL EVALUATION

In this section, we present an experimental evaluation of the proposed approach. In Section 5.1, we describe the experimental setup. From Section 5.2 to Section 5.5, we analyze the results of our approach. Section 5.6 and Section 5.7 present the applications our approach in scene interpolation and scene completion, respectively.

5.1 Experimental Setup

Dataset. We perform experimental evaluation on two room types extracted from SUNCG [Song et al. 2017]: *Bedroom* and *Living Room*. SUNCG contains a large number of 3D scenes created by online users using the Planner5d interior design tool [Planner5d 2017]. SUNCG contains more than 45,000 3D scenes, and each scene is segmented into different room types. In this work, we used the 30 most frequent classes. Please refer to Appendix D for the names of these classes and other statistics. For each class, we constrain that the maximum repetition of an object category is 4, and for each room, we constrain that the number of objects must be higher than 15 for bedroom and higher than 13 for living room. Using higher number of object count helps to extract scenes with more features, and the numbers are chosen empirically so that we have more than 6000 training instances, which is enough for the network to learn scene semantics based on the experimental results. We do not constrain the shape of the rooms. Please refer to Appendix D for the statistics on room geometry in the training datasets. We use these constraints to gather all suitable scenes (i.e., all objects fall in those 30 most frequent classes and the largest repetition count is 4, and the number of objects). In total, we collect 8054 and 6401 Bedroom and Living Room scenes, respectively. We select 7000 and 6000 scenes for training, for bedroom and living room respectively. The remaining scenes are left for testing. We directly use the 3D models associated with SUNCG as the shape database for our approach.

Baseline approaches. Since we are unaware of any existing methods that have the exact input and output settings as our approach, we perform evaluation against variants of our approach:

- *Baseline I:* The first baseline removes the image-based module and the scene discriminator, and only applies VAE on the object arrangement representation.
- *Baseline II:* The second baseline removes the image-based module and only applies VAE-GAN on the object arrangement representation.
- *Baseline III:* The third baseline removes the image-based module and changes all the sparse fully connected layers to fully connected layers. For fair comparison, we use similar number of parameters compared to Baseline II.
- *Baseline IV:* The fourth baseline removes the image-based module and applies all convolution layers for the VAE-GAN architecture. The architecture for encoder and scene discriminator is the same as the discriminator used in DCGAN [Radford et al. 2015], and the architecture for decoder is the same as the generator used in DCGAN.

In Section 5.7, we compare our approach with two state-of-the-art data-driven scene synthesis [Fisher et al. 2012; Kermani et al. 2016] for the task of scene completion.

5.2 Experimental Results

Figure 1, Figure 7, and Figure 8 show randomly generated scenes using our approach. The overall quality matches that of the training data (the quantitative evaluation is presented in Section 5.3). The synthesized scenes are also diverse, exhibiting large variations in number of objects in a scene, spatial layouts, and correlated groups of objects. Figure 9 shows the closest scenes to some generated scenes from 1000 generated scenes. We use the Euclidean distance between the data matrix to find the closest pair. To better analyze the diversity of object arrangement in generated scenes, we ignore the object descriptor when finding the closest scenes. Note that, within 1000 generated scenes, the closest pair for some randomly selected scenes are not the same. Although the closest scenes are similar, there are always some changes in terms of object arrangements. To fit a smooth manifold of training data, we expect neighboring generated scenes to have similar but slightly different object arrangements. The result shows that our model does not show typical overfitting issues of generative models, which generate the same instances for different input noise vectors, and for neighboring generated scenes, it shows diversity in terms of object arrangements.

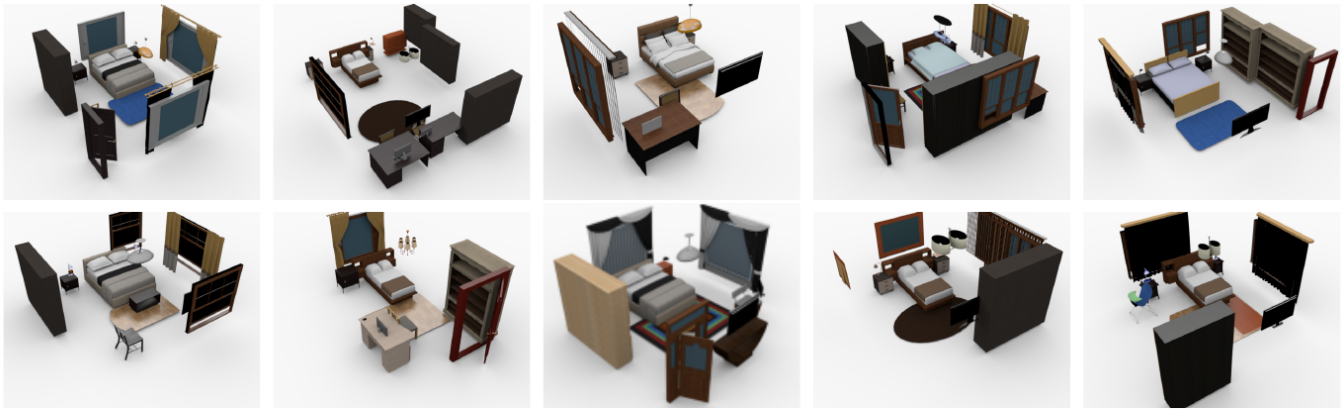

Fig. 7. Randomly generated scenes of bedrooms.

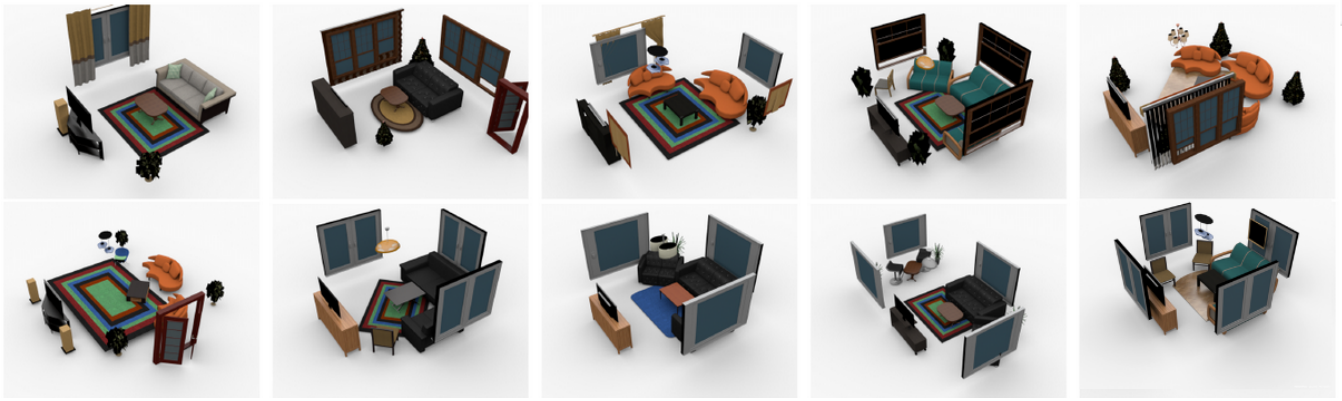

Fig. 8. Randomly generated scenes of living rooms.

Figure 1 compares the generated scenes with their closest scenes in the training data. Here we simply employ the Euclidean distance in the latent scene space for computing closest scenes. We can see that the generated scenes exhibit noticeable variations in spatial object layout and object existence. This means that our approach learns meaningful patterns in the training data and uses them for scene synthesis, instead of merely memorizing the data.

The training loss convergence curve has been shown and analyzed in Appendix A.

5.3 Perceptual Study

We have performed a user study to evaluate the visual quality of our approach versus baseline approaches by following the protocol described in [Shrivastava et al. 2016]. Specifically, for each approach and each scene type we generate 20 scenes. For each scene, we extract the closest scene in the training data. We then present these 20 pairs to users and ask them to choose the scene which they think is generated. Each study is summarized using a pair of percentages $(a, 100 - a)$, where a indicates the percentage that scenes in the synthetic data are marked as generated.

Figure 10 plots the statistics among our approach and the four baseline approaches. We can see that for both the Bedroom and Living Room datasets, our approach yields significantly better results than baseline approaches. In other words, the design choices of using sparsely connected layers and image-based discriminator loss are important for learning to generate realistic scenes. In addition, our approach achieved 61.4%/38.6% and 55.6%/44.4% on Bedroom and Living Room, respectively. Given that the training data mostly consists of high-quality user designed scenes, these numbers are quite encouraging, as more than 30% of the time, users favored our synthesis results rather than user designed scenes. The study was conducted on Amazon Mechanical Turk, and for each approach, we showed 20 different comparisons in the survey and collected 10 surveys from different users.

5.4 What are Learned

In this section, we analyze the performance of our approach by studying what are learned by the neural network. Our protocol is to assess whether important distributions about objects and object pairs in the training data are learned by the network, i.e., if the generated scenes have a distribution similar to the training data.

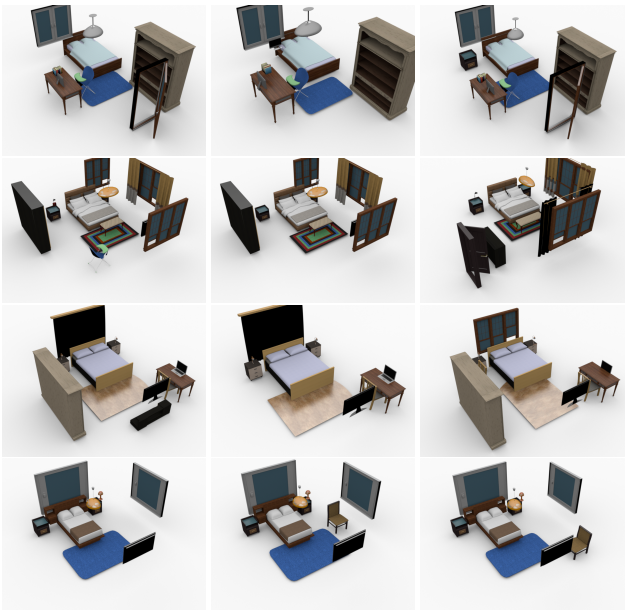

Fig. 9. Visual comparisons between synthesized scenes. For each row, the first scene is randomly selected from 1000 generated scenes and the second and third scenes are the **closest** scenes to the first in 1000 generated scenes. We use the same object shape and texture for better comparison.

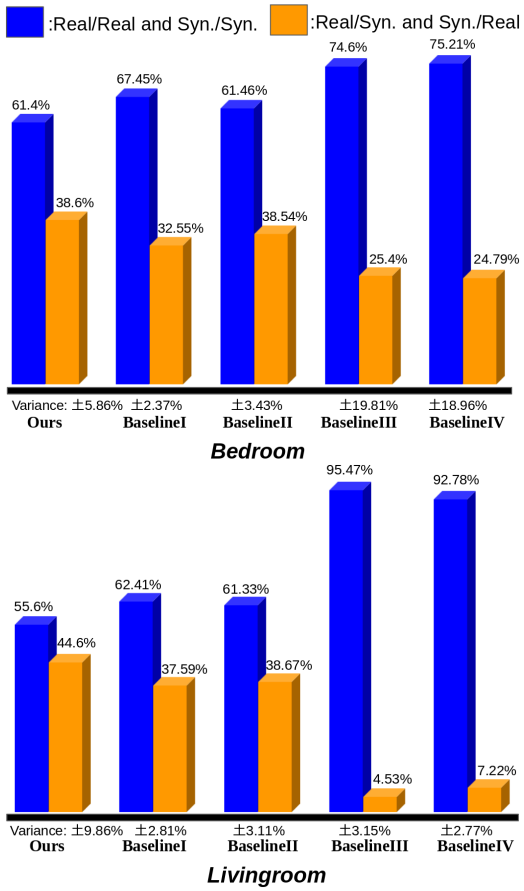

Fig. 10. User study on generated scenes using our approach and two baseline approaches on the Bedroom and Living Room datasets. Blue bar indicates the percentage of Real/Synthetic pairs that are marked as Real/Synthetic, where users obtain correct answers. Likewise, Orange bar indicates the percentage of Real/Synthetic pairs that are marked as Synthetic/Real, where users obtain wrong answers. For blue bar, the higher the better. For orange bar, the lower the better.

Pairwise correlations of objects. We first evaluate whether important pairwise distributions between objects are learned properly by our generator. We plot the distributions of the relative location between the second object and the first object. For simplicity, we only plot the marginal distribution on the x-y plane (or the top view), which captures most of the signals. Here the relative location is evaluated with respect to the coordinate system, whose origin is given by the point on the boundary of the bounding box, and whose direction to the bounding box center aligns with the front orientation. We plot the heat-map of the distributions. In this experiment, we consider Desk/Chair, Bed/Nightstand, Bed/Television, and Chair/Computer for Bedroom, and Sofa/Table, Table/Television, Plant/Sofa, and Sofa/Television for Living Room. If there are multiple pairs on one scene, we only extract the pairs with closest spatial distance.

We collect statistics from 7000 training scenes for bedroom, 6000 training scenes for living room, and from 5000 randomly generated scenes for each room type.

As illustrated in Figure 12 and 13, our generator nicely matches the distributions in the training data. With convolution layers and fully connected layers, the generator does not learn the pairwise pattern. An intuition is that the sparse layers explicitly force the network to learn features between grouped features of different objects, which helps to identify low-order interactions among the current feature representations, e.g., spatial correlations, while the dense layers or convolution layers only aggregate detected patterns globally. With the scene discriminator, the generator is able to learn better pairwise pattern between Chair/Computer for Bedroom and Plant/Sofa for Living Room compared to the generator trained without it. With the image discriminator, some pairwise pattern between objects has been smoothed and moved closer to the pairwise distribution in the training data, such as Bed/Night in Bedroom and Plant/Sofa in Living Room. In particular, for Desk/Chair and Bed/Nightstand, the learned distribution and the original distribution are very similar to each other. In other words, our approach learns better pairwise relations in the training data.

Figure 14 and Figure 15 show the distribution of the relative angles between the front orientations. We have quantized the generated angles, range from 0 to 2π, into 4 bins. The bottom of the circle corresponds to the case where the two object shares the same orientation. With convolution layers and fully connected layers, the generator again does not learn the pairwise pattern, and compared to other baseline approaches, our method learns such distributions slightly better.

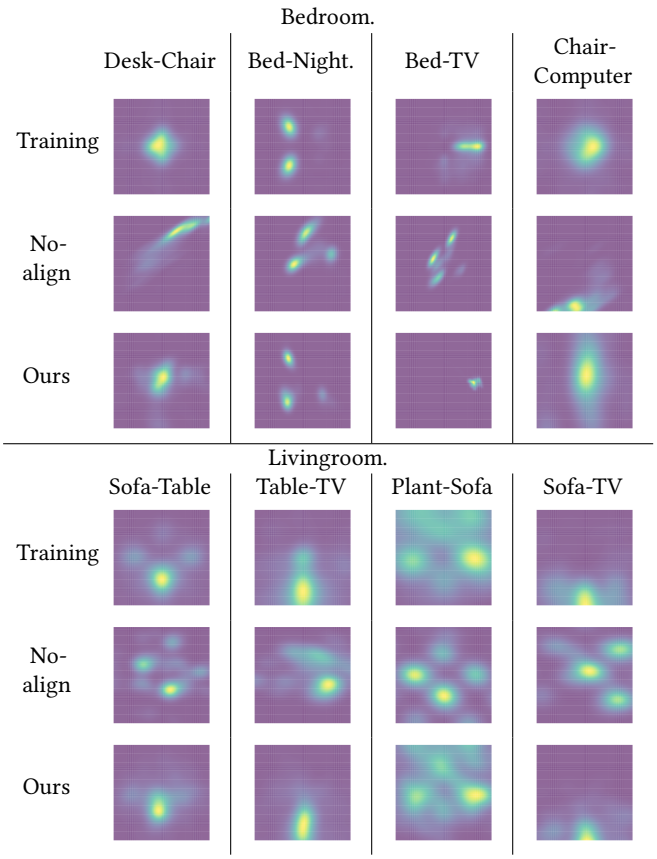

Fig. 11. Distributions of relative positions between selected object pairs. The origin lies in the image center. Top three rows: Distributions of selected pairs of classes in Bedroom for training data, synthesized scenes with no global scene alignment, and our synthesized scenes respectively (from left to right: Desk/Chair, Bed/Nightstand, Bed/Television, and Chair/Computer). Bottom three rows: Distributions of selected pairs of classes in Living Room for training data, synthesized scenes with no global scene alignment, and our synthesized scenes (from left to right: Sofa/Table, Table/Television, Plant/Sofa, and Sofa/Television).

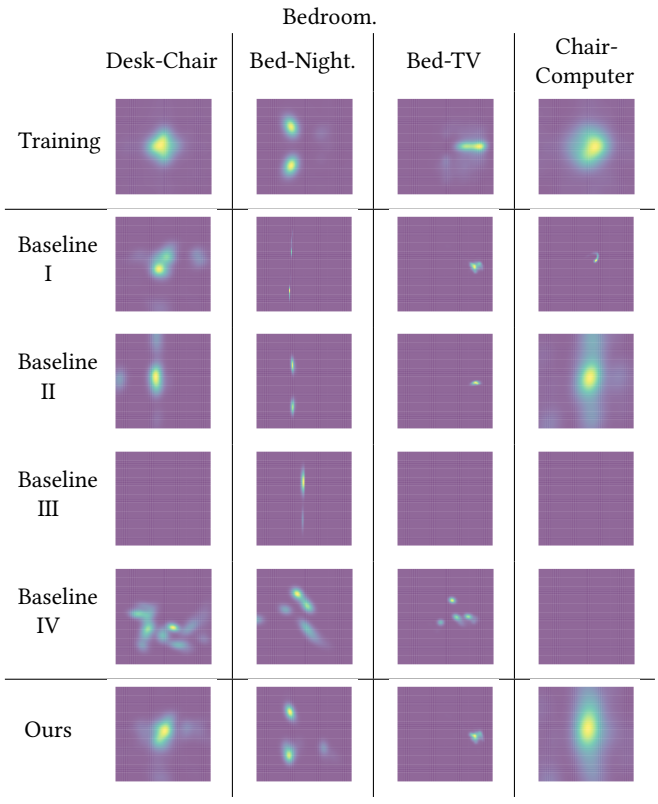

Fig. 12. Distributions of relative positions between selected object pairs. The origin lies in the image center. The position of the first object is in the center. The origin is at the center of the figure; the x-axis goes left; the y-axis goes up. Top to bottom: distributions of selected pairs of classes in Bedroom for training data, synthesized scenes with Baseline I, Baseline II, Baseline III, Baseline IV, and our synthesized scenes respectively (from left to right: Desk/Chair, Bed/Nightstand, Bed/Television, and Chair/Computer).

5.5 The Importance of Joint Scene Alignment

In this section, we perform an additional study to justify that jointly optimizing 3D scenes as a preprocessing step is important. As an evaluation metric, we use the distribution between selected pairs of objects between the Chair class and Table class and the Bed class and the NightStand class locations and orientation on the Bedroom dataset.

Global scene alignment. As illustrated in Figure 11, deactivating the global scene alignment step (i.e., applying our alternating minimization procedure on the raw input data directly) causes the network to not learn correlations between salient patterns. The distributions of absolute locations on generated scenes are significantly different from that on the training data. This justifies that global scene alignment is crucial to the success of our system. In other words, it is insufficient for local formulation to align the input scenes.

We have also evaluated whether our approach learns important distributions of the absolute locations of the objects in Appendix A.

5.6 Applications in Scene Interpolation

In this section, we show the application of our approach in scene interpolation. Given two input scenes, we first compute their associated latent parameters. We then interpolate these two latent parameters along the straight line between the two parameters and use the generator to generate the corresponding synthetic scenes. Figure 16 shows four examples. The first two examples are interpolations of bedroom scenes, and the second two are from living room scenes. For bedroom scenes, the first example consists of two scenes with very different configurations, where the orientation of two rooms are not aligned. The intermediate scenes gradually remove the original bed, table and desk, and then add the bed with new location and orientation, which is semantically meaningful. The second example consists of two similar bedroom scenes with

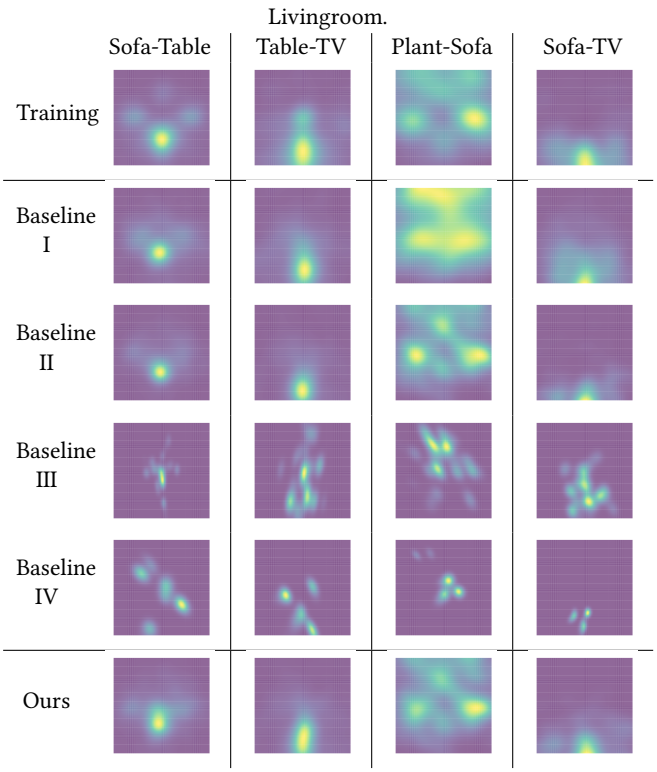

Fig. 13. Distributions of relative positions between selected object pairs. The origin lies in the image center. The position of the first object is in the center. The origin is at the center of the figure; the x-axis goes left; the y-axis goes up. Top to bottom: distributions of selected pairs of classes in Living Room for training data, synthesized scenes with Baseline I, Baseline II, Baseline III, Baseline IV, and our synthesized scenes (from left to right: Sofa/Table, Table/Television, Plant/Sofa, and Sofa/Television).

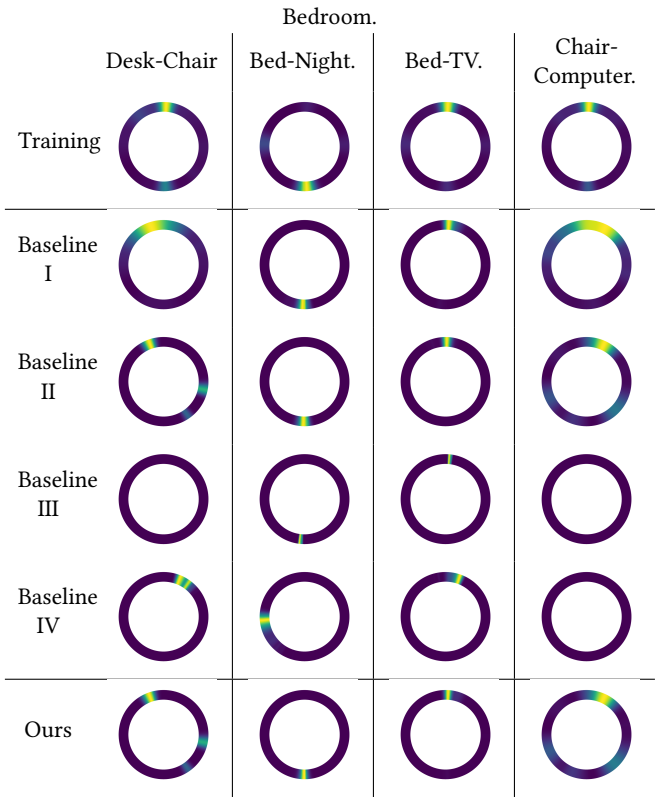

Fig. 14. Distributions of relative orientations between selected object pairs. The bottom of the circle represents the case where the two objects share the same orientation, the left or right represents the case where the two objects are orthogonal to each other, and the top represents the case where the two objects are facing opposite directions. In other words, the bottom of the circle shows zero difference, and the difference of orientations increases counter-clockwise, where the difference at the right of the circle is 90, the difference at the top of the circle is 180, etc. Yellow means high density while blue means low density. Top to bottom: distributions of selected pairs of classes in Bedroom for training data, synthesized scenes with Baseline I, Baseline II, Baseline III, Baseline IV, and our synthesized scenes respectively (from left to right: Desk/Chair, Bed/Nightstand, Bed/Television, and Chair/Computer).

different objects. In intermediate scenes, bed, night stands and table lamps stay the same, while wardrobes are gradually removed and the a table and a laptop are added gradually. These are meaningful interpolations. For living room scenes, the first example consists of two scenes with similar configuration of objects but the locations of the chairs are reversed with respect to the sofa object. The intermediate scenes gradually remove objects on one side and then add objects on the other side, leading to a meaningful interpolation. The second example shows a configuration where the source scene has different objects than the target scene. The intermediate scenes progressively delete objects and then add new objects in different category, which is again semantically meaningful.

5.7 Applications in Scene Completion

We then show the application of our approach on the application of scene completion. In this task, we are given a partial scene, and our task is to find the optimal scene that completes the input scene. Towards this end, we solve the following optimization problem:

$$\mathbf{z}^{\star} = \underset{\mathbf{z},\mathcal{T},\mathcal{S}}{\operatorname{argmin}} \|\mathcal{T}\mathcal{S}(M_{in}) - C.G_{\theta}(\mathbf{z})\|_F^2 + \alpha\|\mathbf{z}\|^2 \qquad (10)$$

where C is the mask associated with M_{in}, and it constraints that the completed scene should contain objects in the input partial scene. . is the elementwise matrix product. Again we apply gradient descent for optimization. We set $\alpha = 1e-3$ in our experiments. Intuitively, our approach seeks to find among all plausible scenes that match the partial observations, the one that is the most popular, e.g., close to the origin.

As a comparison, we compare the synthesis results of [Fisher et al. 2012], [Kermani et al. 2016] and [Wang et al. 2018a]. Since all approaches used different datasets, as for a fair comparison, we re-implemented their approaches on our Bedroom and Living Room datasets.

Figure 17 compares our approach with baseline approaches. The input partial scenes are cropped from scenes in the testing datasets. Since all baseline approaches generate a series of completed scenes,

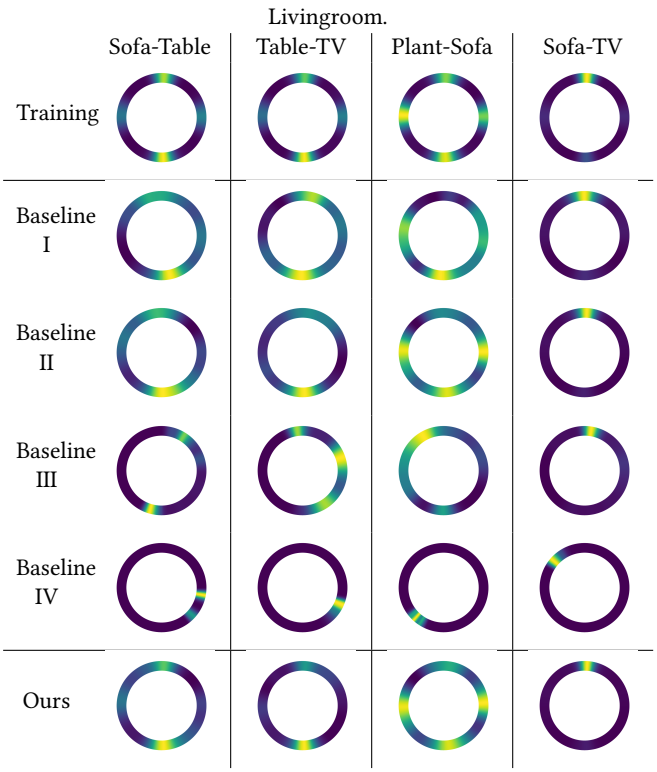

Fig. 15. Distributions of relative orientations between selected object pairs. The bottom of the circle represents the case where the two objects share the same orientation, the left or right represents the case where the two objects are orthogonal to each other, and the top represents the case where the two objects are facing opposite directions. In other words, the bottom of the circle shows zero difference, and the difference of orientations increases counter-clockwise, where the difference at the right of the circle is 90, the difference at the top of the circle is 180, etc. Yellow means high density while blue means low density. Top to bottom: distributions of selected pairs of classes in Living Room for training data, synthesized scenes with Baseline I, Baseline II, Baseline III, Baseline IV, and our synthesized scenes (from left to right: Sofa/Table, Table/Television, Plant/Sofa, and Sofa/Television).

for a fair comparison we choose the ones that have the same or similar number of objects as the output of our approach. We can see that our approach leads to semantically more meaningful results in terms of both groups of co-related objects and locally compatible of object pairs. We can understand this as the fact that our approach optimizes the scene layout with respect to all patterns captured by the neural network. In contrast, other baseline approaches are sequential, despite the usage of local optimization [Yu et al. 2011] to improve scene layouts, they may not be able to explore the entire underlying scene space for generating the completed scenes.

Moreover, our approach is significantly faster than the baseline approaches. Our approach takes 1-2 seconds for solving (10) , while [Fisher et al. 2012], [Kermani et al. 2016] and [Wang et al. 2018a] take 83.1 seconds, 76.4 seconds and 10 minutes in average, respectively. In particular, most of the computational time was spent on running local optimization to improve the scene layouts.

6 CONCLUSIONS

We have studied the problem of 3D scene synthesis using deep generative models. Unlike 2D images, 3D geometries possess multiple varying representations, each with its advantages and disadvantages for the most efficacious deep neural networks. To maximize tradeoffs, we therefore presented a hybrid methodology that trains a 3D scene generator using a combination of a 3D object arrangement representation, and a projected 2D image representation, combining the advantages of both representations. The 3D object arrangement representation ensures local and global neighborhood structure of the synthesized scene, while image-based representations preserve local view-dependent patterns. Moreover the results obtained from the image-based representation is beneficial for training the 3D generator.

Our 3D scene generator is a feed-forward neural network. This network design takes another route from the common recurrent methodology for 3D scene synthesis and modeling. The benefit of the feed-forward architecture is that it can jointly optimize all the factors for 3D synthesis, while it is difficult for a recurrent approach to recover from mistakes made during its sequential processing. Preliminary qualitative evaluations have shown the advantage of the feed-forward architecture over two recurrent approaches. Although it is premature to say that feed-forward approaches shall significantly dominate recurrent approaches, we do believe that free-forward networks have shown great promise in several scenarios, and deserves further research and exploitation.

One limitation of our approach is that we do not completely encode physical properties of the synthesized scenes, which are important for computer aided manufacturing purposes (e.g., 3D printing). To address this issue, one possibility is to develop a suitable 3D representation that explicitly encodes physical properties, e.g., using a shape grammar.

Another limitation of our approach is that all the training data should consist of semantically segmented 3D scenes. This may not always be possible, e.g., reconstructed 3D scenes from point clouds are typically not segmented into individual objects. A potential way to address this issue is to extend the consistent hybrid representation described in this paper, e.g., by enforcing the consistency among three networks: 1) scene synthesis under the image-based representation, 2) scene synthesis under the 3D object arrangement representation and 3) a network that converts a 3D scene into its corresponding 3D object arrangement representation.

Besides, our approach restricts the number of synthesized instances in generated scenes. This strategy may raise issues for object arrangements in large scenes. A potential solution is to extend our model so that it can synthesize new object arrangements for an existing scene layout. For large scenes, we can feed the generated scenes back to acquire more synthesized object arrangements. We leave this as future research.

Our approach uses shape descriptors to query appropriate shapes in the database to determine the shapes of the objects in each synthesized scene. A limitation of this approach is that at the object level, we do not create new shapes. In the future, we plan to address this issue by learning a shape generator for each category and

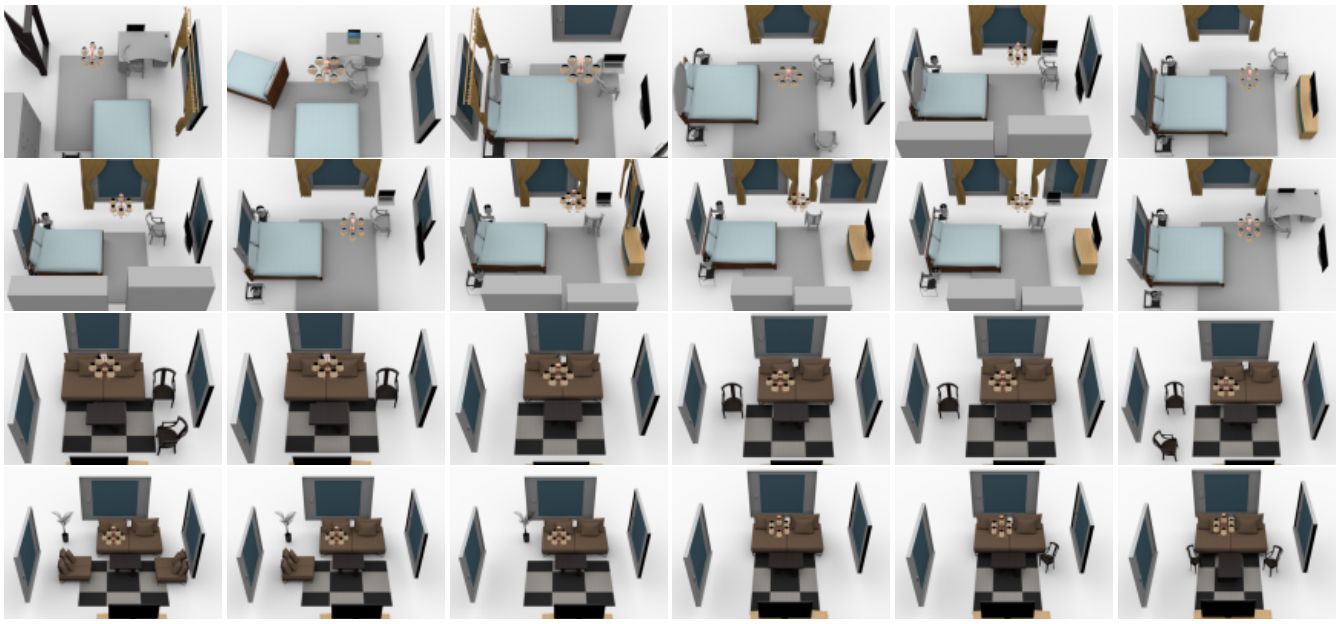

Fig. 16. Scene interpolation results between different pairs of source (left column) and target (right column) scenes.

Fig. 17. Scene completion results. From top to bottom, we show the input objects, completed scenes generated by our method, and the results using [Fisher et al. 2012], [Kermani et al. 2016] and [Wang et al. 2018a], respectively.

modifying our scene generator to synthesize the latent parameter associated with each scene object.

There are multiple other directions and opportunities for future research. As mentioned in the introduction, there are at least five frequently used 3D representations. One could extend our current approach to use more than one 3D representation. For example, we could leverage multi-view representations on which we have rich training data (e.g., internet images). The multi-view representation also provides texture information, useful for synthesizing 3D representations. Finally, we would propose to combine the learned 3D representation with data from other modalities such as natural language descriptions.

Acknowledgement. The authors would like thank Chandrajit Bajaj for many fruitful discussions. Qixing Huang would like to acknowledge support of this research from NSF DMS-1700234, a Gift from Snap Research, and a hardware Donation from NVIDIA.

REFERENCES

Brett Allen, Brian Curless, and Zoran Popović. 2003. The Space of Human Body Shapes: Reconstruction and Parameterization from Range Scans. *ACM Trans. Graph.* 22, 3 (July 2003), 587–594. https://doi.org/10.1145/882262.882311

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: Shape Completion and Animation of People. *ACM Trans. Graph.* 24, 3 (July 2005), 408–416. https://doi.org/10.1145/1073204.1073207

Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *ICML (Proceedings of Machine Learning Research)*, Vol. 70. PMLR, Sydney, Australia, 214–223.

Volker Blanz and Thomas Vetter. 1999. A Morphable Model for the Synthesis of 3D Faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 187–194. https://doi.org/10.1145/311535.311556

Zhangjie Cao, Qixing Huang, and Karthik Ramani. 2017. 3D Object Classification via Spherical Projections. In *3DV*. IEEE Computer Society, Qingdao, Shandong, China, 566–574.

Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. *CoRR* abs/1512.03012 (2015). http://arxiv.org/abs/1512.03012

Avishek Chatterjee and Venu Madhav Govindu. 2013. Efficient and Robust Large-Scale Rotation Averaging. In *ICCV*. IEEE Computer Society, Sydney, Australia, 521–528.

Siddhartha Chaudhuri, Evangelos Kalogerakis, Stephen Giguere, and Thomas Funkhouser. 2013. Attribit: Content Creation with Semantic Attributes. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 193–202. https://doi.org/10.1145/2501988.2502008

Siddhartha Chaudhuri, Evangelos Kalogerakis, Leonidas Guibas, and Vladlen Koltun. 2011. Probabilistic Reasoning for Assembly-based 3D Modeling. *ACM Trans. Graph.* 30, 4, Article 35 (July 2011), 10 pages.

Siddhartha Chaudhuri and Vladlen Koltun. 2010. Data-driven Suggestions for Creativity Support in 3D Modeling. *ACM Trans. Graph.* 29, 6, Article 183 (Dec. 2010), 10 pages. https://doi.org/10.1145/1882261.1866205

Kang Chen, Yu-Kun Lai, Yu-Xin Wu, Ralph Martin, and Shi-Min Hu. 2014. Automatic Semantic Modeling of Indoor Scenes from Low-quality RGB-D Data Using Contextual Information. *ACM Trans. Graph.* 33, 6, Article 208 (Nov. 2014), 12 pages. https://doi.org/10.1145/2661229.2661239

Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. 2018. Spherical CNNs. *CoRR* abs/1801.10130 (2018).

Ingrid Daubechies, Ronald Devore, Massimo Fornasier, and C. Sinan GÃijntÃijrk. 2010. Iteratively reweighted least squares minimization for sparse recovery. *Comm. Pure Appl. Math* 63 (January 2010), 1–38. Issue 1.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*. IEEE Computer Society, 248–255.

Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. 2017. 3D object classification and retrieval with Spherical CNNs. *CoRR* abs/1711.06721 (2017).

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. 2017. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *CVPR*. IEEE Computer Society, Honolulu, Hawaii,USA, 2463–2471.

Matthew Fisher and Pat Hanrahan. 2010. Context-based Search for 3D Models. *ACM Trans. Graph.* 29, 6, Article 182 (Dec. 2010), 10 pages. https://doi.org/10.1145/1882261.1866204

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. 2012. Example-based Synthesis of 3D Object Arrangements. *ACM Trans. Graph.* 31, 6, Article 135 (Nov. 2012), 11 pages.

Matthew Fisher, Manolis Savva, and Pat Hanrahan. 2011. Characterizing Structural Relationships in Scenes Using Graph Kernels. *ACM Trans. Graph.* 30, 4, Article 34 (July 2011), 12 pages. https://doi.org/10.1145/2010324.1964929

Matthew Fisher, Manolis Savva, Yangyan Li, Pat Hanrahan, and Matthias Niessner. 2015. Activity-centric Scene Synthesis for Functional 3D Scene Modeling. *ACM Trans. Graph.* 34, 6, Article 179 (Oct. 2015), 13 pages. https://doi.org/10.1145/2816795.2818057

Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. 2004. Modeling by Example. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 652–663. https://doi.org/10.1145/1015706.1015775

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

David Ha and Douglas Eck. 2017. A Neural Representation of Sketch Drawings. *CoRR* abs/1704.03477 (2017).

Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *CoRR* abs/1510.00149 (2015).

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Shijian Tang, Erich Elsen, Bryan Catanzaro, John Tran, and William J. Dally. 2016. DSD: Regularizing Deep Neural Networks with Dense-Sparse-Dense Training Flow. *CoRR* abs/1607.04381 (2016).

Christian Häne, Shubham Tulsiani, and Jitendra Malik. 2017. Hierarchical Surface Prediction for 3D Object Reconstruction. *CoRR* abs/1704.00710 (2017).

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 770–778.

Kyle Heath, Natasha Gelfand, Maks Ovsjanikov, Mridul Aanjaneya, and Leonidas J. Guibas. 2010. Image webs: Computing and exploiting connectivity in image collections.. In *CVPR*. IEEE Computer Society, 3432–3439. http://dblp.uni-trier.de/db/conf/cvpr/cvpr2010.html#HeathGOAG10

Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. *CoRR* abs/1506.05163 (2015).

Berthold K. P. Horn. 1987. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A* 4, 4 (1987), 629–642.

Qixing Huang, Hai Wang, and Vladlen Koltun. 2015. Single-view Reconstruction via Joint Analysis of Image and Shape Collections. *ACM Trans. Graph.* 34, 4, Article 87 (July 2015), 10 pages.

Qi-Xing Huang, Simon Flöry, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. 2006. Reassembling Fractured Objects by Geometric Matching. *ACM Trans. Graph.* 25, 3 (July 2006), 569–578. https://doi.org/10.1145/1141911.1141925

Qi-Xing Huang and Leonidas Guibas. 2013. Consistent Shape Maps via Semidefinite Programming. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing (SGP '13)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 177–186. https://doi.org/10.1111/cgf.12184

Xiangru Huang, Zhenxiao Liang, Chandrajit Bajaj, and Qixing Huang. 2017. Translation Synchronization via Truncated Least Squares. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 1459–1468. http://papers.nips.cc/paper/6744-translation-synchronization-via-truncated-least-squares.pdf

Daniel Huber. 2002. *Automatic Three-dimensional Modeling from Reality*. Ph.D. Dissertation. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Hamid Izadinia, Qi Shan, and Steven M. Seitz. 2016. IM2CAD. *CoRR* abs/1608.05137 (2016).

Yun Jiang, Marcus Lim, and Ashutosh Saxena. 2012. Learning Object Arrangements in 3D Scenes using Human Context. In *ICML*. icml.cc / Omnipress.

Evangelos Kalogerakis, Siddhartha Chaudhuri, Daphne Koller, and Vladlen Koltun. 2012. A Probabilistic Model for Component-based Shape Synthesis. *ACM Trans. Graph.* 31, 4, Article 55 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185551

Z. Sadeghipour Kermani, Z. Liao, P. Tan, and H. Zhang. 2016. Learning 3D Scene Synthesis from Annotated RGB-D Images. *Comput. Graph. Forum* 35, 5 (Aug. 2016), 197–206. https://doi.org/10.1111/cgf.12976

Vladimir G. Kim, Wilmot Li, Niloy J. Mitra, Stephen DiVerdi, and Thomas Funkhouser. 2012. Exploring Collections of 3D Models Using Fuzzy Correspondences. *ACM Trans. Graph.* 31, 4, Article 54 (July 2012), 11 pages. https://doi.org/10.1145/2185520.2185550

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). http://dblp.uni-trier.de/db/journals/corr/corr1412.html#KingmaB14

Diederik P. Kingma, Tim Salimans, and Max Welling. 2016. Improving Variational Inference with Inverse Autoregressive Flow. *CoRR* abs/1606.04934 (2016). http://arxiv.org/abs/1606.04934

Diederik P. Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. *CoRR* abs/1312.6114 (2013). http://arxiv.org/abs/1312.6114

Roman Klokov and Victor S. Lempitsky. 2017. Escape from Cells: Deep Kd-Networks for The Recognition of 3D Point Cloud Models. *CoRR* abs/1704.01222 (2017).

Vladislav Kreavoy, Dan Julius, and Alla Sheffer. 2007. Model Composition from Interchangeable Components. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications (PG '07)*. IEEE Computer Society, Washington, DC, USA, 129–138. https://doi.org/10.1109/PG.2007.43

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. 2016. Autoencoding Beyond Pixels Using a Learned Similarity Metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org, 1558–1566. http://dl.acm.org/citation.cfm?id=3045390.3045555

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. GRASS: Generative Recursive Autoencoders for Shape Structures. *ACM Trans. Graph.* 36, 4, Article 52 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073637

Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. 2019. Grains: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)* 38, 2 (2019), 12.

Tianqiang Liu, Siddhartha Chaudhuri, Vladimir G. Kim, Qixing Huang, Niloy J. Mitra, and Thomas Funkhouser. 2014. Creating Consistent Scene Graphs Using a Probabilistic Grammar. *ACM Trans. Graph.* 33, 6, Article 211 (Nov. 2014), 12 pages. https://doi.org/10.1145/2661229.2661243

Rui Ma, Honghua Li, Changqing Zou, Zicheng Liao, Xin Tong, and Hao Zhang. 2016. Action-driven 3D Indoor Scene Evolution. *ACM Trans. Graph.* 35, 6, Article 173 (Nov. 2016), 13 pages. https://doi.org/10.1145/2980179.2980223

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

Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. 2015. Geodesic Convolutional Neural Networks on Riemannian Manifolds. In *Proceedings of the 2015 IEEE International Conference on Computer Vision Workshop (ICCVW) (ICCVW '15)*. IEEE Computer Society, Washington, DC, USA, 832–840. https://doi.org/10.1109/ICCVW.2015.112

Paul Merrell, Eric Schkufza, and Vladlen Koltun. 2010. Computer-generated Residential Building Layouts. *ACM Trans. Graph.* 29, 6, Article 181 (Dec. 2010), 12 pages. https://doi.org/10.1145/1882261.1866203

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Jan Svoboda, and Michael M. Bronstein. 2017. Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs. In *CVPR*. IEEE Computer Society, 5425–5434.

C. Nash and C. K. I. Williams. 2017. The Shape Variational Autoencoder: A Deep Generative Model of Part-segmented 3D Objects. *Comput. Graph. Forum* 36, 5 (Aug. 2017), 1–12. https://doi.org/10.1111/cgf.13240

Planner5d. 2017. Home Design Software and Interior Design Tool ONLINE for home and floor plans in 2D and 3D. https://planner5d.com

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. 2016. Exponential expressivity in deep neural networks through transient chaos. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 3360–3368. http://papers.nips.cc/paper/6322-exponential-expressivity-in-deep-neural-networks-through-transient-chaos.pdf

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*. IEEE Computer Society, 77–85.

Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. 2016. Volumetric and Multi-view CNNs for Object Classification on 3D Data. In *CVPR*. IEEE Computer Society, 5648–5656.

Siyuan Qi, Yixin Zhu, Siyuan Huang, Chenfanfu Jiang, and Song-Chun Zhu. 2018. Human-centric Indoor Scene Synthesis Using Stochastic Grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Salt Lake City, UT, USA, 5899–5908.

Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR* abs/1511.06434 (2015).

Gernot Riegler, Ali Osman Ulusoy, Horst Bischof, and Andreas Geiger. 2017. OctNetFusion: Learning Depth Fusion from Data. *CoRR* abs/1704.01047 (2017).

Daniel Ritchie, Anna Thomas, Pat Hanrahan, and Noah D. Goodman. 2016. Neurally-guided Procedural Models: Amortized Inference for Procedural Graphics Programs Using Neural Networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*. Curran Associates Inc., USA, 622–630. http://dl.acm.org/citation.cfm?id=3157096.3157166

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

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved Techniques for Training GANs. *CoRR* abs/1606.03498 (2016).

Manolis Savva, Angel X. Chang, Pat Hanrahan, Matthew Fisher, and Matthias Niessner. 2016. PiGraphs: Learning Interaction Snapshots from Observations. *ACM Trans. Graph.* 35, 4, Article 139 (July 2016), 12 pages. https://doi.org/10.1145/2897824.2925867

Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. 2012. An interactive approach to semantic modeling of indoor scenes with an RGBD camera. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 136.

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. 2017. CSGNet: Neural Shape Parser for Constructive Solid Geometry. *CoRR* abs/1712.08290 (2017).

Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. 2012. Structure Recovery by Part Assembly. *ACM Trans. Graph.* 31, 6, Article 180 (Nov. 2012), 11 pages.

Yanyao Shen, Qixing Huang, Nati Srebro, and Sujay Sanghavi. 2016. Normalized Spectral Map Synchronization. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 4925–4933. http://papers.nips.cc/paper/6128-normalized-spectral-map-synchronization.pdf

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russell Webb. 2016. Learning from Simulated and Unsupervised Images through Adversarial Training. *CoRR* abs/1612.07828 (2016). arXiv:1612.07828 http://arxiv.org/abs/1612.07828

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. 2017. Semantic Scene Completion from a Single Depth Image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition* (2017).

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view Convolutional Neural Networks for 3D Shape Recognition. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (ICCV '15)*. IEEE Computer Society, Washington, DC, USA, 945–953. https://doi.org/10.1109/ICCV.2015.114

Minhyuk Sung, Hao Su, Vladimir G. Kim, Siddhartha Chaudhuri, and Leonidas Guibas. 2017. Complementme: Weakly-supervised Component Suggestions for 3D Modeling. *ACM Trans. Graph.* 36, 6, Article 226 (Nov. 2017), 12 pages. https://doi.org/10.1145/3130800.3130821

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. 2016. Multi-view 3D Models from Single Images with a Convolutional Network. In *ECCV (7) (Lecture Notes in Computer Science)*, Vol. 9911. Springer, 322–337.

Olivier Teboul. 2011. *Shape grammar parsing : application to image-based modeling*. Ph.D. Dissertation. Ecole Centrale Paris. https://tel.archives-ouvertes.fr/tel-00628906

Shubham Tulsiani, Saurabh Gupta, David F. Fouhey, Alexei A. Efros, and Jitendra Malik. 2017a. Factoring Shape, Pose, and Layout from the 2D Image of a 3D Scene. *CoRR* abs/1712.01812 (2017).

Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. 2017b. Learning Shape Abstractions by Assembling Volumetric Primitives. In *CVPR*. IEEE Computer Society, 1466–1474.

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel Recurrent Neural Networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org, New York, NY, USA, 1747–1756. http://dl.acm.org/citation.cfm?id=3045390.3045575

Hao Wang, Nadav Schor, Ruizhen Hu, Haibin Huang, Daniel Cohen-Or, and Hui Huang. 2018b. Global-to-local generative model for 3d shapes. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 214.

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

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

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis. *ACM Trans. Graph.* 36, 4, Article 72 (2017), 11 pages.

Jason Weber and Joseph Penn. 1995. Creation and Rendering of Realistic Trees. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, New York, NY, USA, 119–128. https://doi.org/10.1145/218380.218427

Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. 2016. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-adversarial Modeling. In *Proceedings of the 30th International Conference*

on Neural Information Processing Systems (NIPS'16). Curran Associates Inc., USA, 82–90. http://dl.acm.org/citation.cfm?id=3157096.3157106

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3D ShapeNets: A deep representation for volumetric shapes.. In *CVPR*. IEEE Computer Society, 1912–1920. http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#WuSKYZTX15

Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. 2013. Sketch2Scene: Sketch-based Co-retrieval and Co-placement of 3D Models. *ACM Transactions on Graphics* 32, 4 (2013), 123:1–123:12.

Kai Xu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2012. Fit and Diverse: Set Evolution for Inspiring 3D Shape Galleries. *ACM Trans. Graph.* 31, 4, Article 57 (July 2012), 10 pages.

Li Yi, Hao Su, Xingwen Guo, and Leonidas J. Guibas. 2016. SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation. *CoRR* abs/1612.00606 (2016).

Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. 2011. Make It Home: Automatic Optimization of Furniture Arrangement. *ACM Trans. Graph.* 30, 4, Article 86 (July 2011), 12 pages. https://doi.org/10.1145/2010324.1964981

Junbo Jake Zhao, Michaël Mathieu, and Yann LeCun. 2016. Energy-based Generative Adversarial Network. *CoRR* abs/1609.03126 (2016). http://arxiv.org/abs/1609.03126

Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Renjiao Yi, and Hao Zhang. 2018. Scores: Shape composition with recursive substructure priors. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 211.

Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. 2018. LayoutNet: Reconstructing the 3D Room Layout From a Single RGB Image. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, Salt Lake City, UT, USA, 2051–2059. https://doi.org/10.1109/CVPR.2018.00219

Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 2017. 3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, Venice, Italy, 900–909. https://doi.org/10.1109/ICCV.2017.103

A ADDITIONAL RESULTS AND LESSONS LEARNED

In this section, we show additional results and analysis.

Figure 20 shows the training loss convergence curve. Due to the nature of adversarial training, the loss term is oscillating during training. Thus, we do not include the adversarial loss for convergence analysis. Based on Figure 20, the training procedure is stable and converges smoothly. In addition, without optimizing the pair-wise permutation and translation during training, the loss converges slower.

Additional results have been shown in Figure 19 and Figure 21.

Absolute locations of objects. We evaluate whether our approach learns important distributions of the absolute locations of the objects. For this particular experiment, we picked 5000 training scenes with rectangle shape for both bedroom and living room types, and used it as the training instances for this experiment. With rectangle room geometry, the absolution location distribution can be shown with rectangle heatmap pictures, which serves better visualization purpose. To this end, we have tested the distributions of absolute locations of Nightstand, Bed, Window and Television for Bedroom, and Door, Window, Rug and Plant for Living Room. For each object, we calculate the distributions in the training data (with respect to the aligned scenes $\overline{M}_i$) versus 5000 randomly generated scenes.

For simplicity, we only plot the marginal distribution on the x-y plane (or the top view), which captures most of the signals.

As illustrated in Figure 18, the distributions between synthesized scenes of our approach and training scenes are fairly close. In particular, on Window, the difference between the distributions are not easy to identify. The two distributions are less similar on Plant. An explanation is that there are fewer instances of Plant in the training data than other categories, and thus the generalization behavior performs less well.

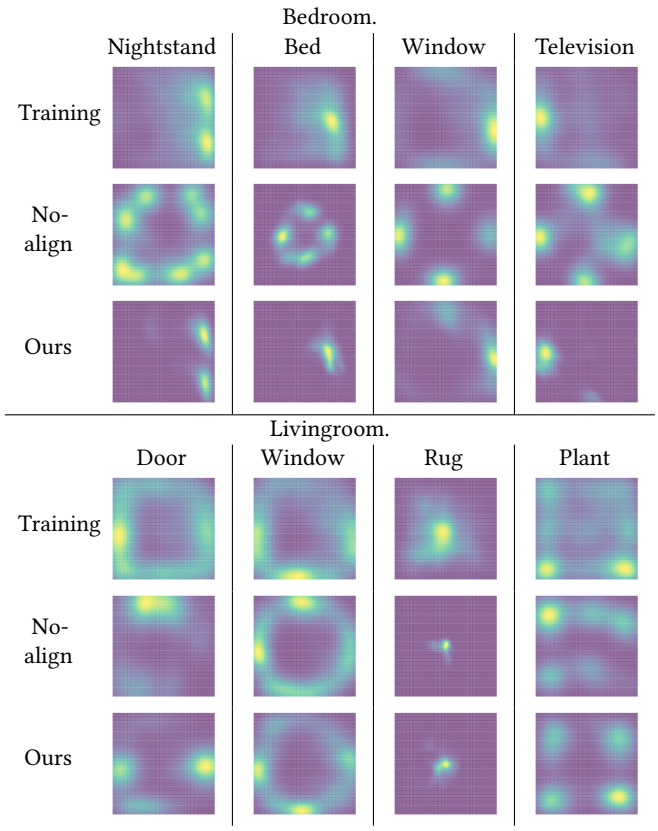

Fig. 18. Distributions of absolute locations of selected classes. Top three rows: Distributions of selected object categories in Bedroom for training data, synthesized scenes with no global scene alignment, and our synthesized scenes respectively (from left to right: Nightstand, Bed, Window, and Television). Bottom three rows: Distributions of selected classes in Livingroom for training data, synthesized scenes with no global scene alignment, and our synthesized scenes respectively (from left to right: Door, Window, Rug, Plant.)

B ADDITIONAL DETAILS ON PAIR-WISE SCENE ALIGNMENT

In this section, we present our numerical optimization approach for solving (5), which combines reweighted non-linear least squares and alternating minimization. To this end, we first introduce a weight vector $\mathbf{w}$ corresponding to the columns of $\mathcal{T}(\mathcal{S}(M_i)) - M_j$ and modify the optimization problem as

$$\mathcal{T}_{ij}^{in}, \mathcal{S}_{ij}^{in} = \underset{\mathcal{T},\mathcal{S}}{\operatorname{argmin}} \|(\mathcal{T}(\mathcal{S}(M_i)) - M_j)\operatorname{diag}(\mathbf{w})\|_F^2. \tag{11}$$

RLSM alternates between fixing $\mathbf{w}$ to solve (11) and using the optimal solution to update $\mathbf{w}$. We set the initial weight vector as $\mathbf{w}^{(0)} = \mathbf{1}$.

Given the weight vector $\mathbf{w}^{(t)}$ at iteration t, we again perform alternating minimization to optimize $\mathcal{T}$ and $\mathcal{S}$. At each inner iteration

Fig. 19. Additional randomly generated scenes of bedrooms.

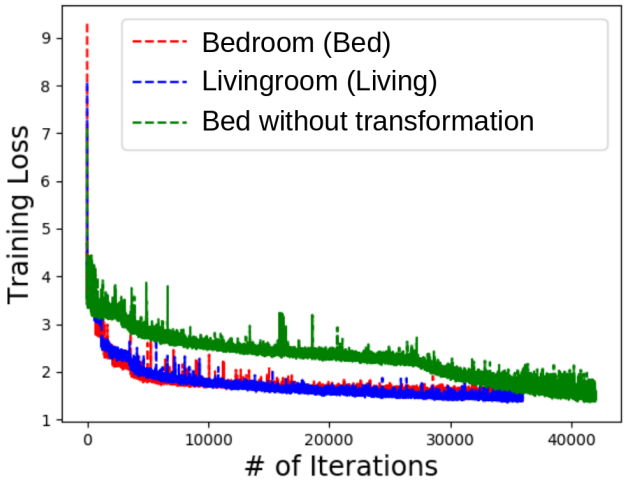

Fig. 20. Training loss without the adversarial loss versus # of iterations for bedroom and living room. Bed without transformation shows the convergence curve without optimizing pairwise permutation and translation during training.

s, the updates are given by

$$\mathcal{T}^{(t,s)} = \underset{\mathcal{T}}{\operatorname{argmin}} \|(\mathcal{T}(\mathcal{S}^{(t,s-1)}(M_i)) - M_j)\operatorname{diag}(\mathbf{w}^{(t)})\|_F^2, \quad (12)$$

$$\mathcal{S}^{(t,s)} = \underset{\mathcal{S}}{\operatorname{argmin}} \|(\mathcal{T}^{(t,s)}(\mathcal{S}(M_i)) - M_j)\operatorname{diag}(\mathbf{w}^{(t)})\|_F^2. \quad (13)$$

In this case, both (12) and (13) admit closed-form solutions. The optimal solution of (12) can be computed using [Horn 1987]. The optimal solution of (13) can be computed by solving a linear assignment.

This alternate minimization procedure converges fairly fast, we apply 4 iterations in our implementation.

Given the solution $\mathcal{T}^{(t)}$ and $\mathcal{S}^{(t)}$ from the alternating minimization procedure described above, we update the weight vector at iteration $t+1$ as

$$w_k^{(t+1)} = \epsilon / \sqrt{\epsilon^2 + \|(\mathcal{T}^{(t)}\mathcal{S}^{(t)}(M_i) - M_j)\mathbf{e}_k\|^2}, \quad 1 \le k \le K,$$

where $\mathbf{e}_k$ is the k-th canonical basis of $\mathbb{R}^K$. $\epsilon = 10^{-3}$ is chosen to be a small value. In our implementation, we apply 4 iterations of reweighted least squares.

C GRADIENT OF THE IMAGE PROJECTION

Since $\mathcal{P}(M)$ is an image, and the pixel values are summation of signed distance function values. In addition, the signed distance function is with respect to a oriented box. Thus, it is sufficient to derive the formula for computing the gradient of a point $\mathbf{p}$ with respect to a line l parameterized by an orientation $\mathbf{n}$ and a point

Fig. 21. Additional randomly generated scenes of living rooms.

$\mathbf{q} = \mathbf{o} + s\mathbf{n}$ on l:

$$\begin{aligned} d(\mathbf{p}, l) &:= (\mathbf{p} - \mathbf{q})^T \mathbf{n} \\ &= (\mathbf{p} - \mathbf{o} - s\mathbf{n})^T \mathbf{n} \\ &= (\mathbf{p} - \mathbf{o})^T \mathbf{n} - s. \end{aligned} \tag{14}$$

Here $\mathbf{o}$ represents the center of the box, $\mathbf{n}$ is the axis that is perpendicular to the line, and s is the size along this axis.

It is easy to see that the derivative of $d(\mathbf{p}, l)$ with respect to $\mathbf{o}$, $\mathbf{n}$ and s are given by

$$\frac{\partial d(\mathbf{p}, l)}{\partial \mathbf{o}} = -\mathbf{n}, \quad \frac{\partial d(\mathbf{p}, l)}{\partial \mathbf{n}} = \left((\mathbf{p} - \mathbf{o})^T \mathbf{n}^\perp\right) \cdot \mathbf{n}^\perp, \quad \frac{\partial d(\mathbf{p}, l)}{\partial s} = -1. \tag{15}$$

Here $\mathbf{n}^\perp$ is a vector that is perpendicular to $\mathbf{n}$.

Some examples of topview projection on real scenes have been shown in Figure 22.

Fig. 22. Visual examples of top-down image projection.

D STATISTICS ON SUNCG

Table 1 and Table 2 collect statistics on Bedroom and Living Room, respectively. Table 3 and Table 4 collect statistics of different room shapes for Bedroom and Living Room, respectively.

Received February 2007; revised March 2009; final version June 2009; accepted July 2009

Name	window	bed	wardrobe
Count	8156	7288	7134
Name	stand	door	table lamp
Count	6921	6715	6375
Name	television	curtain	rug
Count	5396	5111	4705
Name	computer	computer	chandelier
Count	4687	4551	4372
Name	desk	picture frame	shelving
Count	4246	3772	3674
Name	dresser	plant	table
Count	3333	3168	2647
Name	dressing table	tv stand	books
Count	2473	2433	2339
Name	ottoman	mirror	air conditioner
Count	2256	2155	2153
Name	floor lamp	wall lamp	sofa
Count	2050	1953	1651
Name	vase	hanger	heater
Count	1642	1182	1104

Table 1. Names of classes and number of instances in each class of the Bedroom dataset.

Name	sofa	window	table
Count	9608	8217	7873
Name	chair	television	plant
Count	5898	5680	5070
Name	door	chandelier	rug
Count	4480	4129	3978
Name	curtain	tv stand	picture frame
Count	3936	3778	3385
Name	shelving	floor lamp	loudspeaker
Count	3082	3059	2599
Name	vase	ottoman	computer
Count	2514	1871	1841
Name	books	fireplace	air conditioner
Count	1773	1389	1341
Name	wall lamp	wardrobe	clock
Count	1286	1255	1238
Name	stereo set	kitchen cabinet	desk
Count	1204	1178	1159
Name	heater	fish tank	playstation
Count	1016	936	906

Table 2. Names of classes and number of instances in each class of the Living Room dataset.

Name	4 walls	5 walls	6 walls
Count	5270	283	1140
Name	7 walls	8 walls	9 walls
Count	110	135	21

Table 3. Number of instances for scenes with different number of walls in the Bedroom dataset. Other room shapes appear fewer than 20 times.

Name	4 walls	5 walls	6 walls
Count	3992	387	995
Name	7 walls	8 walls	9 walls
Count	275	164	73
Name	10 walls	11 walls	12 walls
Count	58	21	19

Table 4. Number of instances for scenes with different number of walls in the Living Room dataset. Other room shapes appear fewer than 10 times.