Neural Scene Decoration
from a Single Photograph

Hong-Wing Pang[1], Yingshu Chen[1], Phuoc-Hieu Le[2], Binh-Son Hua[2],
Duc Thanh Nguyen[3], and Sai-Kit Yeung[1]

[1] Hong Kong University of Science and Technology
[2] VinAI Research, Vietnam
[3] Deakin University

Abstract. Furnishing and rendering indoor scenes has been a long-standing task for interior design, where artists create a conceptual design for the space, build a 3D model of the space, decorate, and then perform rendering. Although the task is important, it is tedious and requires tremendous effort. In this paper, we introduce a new problem of domain-specific indoor scene image synthesis, namely neural scene decoration. Given a photograph of an empty indoor space and a list of decorations with layout determined by user, we aim to synthesize a new image of the same space with desired furnishing and decorations. Neural scene decoration can be applied to create conceptual interior designs in a simple yet effective manner. Our attempt to this research problem is a novel scene generation architecture that transforms an empty scene and an object layout into a realistic furnished scene photograph. We demonstrate the performance of our proposed method by comparing it with conditional image synthesis baselines built upon prevailing image translation approaches both qualitatively and quantitatively. We conduct extensive experiments to further validate the plausibility and aesthetics of our generated scenes. Our implementation is available at https://github.com/hkust-vgd/neural_scene_decoration.

Keywords: GANs, image synthesis, indoor scenes rendering

1 Introduction

Furnishing and rendering indoor scenes is a common task for interior design. This is typically performed by professional designers who carefully craft a conceptual design and furniture placement, followed by extensive modeling via CAD/CAM software to finally create a realistic image using a powerful rendering engine. Such a task often requires extensive background knowledge and experience in the field of interior design, as well as high-end professional software. This makes it difficult for lay users to design their own scenes from scratch.

On the other hand, different image synthesis methods have been developed and become popular in the field. Various types of deep neural networks - typically in the form of auto-encoders [20] and generative adversarial networks [5] - have

shown their capability of synthesizing images from a variety of inputs, e.g., semantic maps, text description.

Deep neural networks have also been explored in the generation of indoor scenes. Specifically, an indoor scene can be generated from a collection of objects placed in specific arrangements in 3D space or 2D forms such as top-down coordinates of objects or a floor plane.

In this paper, we combine both research directions: image synthesis and scene generation, into a new task, namely *neural scene decoration*. To solve this task, we propose a scene generation architecture that accepts an empty scene and an object layout as input and produces images of the scene decorated with furniture provided in the object layout (see Figure 2). Although several existing image synthesis techniques could be applied to create scenes from simple input, e.g., object bounding boxes, the problem of scene generation from an empty background has not yet been adequately explored. Our goal is to generate images of decorated scenes with improved visual quality, e.g., the placement of objects should be coherent.

An immediate application of neural scene decoration is creating conceptual designs of interior space. In particular, based on a few example images of a scene, a conceptual design of the scene can be made prior to texturing and rendering objects in the scene. Despite the availability of computer-aided design software, creating interior designs is still a challenging task as it requires close collaborations between artists and customers, and may involve third parties. The entire process of making interior designs is also expensive as it requires manual operations, and thus could take several days to complete one design. In this paper, we aim to make neural scene decoration simple and effective in creating realistic furnished scenes. To this end, we have made the following contributions in our work.

- A new task on scene synthesis and modeling that we name as *neural scene decoration*: synthesize a realistic image with furnished decorations from an empty background image of a scene and an object layout.
- A neural network architecture that enables neural scene decoration in a simple and effective manner.
- Extensive experiments that demonstrate the performance of our proposed method and its potential for future research. Quantitative evaluation results show that our method outperforms previous image translation works. Qualitative results also confirm the ability of our method in generating realistic-looking scenes.

2   Related Work

2.1   Image Synthesis

Prior to the resurgence of deep learning, editing a single photograph has often been done by building a physical model of the scene in the photo for object

composition and rendering [18,3,19,17]. Such an approach requires tremendous effort as it involves huge manual manipulations.

Deep neural networks, well-known for their learning and generalization capabilities, have been used recently for image synthesis. Technically, network architectures used in existing image synthesis methods fall within two categories: auto-encoders (AEs) [20] and generative adversarial networks (GANs) [5]. Conditional GAN (cGAN) [29] is a variant of GANs where generated results are conditioned on input data. A seminal work of cGAN for image synthesis is the image translation method (a.k.a pix2pix) in [10], where a cGAN was used to translate an image from one domain to another domain. Much effort has been made to extend the approach in various directions such as generation of high-resolution images [42], greater appearance diversity [51], and from arbitrary viewpoints [38]. CycleGAN proposed by Zhu et al. [50] extended the image translation to unpaired data (i.e., there are no pairs of images in different domains in training data) by adopting a cycle-consistency loss. Bau et al. [1] allowed users to edit the latent layer in the generator to control the generated content. Recently, Park et al. [33] proposed spatially-adaptive normalization (SPADE) to modulate intermediate activations, opposed to feeding input data directly into the generator, to strengthen semantic information during the generation process.

Recent developments in GANs led to the generation of high-resolution images [11] with different styles [14,16,12] and fewer aliasing [13] in the family of StyleGANs [15] especially for human faces. For natural and indoor scenes, GANs have also been applied to 2D layout generation. For example, in LayoutGAN [22], a layout was treated as an image and generated in a GAN-style manner. In HiGAN [43], latent layers capturing semantic information such as layout, category and lighting were used for scene synthesis. In HouseGAN [30], planar apartment room layout and furniture layout were generated. A class of conditional image synthesis methods focuses on the problem of translating layout to images [24,35,6,25]. In these methods, Li et al. [24] combined salient object layout and background hallucination GAN (BachGAN) [24] for image generation. LostGANs [35] explored the reconfigurability of the layouts, but their results mainly have objects on a simple background, which is structurally different from indoor scenes where objects and background must be geometrically aligned. Recent works focus on improving the image generation quality by further using context [6] and locality [25]. Compared to these works, our method is a conditional image synthesis focused on indoor scenes while the previous works are often tested with images in the wild like those in MS-COCO [27] and Visual Genome dataset [21]. The most related work to ours is perhaps BachGAN, where the generation of indoor scene images from layouts are demonstrated but with random backgrounds.

2.2   Scene Modeling

The problem of neural scene decoration is also relevant to the topic of image-based and 3D scene modeling. Earlier works include the spatial arrangement of objects into a 3D scene with spatial constraints, which is often modeled into objective

functions that can be solved using optimization techniques [4,44]. Additional constraints can also be used to model object relations in a scene. For example, Henderson et al. [7] modeled the relationships among object placement, room type, room shape, and other high-end factors using graphical models. Recently, Li et al. [23] defined object relations in a scene in a hierarchical structure. Ritchie et al. [34] proposed to iteratively insert objects into a scene from a top-down view by four different convolutional neural networks (CNNs) capturing the category, location, orientation, and dimension of objects. Wang et al. [40] proposed PlanIT, a framework that defines a high-level hierarchical structure of objects before learning to place objects into a scene. Zhang et al. [48] proposed a GAN-like architecture to model the distribution of position and orientation of indoor furniture, and to jointly optimize discriminators in both 3D and 2D (i.e., rendered scene images). Several works utilized spatial constraints as priors, e.g., relation graph prior [40,9,30], convolution prior [41], and performed well on spatial organization.

Our solution for neural scene decoration shares some ideas with scene unfurnishing [46] and scene furnishing [45,47,26]. Instead of utilizing RGBD as input [46], our method takes only a single input photograph of an empty scene and an object layout. Both ClutterPalette [45] and MageAdd [47] make 3D scenes by letting users select objects from a synthetic object database. In contrast, our method generates 2D scenes by automatically learning object appearance from training data. DecorIn [26] only predicts decoration locations on walls, while our method really decorates a scene image from a given background and object layout.

3   Proposed Method

3.1   Problem Formulation

Our goal is to develop a neural scene decoration (NSD) system that produces a decorated scene image $\hat{Y} \in \mathbb{R}^{3 \times W \times H}$, given a background image $X \in \mathbb{R}^{3 \times W \times H}$, and an object layout L for a list of objects to be added in the scene (see Figure 2). Note that both the generated image $\hat{Y}$ and the background image X are captured from the same scene. Ideally, the NSD system should be able to make $\hat{Y}$ realistically decorated with objects specified in L, and also assimilate $\hat{Y}$ to the provided background image X.

The format of the object layout L is crucial in determining how easy and effective the NSD system is. In SPADE [33], synthesized objects are labeled using a pixel-wise fashion. This manner, however, requires detailed labeling which is not effective in describing complicated objects and also takes considerable effort. In our work, we propose to represent L using simple yet effective formats: *box label* and *point label*. Specifically, let $O = \{o_1, ..., o_N\}$ be a set of objects added to X; these objects belong to K different classes, e.g., chair, desk, lamp, etc. Each object o_i is associated with a class vector $I_i \in \{0, 1\}^{K \times 1}$ and a layout map $f_i \in \mathbb{R}^{1 \times W \times H}$. The class vector I_i is designed such as $I_i(k) = 1$ if k is the class

ID of o_i, and $I_i(k) = 0$, otherwise. We define the object layout L as a 3D tensor: $L \in \mathbb{R}^{K \times W \times H}$ as follow:

$$L = \sum_{i=1}^{N} I_i f_i \qquad (1)$$

Box label. Like BachGAN [24], a box label indicates the presence of an object by its bounding box. For each object o_i, the layout map f_i is constructed by simply filling the entire area of the bounding box of o_i with 1s, and elsewhere with 0s. Mathematically, we define:

$$f_i(1, x, y) = \begin{cases} 1, & \text{if } (x, y) \in \text{bounding\_box}(o_i) \\ 0, & \text{otherwise} \end{cases} \qquad (2)$$

Box label format has the advantage of indicating the boundary where objects should be inserted and allowing finer control over the rough shape of objects.

Point label. We devise another representation for f_i, where each o_i is specified by its center $\mathbf{c}_i = (c_{i,x}, c_{i,y})$ and its size s_i (i.e., a rough estimate of the area of o_i). f_i is defined as a heat-map of $\mathbf{c}_i$ and s_i as follow:

$$f_i(1, x, y) = \exp\left(-\frac{\|(x, y) - (c_{i,x}, c_{i,y})\|_2^2}{s_i} \right) \qquad (3)$$

We find this format interesting to explore, as it allows the user to place decorated objects by simply specifying a rough location and size. The exact forms of decorated objects are automatically inferred based on observed training data.

3.2 Architecture Design

Our generator $G(X, L)$ is designed to take a pair of a background image X and an object layout L, and generate a decorated scene image $\hat{Y} = G(X, L)$. We also train a discriminator $D(\hat{Y}, L)$ to classify the image $\hat{Y}$ (as synthesized image vs. real image) and to validate if $\hat{Y}$ conforms to the object layout L.

Generator The generator $G(X, L)$ is depicted in Figure 1, and includes multiple generator blocks varying from low to high resolutions. For example, the "GeneratorBlock4 $\rightarrow$ 8" generates a feature map of spatial dimension 8×8 from 4×4.

Each generator block takes input from a feature map produced by its preceding generator (except the first one), an object layout and a background image at a corresponding resolution, and results in a feature map at a higher resolution. Each generator employs a SPADE block [33] to learn an object layout feature map (used to condition generated content), then up-samples the feature map (by a factor of 2), concatenates it with a down-scaled version of the background
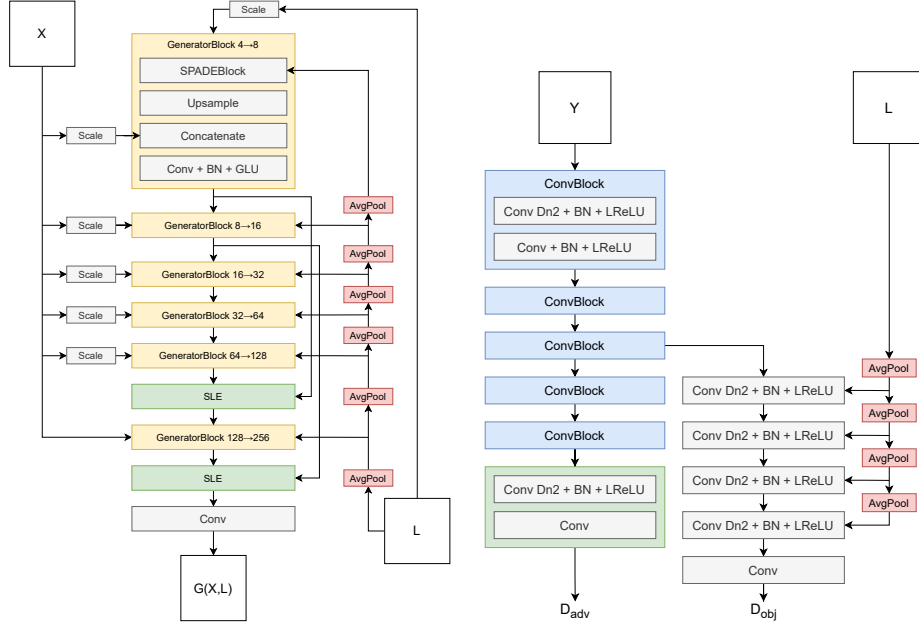
Fig. 1. Overview of our generator (left) and discriminator (right). Convolution layers labeled with Dn2 halve the spatial dimensions of input feature maps using stride 2.

image, and finally performs a convolution (Conv) with batch normalization (BN) and GLU. We follow [28] using skip-layer excitations (SLE) that modulates the output of the last two generator blocks with that of the first two blocks. The resulting feature map is passed through a final convolution layer to produce the synthesized image $\hat{Y}$.

To enforce the integration of object layout in the generation process, we insert L into every generator block in a bottom-up manner. Specifically, down-scaled versions of L are created by consecutive average pooling layers and then fused with corresponding feature maps at different resolutions. Likewise, to constrain the consistency in the background of the synthesized image $\hat{Y}$ and the input image X, we make X in different scales and insert these scaled images into every generator block. We refer readers to the supplementary material for further details of our architecture.

Discriminator The discriminator $D(Y, L)$ consists of an adversarial discriminator $D_{adv}(Y)$ and an object layout discriminator $D_{obj}(Y, L)$. We show our discriminator in Figure 1.

The adversarial discriminator D_{adv} takes in only the generated image $\hat{Y} = G(X, L)$ as input, and is solely responsible for fitting it towards the target image distribution. In our implementation, D_{adv} is a feedforward network and associated with a global adversarial loss.

We introduce an additional discriminator, D_{obj}, to encourage the generated image to follow the object layout provided in L. D_{obj} branches off from D_{adv} at the feature map of size 32×32 (see Figure 1). In this branch, each feature map is concatenated with an object layout L_i in a proper size $s_i \in \{4, 8, 16, 32\}$. We empirically chose to branch off from D_{adv} at this level as experimental results show that this is sufficient for generating reasonable images. We also found the use of D_{obj} in addition to D_{adv} produces better generation results (see our ablation study).

3.3   Training

We train our NSD system using the conventional GAN training procedure, i.e., jointly optimizing both the generator G and discriminator D. Specifically, we make use of the hinge adversarial loss function to train D as:

$$\mathcal{L}_D = \mathbb{E}_Y[\max(0, 1 + D_{adv}(Y))]$$
$$+ \mathbb{E}_{\hat{Y}, L}[\max(0, 1 - D_{adv}(\hat{Y}) - \lambda_{obj} D_{obj}(\hat{Y}, L)] \tag{4}$$

where Y is the decorated scene image paired with X (from training data) and $\hat{Y} = G(X, L)$. Y is also the source image where objects in L are defined.

The generator G is updated to push the discriminator's output towards the real image direction:

$$\mathcal{L}_G = \mathbb{E}_{X,L}[D(G(X, L))] \tag{5}$$

In our implementation, we optimize both $\min_D \mathcal{L}_D$ and $\min_G \mathcal{L}_G$ simultaneously and iteratively. We observed that setting λ_{obj} to small values is sufficient for generating plausible results. We set $\lambda_{obj} = 0.01$, which empirically works well in our experiments. We trained the discriminator and generator with Adam optimizer. We set the learning rate and batch size to 0.0001 and 32, respectively, and updated the gradients every four iterations. An exponential moving average of the generator weights was applied. The training was conducted on a single RTX 3090 GPU and within 400,000 iterations, which took approximately 44 hours. For inference, the generator performed at 0.760 seconds per image on average.

4   Experiments

4.1   Dataset

We chose to conduct experiments on the Structured3D dataset [49], as it is, to the best of our knowledge, the only publicly available dataset with pre-rendered image pairs of empty and decorated scenes. The Structured3D dataset consists of $78,463$ pairs of decorated and empty indoor images, rendered from a total of $3,500$ distinct 3D scenes. Following the recommendation of the dataset authors, we use 3,000 scenes for training and the remaining scenes for validation.

The Structured3D dataset consists of a variety of indoor scenes, including bedrooms, living rooms, and non-residential locations. However, we carried out experiments on bedrooms and living rooms scenes as those scenes contain sufficient samples for training and testing.

4.2   Baselines

Since we propose a new problem formulation, it is hard to find existing baselines that exactly address the same problem setting. However, our research problem and conditional image synthesis somewhat share a common objective, which is to generate an image conditioned on some given input. Therefore, some image synthesis methods could be adapted and modified to build baselines for comparison. Particularly, we adopted three state-of-the-art image synthesis methods, namely SPADE [33], BachGAN [24], and context-aware layout to image generation [6] for our baselines. Those methods were selected for several reasons. First, they follow conditional image generation setting, i.e., generated contents are conformed to input conditions, e.g., semantic maps in SPADE [33] and BachGAN [24], and layout structures in [6,25,36]. Second, their input conditions can be customised with minimal modifications to be comparable with ours. We present these modifications below.

SPADE [33] is a generative architecture conditioned on pixel-wise semantic maps. We concatenate the background image and object layout into each SPADE layer, in place of pixel-wise semantic maps. We also concatenate this tensor to the generated image before passing it to the discriminator.

BachGAN [24] is built upon the SPADE's generator and synthesizes a scene image from foreground object bounding boxes. It uses a *background hallucination module* to make generated background match with the object layout. To conform BachGAN to our problem setting, we directly feed the input background image X into the background hallucination module, instead of pooling features from multiple background image candidates.

We also compare to the context-aware layout to image generation method by He et al. [6], which is state-of-the-art in layout-based image generation for general scenes. Their method is reportedly better than LostGANs v2 [36] in terms of quantitative evaluation, and the implementation is publicly available. We modified their implementation by inserting the background into every block in their generator.

In the supplementary material, we provide an additional comparisons to GLIDE [31], a text-guided image synthesis method, which also use coarse layout descriptions similar to ours, unlike fine-grained semantic maps. Please refer to the supplementary material for this result.

4.3   Quantitative Evaluation

We quantitatively evaluate the image synthesis ability of our method using the Frechet Inception Distance (FID) [8] and the Kernel Inception Distance

| | Bedroom | | | | Living room | | | |
| | Box label | | Point label | | Box label | | Point label | |
Method	FID ↓	KID×10^3 ↓	FID ↓	KID×10^3 ↓	FID ↓	KID×10^3 ↓	FID ↓	KID×10^3 ↓
SPADE [33]	23.780	12.622	20.345	9.850	21.527	12.594	19.471	10.412
BachGAN [24]	21.319	10.054	18.829	7.932	20.463	11.961	18.997	9.446
He et al. [6]	21.311	**9.869**	18.899	7.958	19.732	10.828	18.762	9.656
Ours	**20.596**	11.609	**15.108**	**6.797**	**18.478**	**10.113**	**17.986**	**9.421**

Table 1. Quantitative assessment of our method against various baselines. Lower FID/KID values indicate better performance.

| | Bedroom | | | | Living room | | | |
| | Boxes | | Points | | Boxes | | Points | |
Descriptor	FID ↓	KID×10^3 ↓	FID ↓	KID×10^3 ↓	FID ↓	KID×10^3 ↓	FID ↓	KID×10^3 ↓
D_{adv}	22.341	13.245	21.069	11.231	25.051	15.950	24.786	14.562
$D_{adv} + D_{obj}$	**20.596**	**11.609**	**15.108**	**6.797**	**18.478**	**10.113**	**17.986**	**9.421**

Table 2. Comparison of the use of D_{adv} only, and the combination of D_{adv} and D_{obj} as in our design.

(KID) [2]. Both FID and KID measure the dissimilarity between inception representations [37] of a synthesized output and its real version. Wasserstein distance and polynomial kernel are used in FID and KID respectively as dissimilarity metrics [32].

For quantitative evaluation purpose, we used pairs of background and decorated images from ground-truth. We also extracted bounding boxes and object masks of decorated objects from the ground-truth to construct box labels and point labels. We report the performance of our method and other baselines in Table 1. As shown in the results, our method outperforms all the baselines on both bedroom and living room test sets, with both box label and point label format in FID metric. The same observation is true for the KID metric, with the only exception that He et al.'s method [6] is ranked best with box label format for bedroom scenes.

Our method also has a computational advantage. Specifically, the BachGAN baseline took roughly the same amount of training time as our method but required four GPUs. In contrast, our method can produce even better results using less computational resources.

Table 1 also shows that point label scheme slightly outperforms box label scheme. However, as discussed in the next section, each scheme is favored to specific types of objects. For example, from a usage perspective, box label format has a strong focus on user's desire on how a decorated object appears, while point label format offers more flexibility and autonomy to the NSD system.

Input	BachGAN	SPADE	He et al.	Ours

■ lamp   ■ picture   ■ bed   ■ curtain   ■ desk   ■ chair   ■ nightstand   ■ sofa

Fig. 2. Generation results of our method and other baselines, using box label format (the first two rows) and point label format (the last two rows). For point label format, the center and radius of each circle represent the location $\mathbf{c}_i$ and size s_i of an object (see Eq. (3)). Best view with zoom.

4.4   Ablation Study

In this experiment, we prove the role of the additional discriminator D_{obj}. Recall that D_{obj} is branched off from D_{adv} and combines L at various scales (see Figure 1). To validate the role of D_{obj}, we amended the architecture of the discriminator D by directly concatenating the object layout L with the decorated image Y to make the input for D, like the designs in [33] and [24]. Experimental results are in Table 2, which clearly confirms the superiority of our design for the discriminator (i.e., using both D_{adv} and D_{obj}) over the use of D_{adv} only.

4.5   Qualitative Evaluation

Generation quality. We qualitatively compare our method with the baselines, on both box label and point label format in Figure 2. As shown in these results,
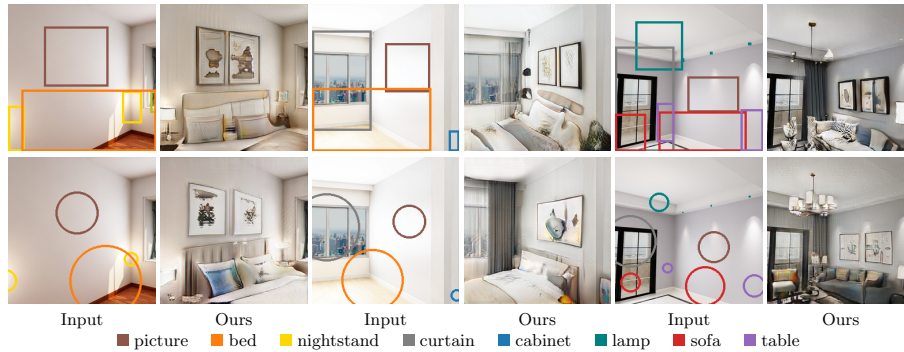
| Input | Ours | Input | Ours | Input | Ours |

■ picture ■ bed ■ nightstand ■ curtain ■ cabinet ■ lamp ■ sofa ■ table

Fig. 3. Generation results (from the same input) using box label format (top row) and point label format (bottom row). While box label format suits small and relatively fixed-size objects, point label format is more flexible to describe large objects whose dimensions can be adjusted automatically.

Method	Bedroom		Living room	
	FID	KID$_{\times 10^3}$	FID	KID$_{\times 10^3}$
SPADE [33]	22.985	10.993	23.630	13.620
BachGAN [24]	19.914	8.119	23.391	13.006
Ours	**17.489**	**8.007**	**21.019**	**11.277**

Table 3. Performance of our method and the baselines using default object sizes. Lower FID/KID values indicate better performance.

our method is able to generate details in foreground objects. We hypothesize that our method successfully incorporates the layout information in early layers in the generator as each pixel in a down-scaled object layout encourages object generation within a specific local region in the output image.

Box label vs. point label. Figure 3 visualizes some generation results using box label and point label format on the same input background. In this experiment, on each scene, box labels and point labels were derived from the same set of objects. We observed that some object classes are better suited to a particular label format. For example, small objects and those whose aspect ratio can be varied (e.g., pictures can appear in either portrait or landscape shape) should be described using box label format. On the other hand, objects that often occupy large areas in a scene, such as beds and sofas, tend to have less distortion and clearer details when represented with point label format.

Fig. 4. Generation results using default values for object size.

	Bedroom		Living room	
Method	FID	KID$_{\times 10^3}$	FID	KID$_{\times 10^3}$
Class mean	20.145	10.457	27.365	14.639
Class median	17.489	8.007	21.019	11.277
$m = 4.0$	24.045	14.701	28.128	15.979
$m = 2.5(*)$	**15.108**	**6.797**	**17.986**	**9.421**
$m = 1.0$	20.201	9.105	19.114	11.797

Table 4. Performance using different object sizes methods. (*) indicate the default setting used in our experiments.

4.6 Setting Object Sizes

In the quantitative assessment, the sizes of decorated objects in the point label format (i.e., s_i in Eq. (3)) were retrieved from ground-truth. However, in reality, this information is provided by the user. In this experiment, we investigate a simpler input for the point label format where the object sizes are set by default values rather than given by either the ground-truth or user. In particular, for each decorated object o_i, we set the size s_i to the median size of all the objects having the same object class with o_i in the training data. Particularly, the ground truth value of s_i of each object is given by $s_i = m\sqrt{A_i}$, where m is a fixed constant and A_i is the area (i.e., the no. of pixels) of the object mask of o_i. Here we set $m = 2.5$ for our experiments.

We applied this setting to all the baselines and reported their performance in Table 3. We observed that compared with using ground-truth values for the
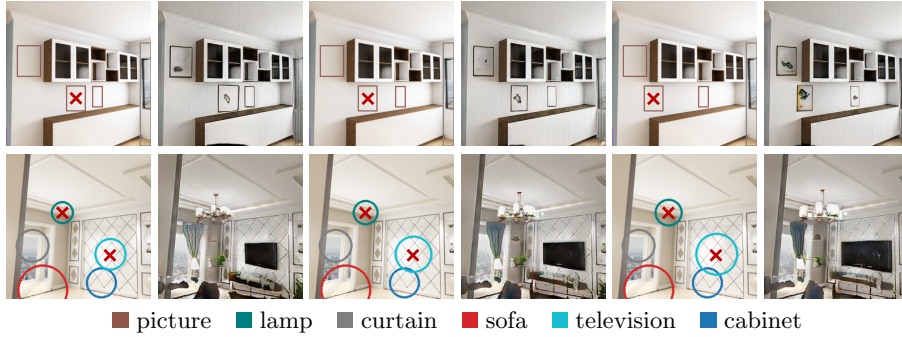
picture ■ lamp ■ curtain ■ sofa ■ television ■ cabinet

Fig. 5. Generation results under different settings for objects' locations and sizes. Manipulated objects are marked with "X". Top two rows: we change the location of a painting by moving its bounding box towards the left. Bottom two rows: we adjust the size of a ceiling light and a TV by changing the radius at their centers.

Background   Generated   Background   Generated

■ picture ■ bed ■ nightstand ■ lamp

Fig. 6. Generation results on real-world scenes.

object sizes (see Table 1), setting the object sizes to default values degrades the performance of all the competitors. However, our method still consistently outperforms all the baselines on all the test sets, using both FID and KID metrics. We illustrate several results of this setting in Figure 4. In Table 4, we further compare different variants of point label format, including the use of mean value and median value to compute s_i. We show the results of using alternate values for m in Table 4, which clearly confirms our choice for m for the best performance.

4.7   Layout Manipulation

We demonstrate in Figure 5 how our method performs under various settings for decorated objects' positions and sizes. For box label format, these settings can be done by changing the location and dimensions of objects' bounding boxes. For point label format, we adjust the center position and radius of objects. As shown in Figure 5, our method can generate contents adaptively to different settings, proving its applicability in creating conceptual designs of interior space.

	Preference (%)	
Ours	68.52 / 31.48	SPADE
	62.56 / 37.44	BachGAN
	63.67 / 36.33	He et. al.

(a)

	Preference (%)	
Ours-box	64.54 / 35.46	Baselines
Ours-point	70.03 / 29.97	Baselines

(b)

	Preference (%)	
Ours (BR)	67.01 / 32.99	Baselines
Ours (LR)	67.58 / 32.42	Baselines

(c)

Fig. 7. User study's results: Preference with (a) different methods, (b) box labels vs point labels, and (c) different room types. (BR = bedroom, LR = living room)

Please also refer to our supplementary material for additional discussions on the diversity of the generation results, the impact of the background to furniture, and iterative decoration, where the furniture can be added one by one.

4.8   Generalization to Real-World Data

To validate the universality of our method, we experimented with it on several real-world scenes not included in the Structured3D dataset. We present several results of this experiment in Figure 6. Interestingly, we discovered that paintings and furniture are generated in different artistic styles but still in line with the entire decoration and background style.

4.9   User study

In addition to quantitative evaluation, we conducted a user study on the generation quality of our method and other baselines. We collected responses from 26 participants and summarized these results in Figure 7. I general, our model is often preferred in generating images with point label format, especially in the bedroom test case with fewer objects and clutter. When using box label format, our method still produces results around the same level of quality compared with the baselines. More details of our user study is in the supplementary material.

5   Conclusion

We introduced a new task called neural scene decoration. The task aims to render an empty indoor space with furniture and decorations specified in a layout map. To realize this task, we propose an architecture conditioned on a background image and an object layout map where decorated objects are described via either bounding boxes or rough locations and sizes. We demonstrate the capability of our method in scene design over previous works on the Structured3D dataset. Neural scene decoration is henceforth a step toward building the next generation of user-friendly interior design and rendering applications. Future work may include better support of sequential object generation [39], interactive scene decoration, and integration of more advanced network architecture.

References

1. Bau, D., Strobelt, H., Peebles, W.S., Wulff, J., Zhou, B., Zhu, J., Torralba, A.: Semantic photo manipulation with a generative image prior. ACM Transactions on Graphics **38**(4), 1–11 (2019) 3
2. Bińkowski, M., Sutherland, D.J., Arbel, M., Gretton, A.: Demystifying MMD GANs. In: Proceedings of the International Conference on Learning Representations (2018) 9
3. Fisher, M., Ritchie, D., Savva, M., Funkhouser, T.A., Hanrahan, P.: Example-based synthesis of 3d object arrangements. ACM Transactions on Graphics **31**(6), 1–11 (2012) 3
4. Germer, T., Schwarz, M.: Procedural arrangement of furniture for real-time walkthroughs. Computer Graphics Forum **28**(8), 2068–2078 (2009) 4
5. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Proceedings of the Advances in Neural Information Processing Systems (2014) 1, 3
6. He, S., Liao, W., Yang, M., Yang, Y., Song, Y.Z., Rosenhahn, B., Xiang, T.: Context-aware layout to image generation with enhanced object appearance. In: CVPR (2021) 3, 8, 9
7. Henderson, P., Subr, K., Ferrari, V.: Automatic generation of constrained furniture layouts. arXiv preprint arXiv:1711.10939 (2017) 4
8. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs trained by a two time-scale update rule converge to a local nash equilibrium. In: Proceedings of the Advances in Neural Information Processing Systems (2017) 8
9. Hu, R., Huang, Z., Tang, Y., van Kaick, O., Zhang, H., Huang, H.: Graph2Plan: Learning floorplan generation from layout graphs. ACM Transactions on Graphics **39**(4), 118–128 (2020) 4
10. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017) 3
11. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation. In: Proceedings of the International Conference on Learning Representations (2018) 3
12. Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., Aila, T.: Training generative adversarial networks with limited data. In: Proceedings of the Advances in Neural Information Processing Systems (2020) 3
13. Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., Aila, T.: Alias-free generative adversarial networks. arXiv preprint arXiv:2106.12423 (2021) 3
14. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019) 3
15. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(12), 4217–4228 (2021) 3
16. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of StyleGAN. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020) 3
17. Karsch, K.: Inverse Rendering Techniques for Physically Grounded Image Editing. Ph.D. thesis, University of Illinois at Urbana-Champaign (2015) 3

18. Karsch, K., Hedau, V., Forsyth, D., Hoiem, D.: Rendering synthetic objects into legacy photographs. ACM Transactions on Graphics **30**(6), 1–14 (2011) 3
19. Karsch, K., Sunkavalli, K., Hadap, S., Carr, N., Jin, H., Fonte, R., Sittig, M., Forsyth, D.: Automatic scene inference for 3D object compositing. ACM Transactions on Graphics **33**(3), 1–15 (2014) 3
20. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. In: Proceedings of the International Conference on Learning Representations (2014) 1, 3
21. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.J., Shamma, D.A., Bernstein, M., Fei-Fei, L.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. In: International Journal of Computer Vision (2017) 3
22. Li, J., Yang, J., Hertzmann, A., Zhang, J., Xu, T.: LayoutGAN: Generating graphic layouts with wireframe discriminators. In: Proceedings of the International Conference on Learning Representations (2019) 3
23. Li, M., Patil, A.G., Xu, K., Chaudhuri, S., Khan, O., Shamir, A., Tu, C., Chen, B., Cohen-Or, D., Zhang, H.R.: GRAINS: generative recursive autoencoders for indoor scenes. ACM Transactions on Graphics **38**(2), 1–16 (2019) 4
24. Li, Y., Cheng, Y., Gan, Z., Yu, L., Wang, L., Liu, J.: BachGAN: High-resolution image synthesis from salient object layout. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020) 3, 5, 8, 9, 10, 11
25. Li, Z., Wu, J., Koh, I., Tang, Y., Sun, L.: Image synthesis from layout with locality-aware mask adaption. In: ICCV (2021) 3, 8
26. Liang, Y., Fan, L., Ren, P., Xie, X., Hua, X.S.: Decorin: An automatic method for plane-based decorating. IEEE Transactions on Visualization and Computer Graphics (2021) 4
27. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft coco: Common objects in context. In: ECCV (2014) 3
28. Liu, B., Zhu, Y., Song, K., Elgammal, A.: Towards faster and stabilized GAN training for high-fidelity few-shot image synthesis. In: Proceedings of the International Conference on Learning Representations (2021) 6
29. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014) 3
30. Nauata, N., Chang, K.H., Cheng, C.Y., Mori, G., Furukawa, Y.: House-GAN: Relational generative adversarial networks for graph-constrained house layout generation. In: Proceedings of the European Conference on Computer Vision (2020) 3, 4
31. Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., Chen, M.: GLIDE: towards photorealistic image generation and editing with text-guided diffusion models. arXiv preprint 2112.10741 (2021) 8
32. Obukhov, A., Seitzer, M., Wu, P.W., Zhydenko, S., Kyl, J., Lin, E.Y.J.: High-fidelity performance metrics for generative models in pytorch (2020) 9
33. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019) 3, 4, 5, 8, 9, 10, 11
34. Ritchie, D., Wang, K., Lin, Y.a.: Fast and flexible indoor scene synthesis via deep convolutional generative models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019) 4
35. Sun, W., Wu, T.: Image synthesis from reconfigurable layout and style. In: ICCV (2019) 3

36. Sun, W., Wu, T.: Learning layout and style reconfigurable gans for controllable image synthesis. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) (2021) 8
37. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015) 9
38. Tang, H., Xu, D., Sebe, N., Wang, Y., Corso, J.J., Yan, Y.: Multi-channel attention selection GAN with cascaded semantic guidance for cross-view image translation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019) 3
39. Turkoglu, M.O., Thong, W., Spreeuwers, L., Kicanaoglu, B.: A layer-based sequential framework for scene generation with gans. In: AAAI Conference on Artificial Intelligence (2019) 14
40. Wang, K., Lin, Y.A., Weissmann, B., Savva, M., Chang, A.X., Ritchie, D.: Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. ACM Transactions on Graphics **38**(4), 1–15 (2019) 4
41. Wang, K., Savva, M., Chang, A.X., Ritchie, D.: Deep convolutional priors for indoor scene synthesis. ACM Transactions on Graphics **37**(4), 1–14 (2018) 4
42. Wang, T.C., Liu, M.Y., Zhu, J.Y., Tao, A., Kautz, J., Catanzaro, B.: High-resolution image synthesis and semantic manipulation with conditional GANs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018) 3
43. Yang, C., Shen, Y., Zhou, B.: Semantic hierarchy emerges in deep generative representations for scene synthesis. International Journal of Computer Vision **129**(5), 1451–1466 (2020) 3
44. Yu, L.F., Yeung, S.K., Tang, C.K., Terzopoulos, D., Chan, T.F., Osher, S.J.: Make it home: automatic optimization of furniture arrangement. ACM Transactions on Graphics **30**(4), 1–11 (2011) 4
45. Yu, L.F., Yeung, S.K., Terzopoulos, D.: The clutterpalette: An interactive tool for detailing indoor scenes. IEEE Transactions on Visualization and Computer Graphics (2015) 4
46. Zhang, E., Cohen, M.F., Curless, B.: Emptying, refurnishing, and relighting indoor spaces. ACM Transactions on Graphics **35**(6), 1–14 (2016) 4
47. Zhang, S.K., Li, Y.X., He, Y., Yang, Y.L., Zhang, S.H.: Mageadd: Real-time interaction simulation for scene synthesis. In: ACM International Conference on Multimedia (2021) 4
48. Zhang, Z., Yang, Z., Ma, C., Luo, L., Huth, A., Vouga, E., Huang, Q.: Deep generative modeling for scene synthesis via hybrid representations. ACM Transactions on Graphics **39**(2), 1–21 (2020) 4
49. Zheng, J., Zhang, J., Li, J., Tang, R., Gao, S., Zhou, Z.: Structured3D: A large photo-realistic dataset for structured 3D modeling. In: Proceedings of the European Conference on Computer Vision (2020) 7
50. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE International Conference on Computer Vision (2017) 3
51. Zhu, J.Y., Zhang, R., Pathak, D., Darrell, T., Efros, A.A., Wang, O., Shechtman, E.: Toward multimodal image-to-image translation. In: Proceedings of the Advances in Neural Information Processing Systems (2017) 3