Neural Scene Decoration from a Single Photograph –Supplementary Material–

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

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

Abstract. In this supplementary material, we first present more qualitative results with additional analysis on the diversity, background impact, an additional comparison to text-guided image synthesis (Section 1). We then provide an additional ablation study on our discriminator design (Section 2) along with complementary details of our user study (Section 3). Finally, we describe in detail our proposed architecture (Section 4) and its implementation (Section 5).

1 Qualitative Results

1.1 Diversity of generated images

Our method achieves diversity in the following ways. First, the input layout controls the output diversity. One can change the input layout to change how the scene is decorated. Second, given the same background and layout, diversity in the appearance of scene objects can still be obtained. Technically, this is achieved by changing the initial latent code of the generator and finetune the generator.

A current limitation is that our model ignores the noise vector, limiting diversity. This problem is also reported in pix2pix [2]. Revising our network architecture for greater diversity would be a future work, e.g., use the noise injection by OASIS [10].

Same background with different layouts. We show qualitative results of our method using different layouts on the same background image, demonstrating the diversity of generated contents. Several results of this experiment are presented in Figure 1. As can be seen, our model can provide plausible renderings given different layouts.

Same background with same layouts. Here we show results of our method using the same background and layout. The diversity is now controlled by the initial latent code of the generator. The results are presented in Figure 2. As can be seen, our model can provide plausible diversity in the object appearance.

Fig. 1: Diversity evaluation. Generation results under same background image X with different object layouts.

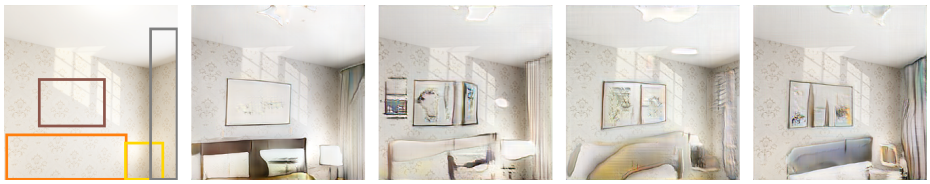


Fig. 2: Diversity evaluation. Generation results from same background image X with different model weights.

1.2 Impact of background to furniture generation.

We analyze the effect of the background to the generation of the furniture. The diagram below shows a simple example where we modify the background image by enlarging the left white backdrop. In Figure 3, we see that objects like paintings can conform to this structural change in the background. Other objects like beds only have appearance change. The quantization of the impact of background images is left for future work.

1.3 Iterative decoration

Our model is designed for inferring the decoration at once. While not designed for iterative object insertion, our method can add one object at a time to a limited extent, thanks to the diversity of the scenes and images in the dataset, as shown in the example in Figure 4. In future work, we could consider using object removal with image inpainting to augment the training data.

Fig. 3: Impact of background to furniture generation.

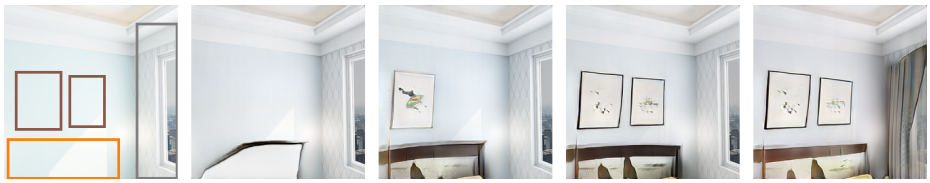


Fig. 4: Generation results by adding the objects one at a time.

1.4 Comparisons with text-guided image synthesis

We provide an additional comparison of our method with text-guided image synthesis, which also use coarse layout descriptions similar to ours, unlike fine-grained semantic maps. For text-guided methods, we chose GLIDE [8] and generated objects by masking target regions and providing a text prompt for each object. Specifically, we used released GLIDE (filtered) model for image inpainting in a masked region conditioned on a text prompt. We generate objects one by one iteratively via masking each object box with target object text to realize semantic spatially generation (Fig. 5 GLIDE-iter column). We also inpaint all areas of same boxes of given empty scene once with one text prompt (Fig. 5 GLIDE column). Compared to our method, GLIDE failed to preserve the background (e.g., windows) properly while the generated objects are unaware of the context, making their results not semantically consistent, e.g., the fireplace in the last image. Additionally, GLIDE takes 4 to 8 seconds to inpaint a 256×256 image which is much slower than our method.

1.5 Visual results of more scenes

We provide more qualitative results of our method and all baselines (SPADE [9], BachGAN [5], He et al. [1]) in Figure 6 and Figure 7. In general, we visually found that bedroom images are often generated in higher quality compared with living room images. This is because bedroom scenes have less variation in their structure, and there are typically less objects decorated in the scenes, leading to lower complexity in scene generation compared to living room scenes. Additionally, we observed that box label format shows more advantages in generating small and relatively fixed size objects. Point label format, on the other hand, allows

Fig. 5: Comparison to text-guided image synthesis method GLIDE [8].

flexibility in determining the object size and thus works well with large and shape-variable objects.

2 Ablation Study

In addition to the ablation study provided in the main paper, here we further explain our discriminator in detail. In the main paper, we take the generated image Y as input to the discriminator. This is known as the unconditional discriminator as it does not depend on the input X. In fact, image translation methods like pix2pix [2], pix2pixHD [11] and SPADE [9] showed that a conditional version of the discriminator can have better image fidelity. Particularly, the conditional discriminator takes a channel-wise concatenation of the original and generated image (background X and generated image Y in our case) as input to the discriminator. Here we provide an experiment to compare the use of unconditional and conditional discriminator in our case. Comparison results are reported in Table 1. As shown in the results, the unconditional discriminator has better results in most cases. The major difference between our method and image translation methods lies in our data, where the domain gap between the background and the decorated scene is less significant compared to data tested in image translation methods, i.e., sketch or semantic maps vs. real images. Therefore, we adopted the unconditional discriminator in our work.

3 User Study

Our user study has 26 participants; each participant was asked with 48 questions. For each question, we presented two decorated images, one image was generated with our method and the other one was generated by a baseline. Both images were generated from the same input scene. We asked each participant to choose

	Bedroom				Living room			
	Boxes		Points		Boxes		Points	
Discriminator	FID ↓	KID×10^3 ↓	FID ↓	KID×10^3 ↓	FID ↓	KID×10^3 ↓	FID ↓	KID×10^3 ↓
Without X	20.596	**11.609**	**15.108**	**6.797**	**18.478**	**10.113**	**17.986**	**9.421**
With X	**20.511**	12.490	16.038	12.564	21.640	14.850	21.653	14.974

Table 1: Comparison between unconditional discriminator (without X) and conditional discriminator (with X).

the image that they considered to be more natural and realistic. Each question belongs to one of 12 test settings, which is a combination of the following factors: 3 baselines to compare, 2 label formats (box label / point label), and 2 test cases (bedroom / living room). We randomly picked 4 samples for each setting, i.e., each participant was presented with a total of 48 image pairs in random order. The order that two images in a pair were shown in each question was also made randomly.

In general, our model is often preferred on images generated 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 with on par quality compared with the baselines.

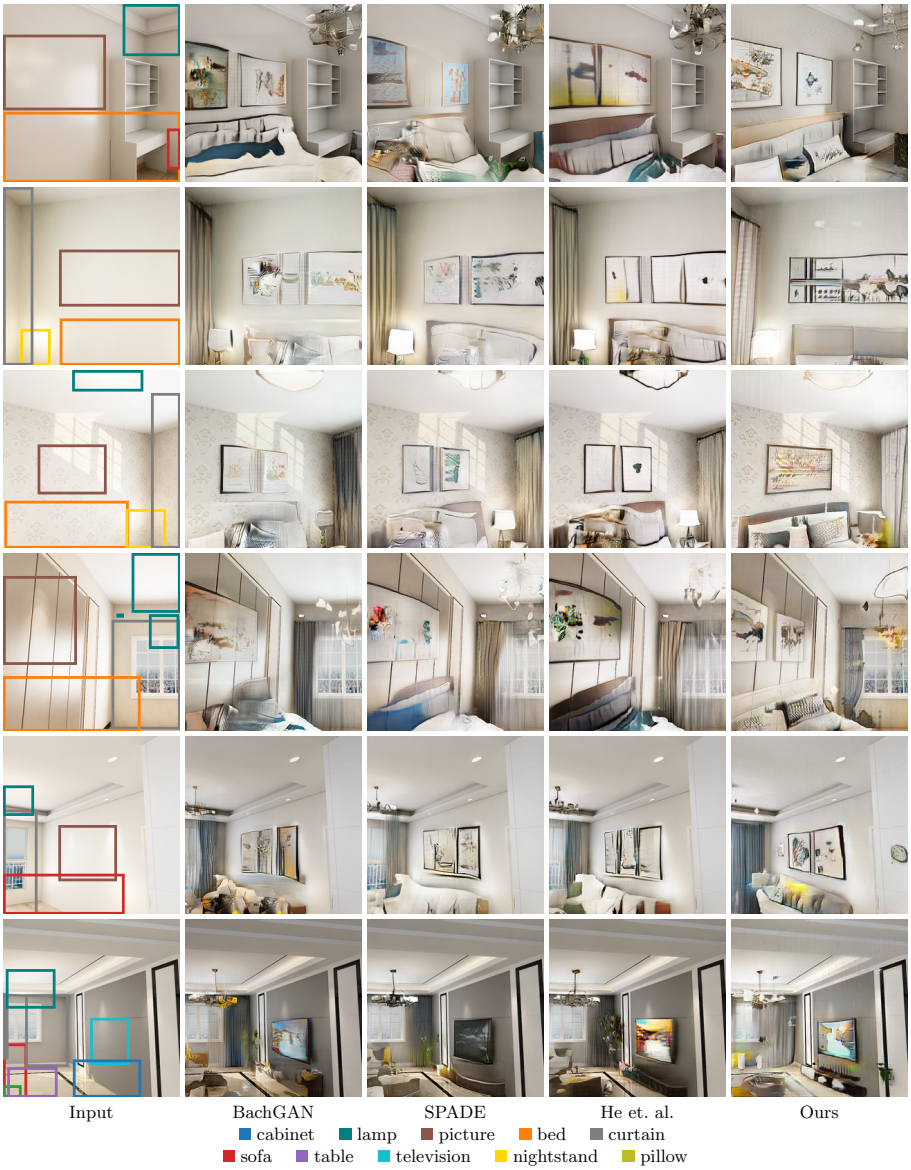


Fig. 6: Additional generation results for box label format.

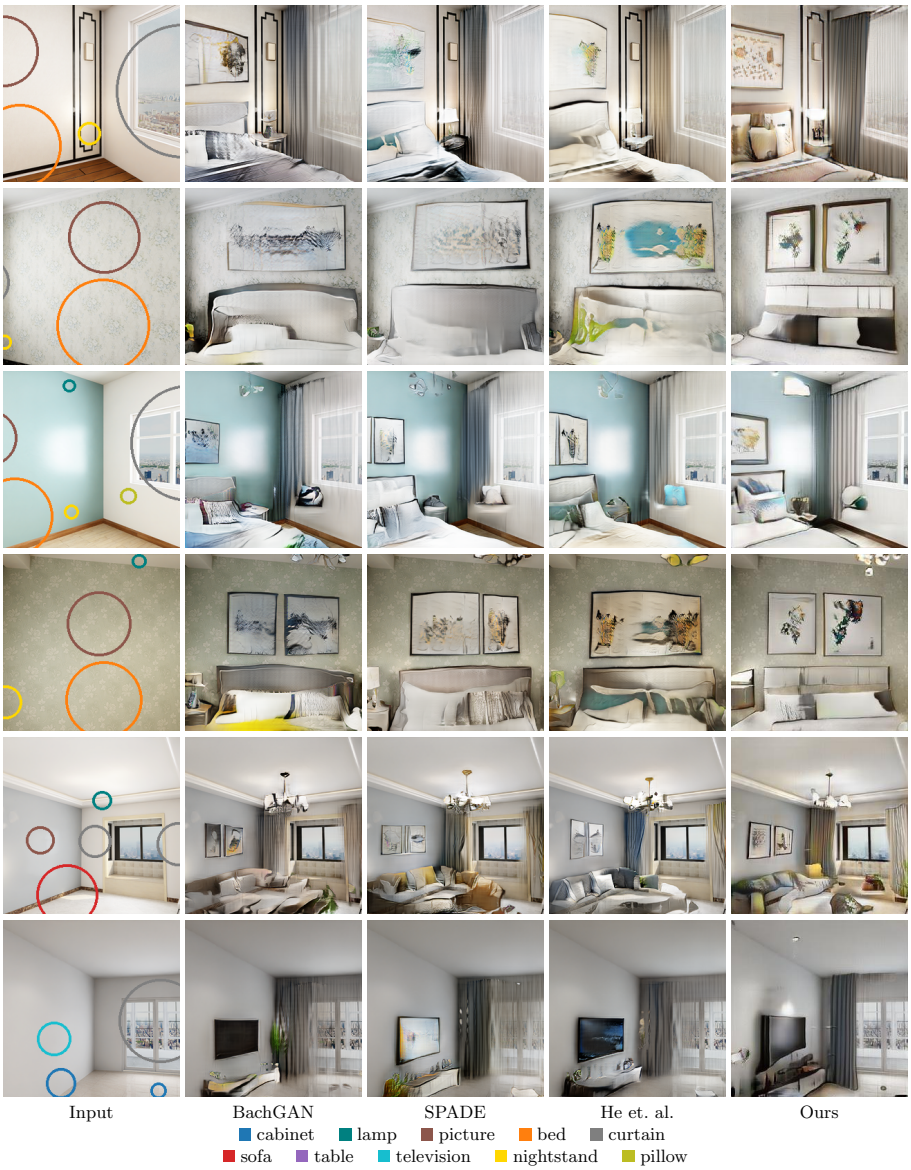


Fig. 7: Additional generation results for point label format.

4 Network Architecture

4.1 Generator

Table 2 describes the input and output dimensions used in the sequence of generator blocks in our generator. For each generator block with v_i input and v_o output channels, the object layout L first modulates the feature map using a SPADE residual block similar [9], which consists of two consecutive SPADE layers with ReLU activations, as well as a skip connection across the block. Unlike [9], we do not add a convolutional layer after each SPADE layer in the residual blocks. The number of channels remains to be v_i before and after the SPADE block, and the number of hidden channels in SPADE layers is set to $v_i/2$. Following the SPADE block, we upsample the feature map by a factor of 2, pass through a convolutional layer with $2c_o$ output channels, batch norm layer and finally through a gated linear unit (GLU), following the convolutional block implementation in [6]. All aforementioned convolutional layers have a kernel size of 3 and padding size of 1.

The last two generator blocks use the SLE module in [6] to modulate the feature maps with earlier, smaller-resolution feature maps. We pass the output of the source generator block through an adaptive pooling layer to reduce its spatial size to 4×4, then use a 4×4 convolutional layer of kernel size of 4 to collapse the spatial dimensions, reducing the feature map to a 1D vector. This is passed through a LeakyReLU (0.1) activation, 1×1 convolutional layer and sigmoid function to obtain a 1D vector of size v_o, where v_o is the number of output channels of the destination generator block. This vector is multiplied channel-wise with the feature map inside the destination generator block, right after the upsample operation.

4.2 Discriminator

The main discriminator D_{adv} consists of five discriminator blocks, followed by an output convolution module. Each discriminator block consists of two sets of convolutional layers. The first set has a kernel size of 4 and stride of 2, and is responsible for downsampling feature maps by a factor of 2. The second one has a kernel size of 3 and padding size of 1, and transforms the feature maps from v_i to v_o channels, where v_i and v_o are the numbers of channels listed in Table 3. The output convolution module downsamples the feature map to 4×4, and is followed by a final 4×4 convolution layer reducing the feature map to one single logit. The object layout discriminator D_{obj} takes the 32×32 feature map output from D_{adv} and repeatedly downsamples the feature map by a factor of 2, using convolutional layers with kernel size of 4 and stride of 2. Like D_{adv}, the final feature maps are reduced to a single logit via a final convolutional layer. Each convolutional layer in D_{adv} and D_{obj} - except the final layer - is followed by batch norm and LeakyReLU (0.1) activations.

Block #	Resolution	SLE source block	Features
2	$4 \rightarrow 8$	–	$12 \rightarrow 512$
3	$8 \rightarrow 16$	–	$512 \rightarrow 512$
4	$16 \rightarrow 32$	–	$512 \rightarrow 256$
5	$32 \rightarrow 64$	–	$256 \rightarrow 128$
6	$64 \rightarrow 128$	2	$128 \rightarrow 64$
7	$128 \rightarrow 256$	3	$64 \rightarrow 32$

Table 2: List of generator blocks and their properties.

Block #	Resolution	Features
7	$256 \rightarrow 128$	$3 \rightarrow 32$
6	$128 \rightarrow 64$	$32 \rightarrow 64$
5	$64 \rightarrow 32$	$64 \rightarrow 128$
4	$32 \rightarrow 16$	$128 \rightarrow 256$
3	$16 \rightarrow 8$	$256 \rightarrow 512$
Output	$8 \rightarrow 1$	$512 \rightarrow 1$
4	$32 \rightarrow 16$	$64 \rightarrow 128$
3	$16 \rightarrow 8$	$128 \rightarrow 256$
2	$8 \rightarrow 4$	$256 \rightarrow 256$
1	$4 \rightarrow 2$	$256 \rightarrow 256$
Output	$2 \rightarrow 1$	$256 \rightarrow 1$

Table 3: List of discriminator blocks in D_{adv} and convolution layers in D_{obj}.

5 Implementation Details

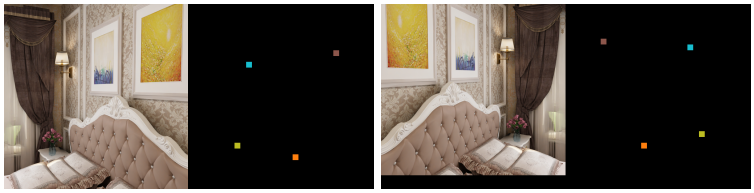
5.1 Dataset

As presented in the main paper, the semantic labels for images in the Structured3D dataset are retrieved from the NYU-Depth V2 dataset [7]. Five classes: `window`, `door`, `wall`, `ceiling`, and `floor` are considered as "background" and appear in both both empty and decorated scenes. The remaining classes represent "foreground" and are used in decorated scenes only. In addition, since the distribution of the foreground classes is highly unbalanced, and some classes do not really exist in the Structured3D dataset, only a subset of these foreground classes were used in our experiments. We show the list of the foreground classes used in our work in Table 4.

We carried out experiments on two subsets of the Structured3D dataset - bedrooms and living rooms, as those sets contain enough samples for training and testing. Note that each scene in the Structured3D dataset is associated with a room type label, that allows us to identify bedroom and living room scenes. To provide enough clue for a scene type, we filtered out images that contain less

Name	Color	Name	Color
cabinet		picture	
bed		curtain	
chair		television	
sofa		nightstand	
table		lamp	
desk		pillow	

Table 4: Foreground classes used in our work.



(a) (b)

Fig. 8: (a) Sample image with corresponding object layout map where each dot shows the location and semantic label (via the color) for an object. (b) Same sample after translation and horizontal flipping.

than 4 objects. For each source image, we resized the image from the original size 1280×720 to 456×256, then cropped two images with size 256×256 from each source image. Images were cropped such that, for foreground object pixels, at least 60% were still present in cropped regions. We report the total number of training and test samples for each set in Table 5.

5.2 Data augmentation

A direct consequence of training on smaller subsets of the Structured3D dataset is that the number of usable training samples the model observes is greatly reduced. To deal with this issue, we implemented the DiffAugment technique [12] in our training process. DiffAugment improves generation quality by randomly

Data split	No training images	No test images
Bedroom	28,038	4,931
Living room	19,636	3,976

Table 5: Statistics of data used for training and testing.

perturbing both the generated and real images with differentiable augmentations when training both G and D, and is reported to significantly boost the generation quality of state-of-the-art unconditional StyleGAN2 [4,3] architecture when training data is limited to a few thousand samples. Thus, we adopt this technique when training on our architecture, in order to compensate for the reduction of training samples.

While the authors of DiffAugment proposed multiple augmentation methods, we only applied translation augmentation to the images. This is because other methods (e.g., random square cutouts) may affect the integrity of decorated scene images. We set the translation augmentation probability to 30%, and also horizontally flipped the images for 50% of the time. For each augmented image, its corresponding object layout was also perturbed in the same manner. Figure 8 shows an example of our augmentation scheme.

5.3 Implementation Notes

While our proposed model can make plausible object locations, we notice that the arrangement of the objects currently lacks flexibility. For example, supplying an object label L with only one to two objects is less likely to result in realistic decorated scenes. This is probably due to the fact that the training dataset only contains fully decorated rooms, and therefore the generator is not trained to produce partially decorated rooms. Likewise, our model also tends to perform fairly on object arrangements that rarely occur in the training dataset.

Additionally, we found that multiple object instances in an image are occasionally labelled by Structured3D with the same object ID, e.g., paintings and curtains. This explains why a single `picture` object label can result in two (or more) generated paintings. Reflections and highlights caused by foreground objects (e.g., lights) are also present in empty scene images, which could hinder the ability of our approach when generalizing to real-life empty scene images that are not lit up.

References

1. 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
2. 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) 1, 4
3. 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) 11
4. 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) 11

5. 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
6. 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) 8
7. Nathan Silberman, Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from RGBD images. In: Proceedings of the European Conference on Computer Vision (2012) 9
8. 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) 3, 4
9. 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, 8
10. Schönfeld, E., Sushko, V., Zhang, D., Gall, J., Schiele, B., Khoreva, A.: You only need adversarial supervision for semantic image synthesis. In: International Conference on Learning Representations (2021) 1
11. 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) 4
12. Zhao, S., Liu, Z., Lin, J., Zhu, J.Y., Han, S.: Differentiable augmentation for data-efficient GAN training. In: Proceedings of the Conference on Neural Information Processing Systems (2020) 10