DiffInDScene: Diffusion-based High-Quality 3D Indoor Scene Generation

Supplementary Material

6. Video Demonstration

To gain a more comprehensive understanding of our method for generating the indoor scene, we kindly invite you to watch the attached video. The video demonstrates an example of the coarse-to-fine generation process, and the the post-processing of texturing using DreamSpace [51]. Furthermore, to provide a more detailed and complete perspective on the inner scene structures, a random walk is conducted within the generated scene.

7. Implementation Details

7.1. Dataset and Preprocessing

Indoor Scene Generation from Scratch. 3D-FRONT [11] provides professionally designed layouts and a large number of rooms populated by high-quality 3D models. However, when organizing the mesh models to a complete scene, the meshes may intersect with each other. Additionally, most of them are not watertight meshes. These factors lead to erroneous Truncated Signed Distance Function (TSDF) volumes. In such cases, the meshes retrieved from TSDF volumes contains lots of wrong connections. To address this problem, we perform a solidification and voxel remeshing on each scene mesh, using a pipeline of modifiers from Blender with a voxel size of 0.02m. All meshes are saved as triangular format. After the watertight meshes are obtained, we derive the SDF volumes by using a open-source software SDFGen [1], with a resolution of 0.04m. Then the SDF volumes are truncated to TSDF by a maximum distance of 0.12m.

Refinement on the Reconstruction from Multi-view Stereo(MVS). We use the official train / validation / test split of ScanNet(v2) dataset, including 1201 / 312 / 100 scenes respectively. For there is no TSDF ground truth provided in this dataset, we adopt a TSDF fusion method like [19] to produce the ground truth as NeuralRecon does. We only use TSDF data without any other data type such as images in the whole training/testing process. To compare the reconstruction results with pretrained NeuralRecon, the grid size of TSDF volume is set to 0.04m, and the truncation distance is set to 0.12m. The default value of the TSDF volume is 1.0.

In the training process, a random volume crop of $96 \times 96 \times 96$ is used as data augmentation, where a random rotation between $[0, 2\pi]$ and a random translation is performed before cropping. To ensure that the sampling crop contains sufficient occupied voxels, the translation is limited in the bounding box of global occupied region, and the entire cropped volume should be within the boundary of this region.

7.2. Sparse Diffusion Model

Network Structure. `TorchSparse` [46] is used to implement the UNet structure of our network for noise prediction. A group normalization(32 groups) and a SiLU activation are used successively before any layer of sparse convolution. The network strctures used in difference stages of our cascacded diffusion are shown in Fig. 9, where SparseRes and Spatial Transformer are key components of our implementation as shown in Fig. 10.

Training & Inference Settings. The network parameters are randomly initialized in training process, and we use the Adam optimizer with a learning rate of 1.0×10^{-4}.

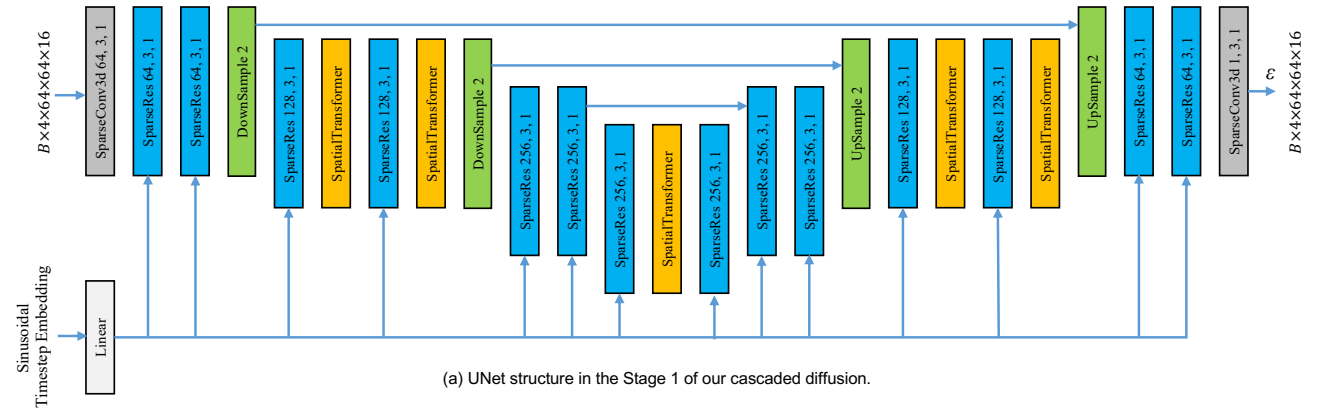
As for the diffusion framework, the `DDIMScheduler` in the open-source diffusers [47] is developed as our codebase. Following [6] and [38], we adopt the $\alpha-$conditioning to stabilize training, and enable the parameter tuning over the noise schedule and the timesteps during inference stage. More concretely, the cumulative product of α_t namely $\bar{\alpha}_t$ is used as a substitute of the timestep t as time embedding in most existing works. In Section 4.1, we use a cosine noise schedule with 2000 timesteps during training, and the same noise schedule is used with 200 time-steps during inference within the DDIM framework. In Section 4.3, we use a linear noise schedule of $(1e-6, 0.01)$ with 2000 timesteps during training, and the same noise schedule is used with 100 timesteps during inference within the DDIM framework. The clip range for TSDF sampling is $[-3.0, 3.0]$.

7.3. PatchVQGAN for Learning the Occupancy Embedding

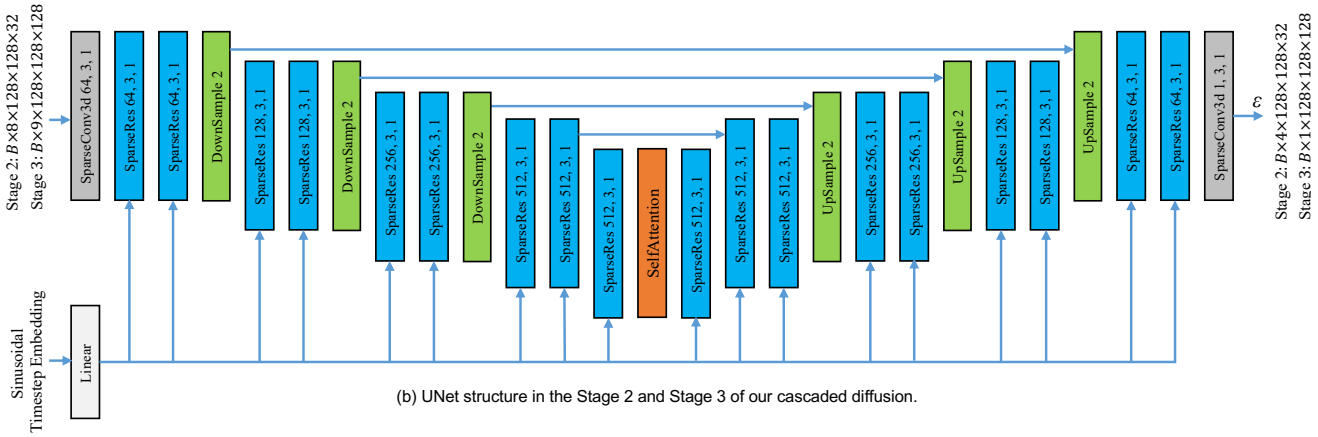
Network Structure. The network structure of PatchVQGAN described in Section 3.3 is shown in Fig. 11. The multi-scale encoding and decoding processes are slightly coupled with each other, while we simplify the description of the whole model for better understanding in Section 3.3. The encoder and decoder are implemented hierarchically as "Encoder 1", "Encoder 2", "Decoder 1", and "Decoder 2" as shown in Fig. 11 (b)-(e). The multi-layer feed-forward discriminator is omitted here.

Different from [9], we use quantizers with Gumbel-Softmax [20] which enables a differentiable discrete sampling. The size of codebook is 8192, with the embedding dimension of 4 as commonly adopted in [9][37].

Training & Inference Settings. The hyper parameters in Eq. (11) are initially set to $\lambda_1 = 1.0$, $\lambda_2 = 0.2$. Additionally, a dynamic weight adapting strategy as [9] is employed

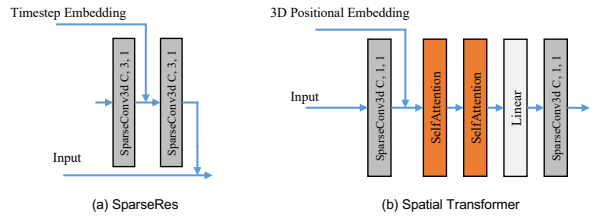


(a) UNet structure in the Stage 1 of our cascaded diffusion.



(b) UNet structure in the Stage 2 and Stage 3 of our cascaded diffusion.

Figure 9. Noise prediction networks in our cascaded diffusion. In Stage 1, we use multiple Spatial Transformers as (a) shows. In Stage 2 and Stage 3, we use same network structure as (b), with only one attention layer in the middle of network.



(a) SparseRes (b) Spatial Transformer

Figure 10. Sparse units widely used in our implementation of noise prediction network in sparse diffusion.

to control λ_2. The network parameters are randomly initialized with normal distribution in training process, and we use the Adam optimizer with a learning rate of 1.0×10^{-5}.

7.4. Local Fusion of Diffusion

The average fusion method mentioned in Section 3.4 is defined as follows.

Average Fusion. Suppose $x_t^k(\mathbf{p}) \sim \mathcal{N}(\mu_t^k(\mathbf{p}), \Sigma_t^k(\mathbf{p}))$, we have:

$$x_t(\mathbf{p}) \sim \mathcal{N}(\frac{1}{|\mathcal{G}(\mathbf{p})|} \sum_{k \in \mathcal{G}(\mathbf{p})} \mu_t^k(\mathbf{p}), \frac{1}{|\mathcal{G}(\mathbf{p})|^2} \sum_{k \in \mathcal{G}(\mathbf{p})} \Sigma_t^k(\mathbf{p})). \tag{14}$$

The rapidly decreasing variance impacts generation diversity and quality. We, therefore, propose a stochastic TSDF fusion algorithm.

7.5. User Study

We conduct two user studies on meshes from generation and reconstruction refinement in Section 4.1 and 4.3, which are slightly different.

Generation. We use same metric as Text2Room [17]: Completeness and Perceptual. In every page of the survey, the users scores one scene from one method by 1-5 points on these 2 metrics. Then we take an average score on each method.

Reconstruction Refinement. We employ more metrics here, including details, completeness, plane quality, and

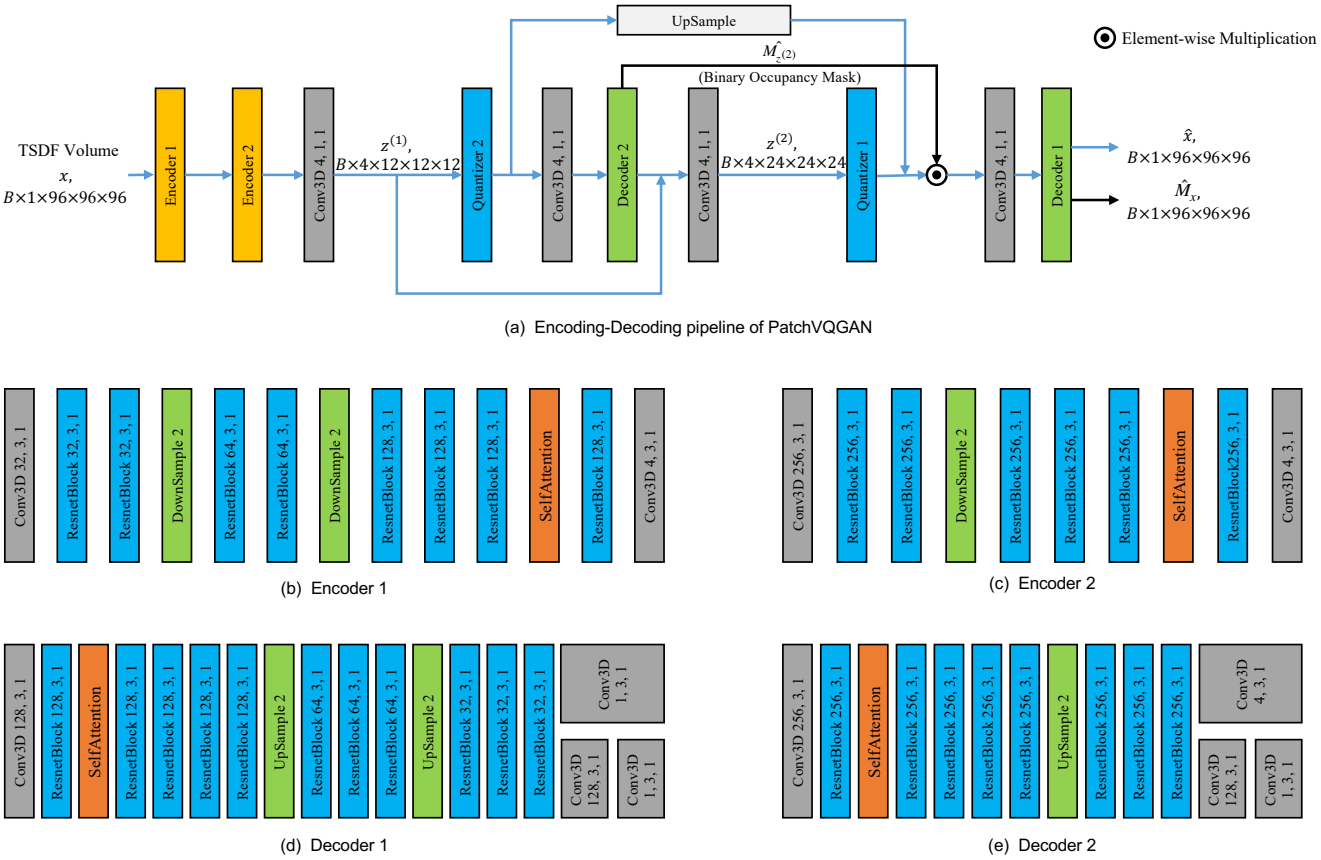


Figure 11. Network structure of PatchVQGAN.

edge quality. To save the time of the users, we use ranking rather than scoring for each scene. The feedback score S_i for the i-th scene is computed as

$$S_i = \frac{1}{d_i} \sum_{j=1}^{d_i} s(r_{i,j}), \tag{15}$$

where $r_{i,j} \in 1, 2, 3, 4$ represents the ranking given by the j-th user for the i-th scene. The function $s(r) = 4 - r$ converts the ranking into a score, with the r-th rank worth $4 - r$ score. d_i is the total number of valid feedbacks for the i-th scene. By summing up the scores across all scenes, we obtain the total score

$$S = \sum_{i=1}^{N} S_i \tag{16}$$

8. More Results on Scene Generation

We provide more scene generation samples as shown in Fig. 12 - Fig. 14.

Fig. 12 is an additional comparison between our method and Text2Room [17]. Since the Poisson [21] reconstruction can produce better results than pure Text2Room, we only show the results of "Text2Room + Poisson". Fig. 13 and Fig. 14 are generated scene samples of our method.

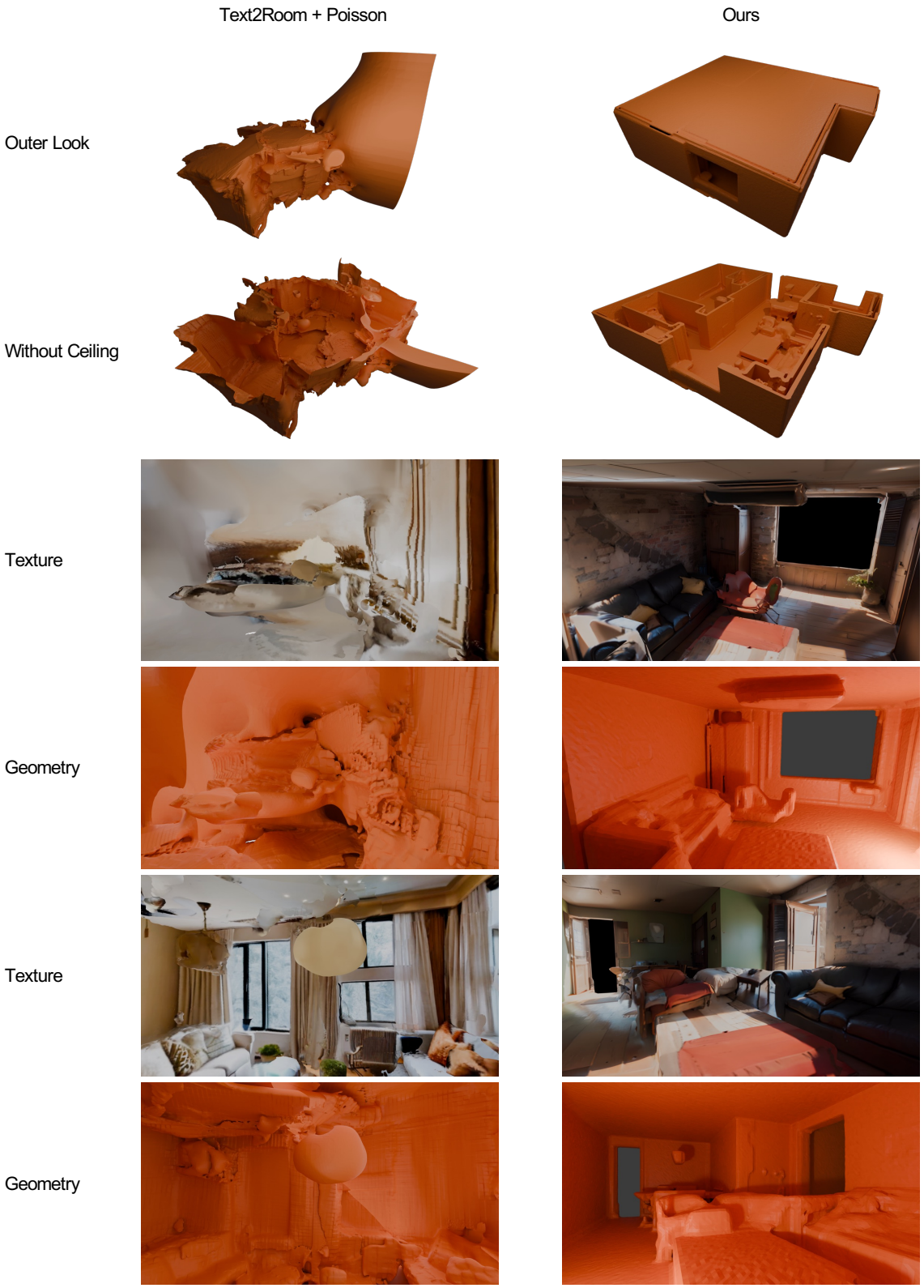


Figure 12. Comparison of Text2Room and our approach in larger views. As previous Fig. 5 shows, Poisson reconstruction significantly improves the performance of pure TextRoom, so that here we only demonstrate the results of Text2Room [17] + Poisson [21]. The textures of our results are produced by DreamSpace [51] as a post-processing of scene geometry generation.

Figure 13. More generation samples in columns.

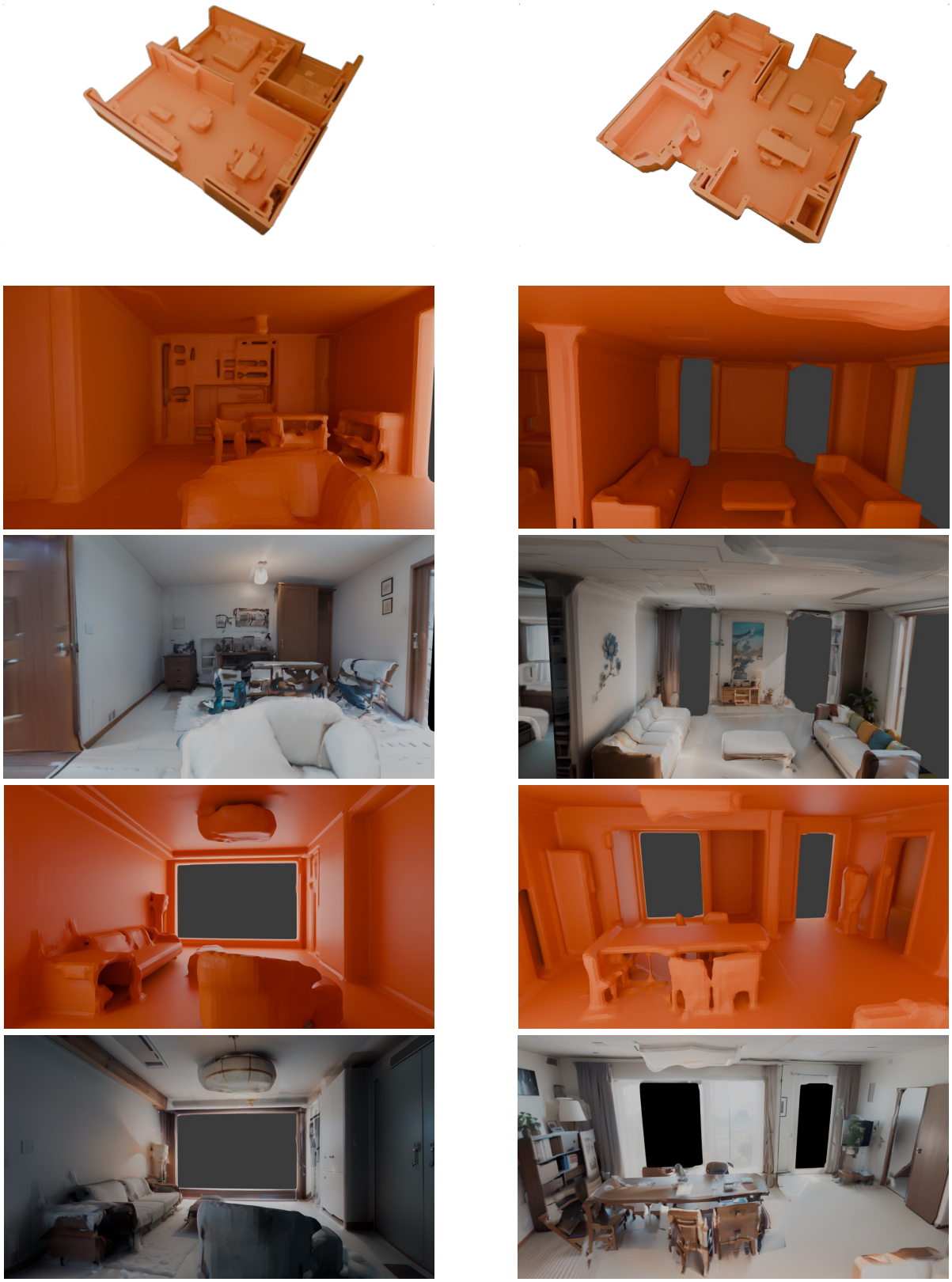


Figure 14. More generation samples in columns.