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 video on our project page: https://akirahero.github.io/diffindscene/. The video demonstrates an example of the coarse-to-fine generation process, and the the post-processing of texturing using DreamSpace [14]. 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 [4] 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 [6] 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` [12] 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}.

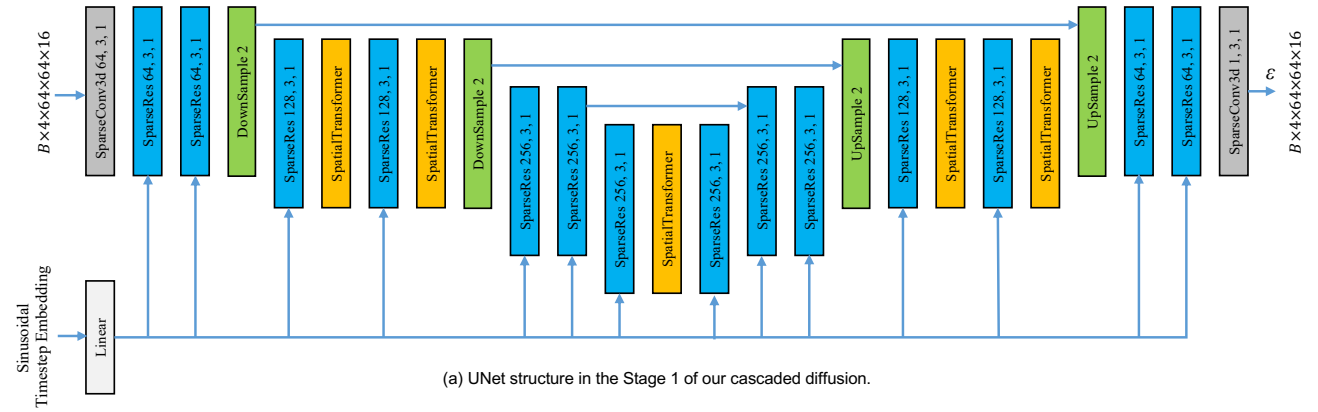
As for the diffusion framework, the `DDIMScheduler` in the open-source diffusers [13] is developed as our codebase. Following [2] and [10], we adopt the $\alpha-$conditioning to stabilize training, and enable the parameter tuning over the noise schedule and the timesteps during inference stage. More 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

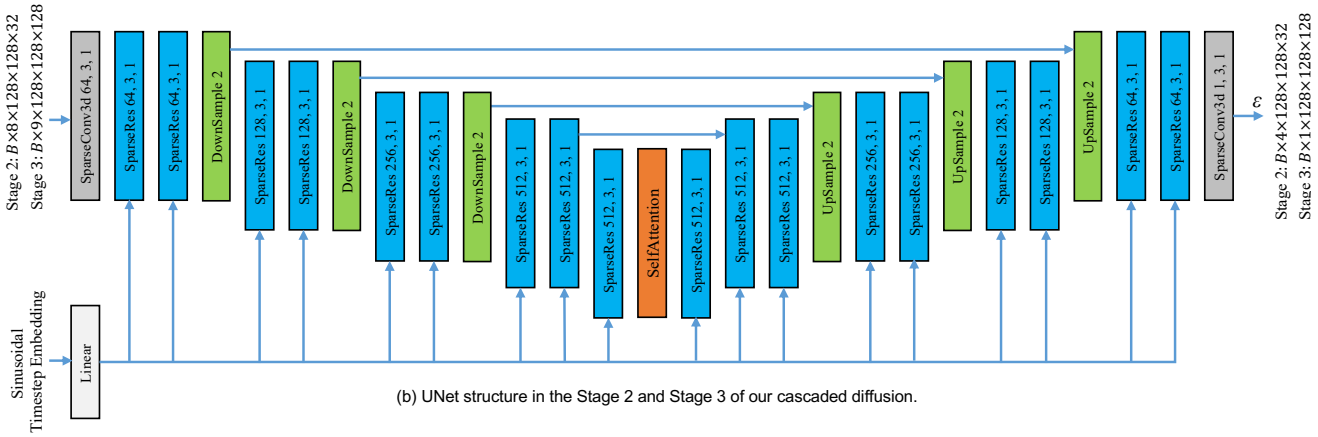
Network Structure. The network structure of PatchVQGAN described in Section 3.3 is shown in Fig. 11. The multi-scale encoding and decoding processes are slightly coupled with each other, while we simplify the description of the whole model for better understanding in Section 3.3. The encoder and decoder 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 [3], we use quantizers with Gumbel-Softmax [7] which enables a differentiable discrete sampling. The size of codebook is 8192, with the embedding dimension of 4 as commonly adopted in [3][9].

Training & Inference Settings. The hyper parameters in Eq. (11) are initially set to $\lambda_1 = 1.0$, $\lambda_2 = 0.2$. Addition-

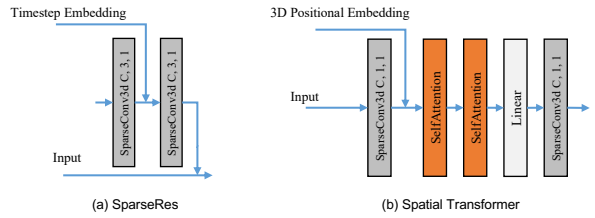


(a) UNet structure in the Stage 1 of our cascaded diffusion.



(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.

ally, a dynamic weight adapting strategy as [3] is employed 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 [5]: 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.

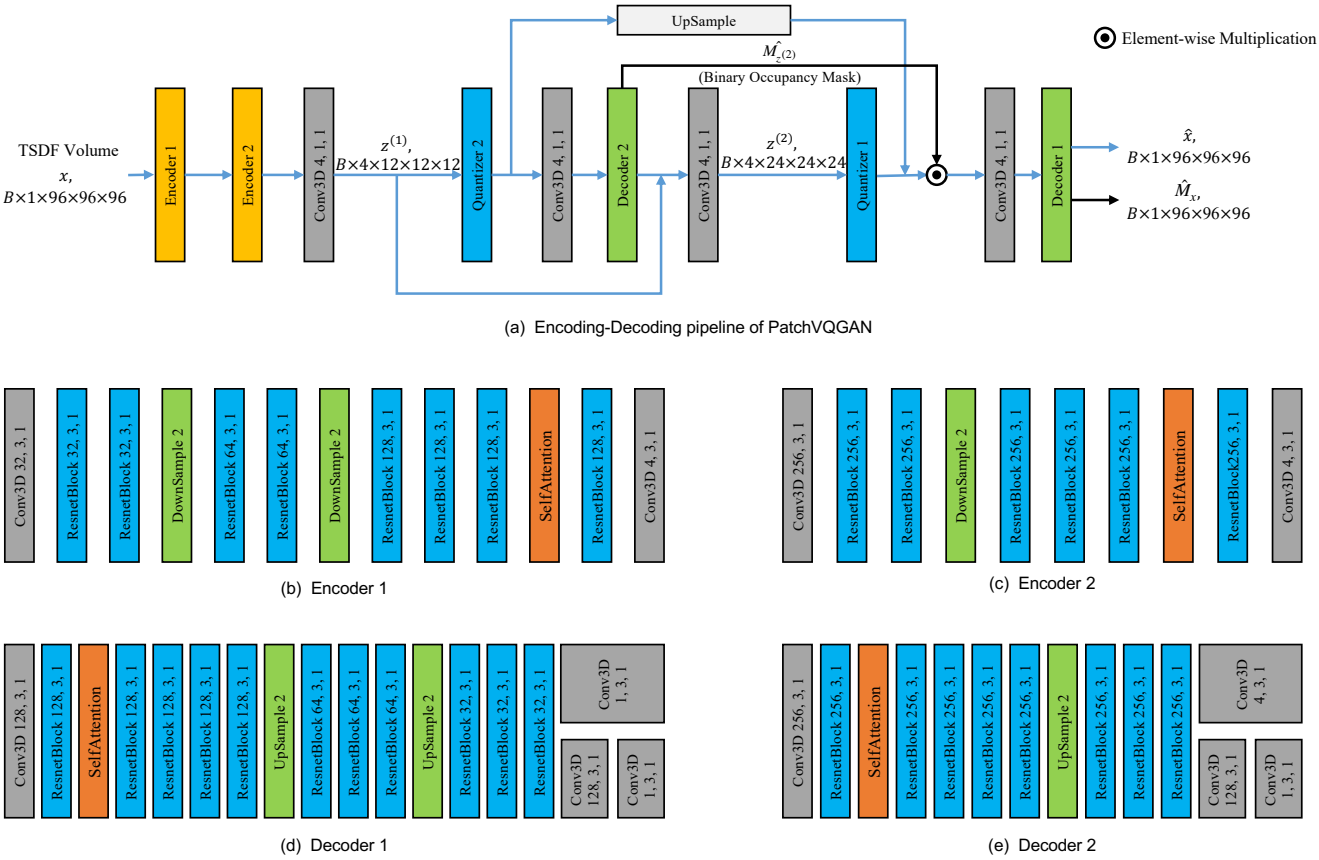


Figure 11. Network structure of PatchVQGAN.

Reconstruction Refinement. We employ more metrics here, including details, completeness, plane quality, and 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 Unconditional 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 [5]. Since the Poisson [8] 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.

9. Conditional Generation: Sketch-to-Scene

We extend our model to accept bird-eye-view sketch image as condition. We concatenate its VAE encoding to the input, and incorporate it to compute cross-attention within all "SpatialTransformers" in Fig. 9. The sketch-to-scene results are as 15 shows, and the generated scene geometry is consistent with the given condition.

Sketch Data. The sketch data is produced by cutting through the middle of 3D occupancy along the up axis. The cutting height is randomly sampled during training. As shown in Fig. 15 (a), the black line actually consists of the occupied voxels.

Sketch Encoding. To capture high-level geometry information in the sketch image, the sketch images are first encoded by a variational autoencoder (VAE). Since most space in the sketch is blank, we add more weight on the informational

black areas in the training process. The sketch images are encoded to 16-channel embedding, with same resolution as the input of the first stage of cascaded diffusion in Fig. 9.

Condition Binding. Because the geometry is mainly decided by the first stage of cascaded diffusion, we only need to insert the condition embedding into this stage. In our implementation, the sketch embedding is bound to both the input and the attention units. First, we concatenate the sketch embedding with $z_T^{(1)}$ as input to the noise prediction network as Fig. 9 (a). Second, all 3D SpatialTransformer units in Fig. 9 (a) are substituted by 2D SpatialTransformer to incorporate this bird-eye-view embedding to cross-attention. To adapt to the 2D SpatialTransformer, the 3D data from the previous blocks is transformed to bird-eye-view 2D data by using a light-weight convolutional network along the up-axis, and then the output from 2D SpatialTransformer is up-sampled back to 3D data.

10. Complementary Evaluation on the Refinement of MVS

We compute the reconstruction metrics "Accuracy" and "Completeness" for Section 4.3, with threshold of 0.05m as described in [11]. The results are listed in Table 6. Our method leads to obvious improvements over NerualRecon results. For Laplacian smoothing and isotropic remeshing, we use the implementation provided in MeshLab.

Table 6. Evaluation on MVS refinement using classical reconstruction metrics.

	Accuracy↑	Completeness↑
NeuralRecon	0.412	0.569
NeuralRecon + Laplacian Smoothing	0.415	0.544
NeuralRecon + Isotropic Remeshing	0.385	0.494
NeuralRecon + Ours	**0.432**	**0.593**

11. Discussion on the Limitations and Failure Cases

Our method generates the entire scene end-to-end without distinguishing different contents. This may lead to unnecessary memory consumption, such as when a planar wall occupies more voxels but with fewer geometry details compared to a table. To address this, we will use a volume with adaptive resolution, leveraging attention mechanisms to determine where to allocate more voxels. Additionally, we will incorporate more conditional control from the perspective of the room designer.

In terms of failure cases, unsatisfying local geometry more often appears in large scenes. Increasing the training data is a straightforward solution, but acquiring a sufficient amount of 3D data is challenging compared to image tasks. To address this, we will train a detector to identify such areas. Subsequently, we can erase these areas and perform a completion. We plan to explore this approach in our future work.

References

[1] Christopher Batty. Signed distance field generator for triangle meshes, 2015. 1

[2] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. Wavegrad: Estimating gradients for waveform generation. *arXiv preprint arXiv:2009.00713*, 2020. 1

[3] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021. 1, 2

[4] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021. 1

[5] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. *arXiv preprint arXiv:2303.11989*, 2023. 2, 3, 6

[6] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011. 1

[7] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 1

[8] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, page 0, 2006. 3, 6

[9] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1

[10] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022. 1

[11] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, pages 519–528. IEEE, 2006. 4

[12] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. Torchsparse: Efficient point cloud inference engine. *Proceedings of Machine Learning and Systems*, 4:302–315, 2022. 1

[13] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022. 1

[14] Bangbang Yang, Wenqi Dong, Lin Ma, Wenbo Hu, Xiao Liu, Zhaopeng Cui, and Yuewen Ma. Dreamspace: Dreaming your room space with text-driven panoramic texture propagation. *arXiv preprint arXiv:2310.13119*, 2023. 1, 6

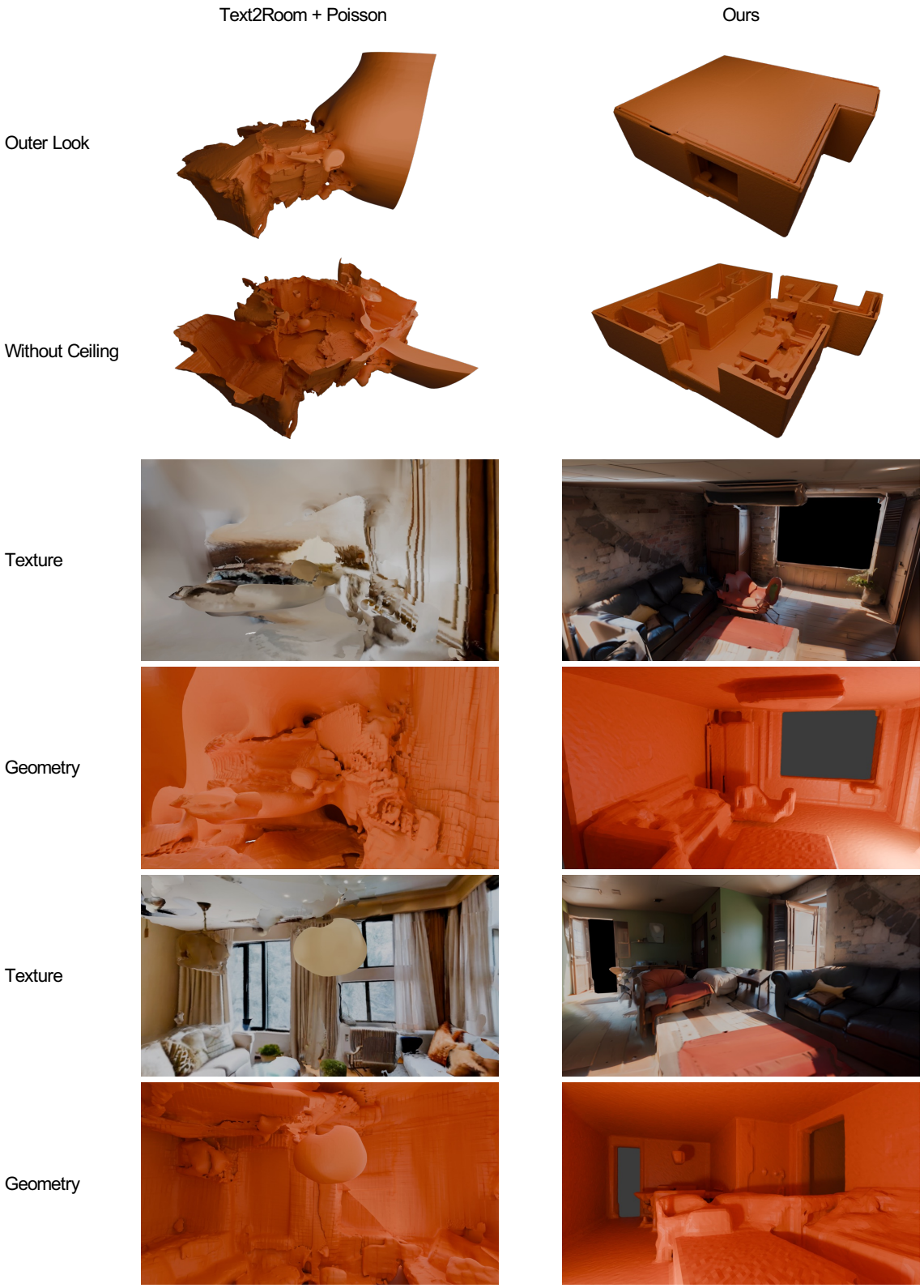


Figure 12. Comparison of Text2Room and our approach in larger views. As previous Fig. 5 shows, Poisson reconstruction significantly improves the performance of pure TextRoom, so that here we only demonstrate the results of Text2Room [5] + Poisson [8]. The textures of our results are produced by DreamSpace [14] as a post-processing of scene geometry generation.

Figure 13. More generation samples in columns.

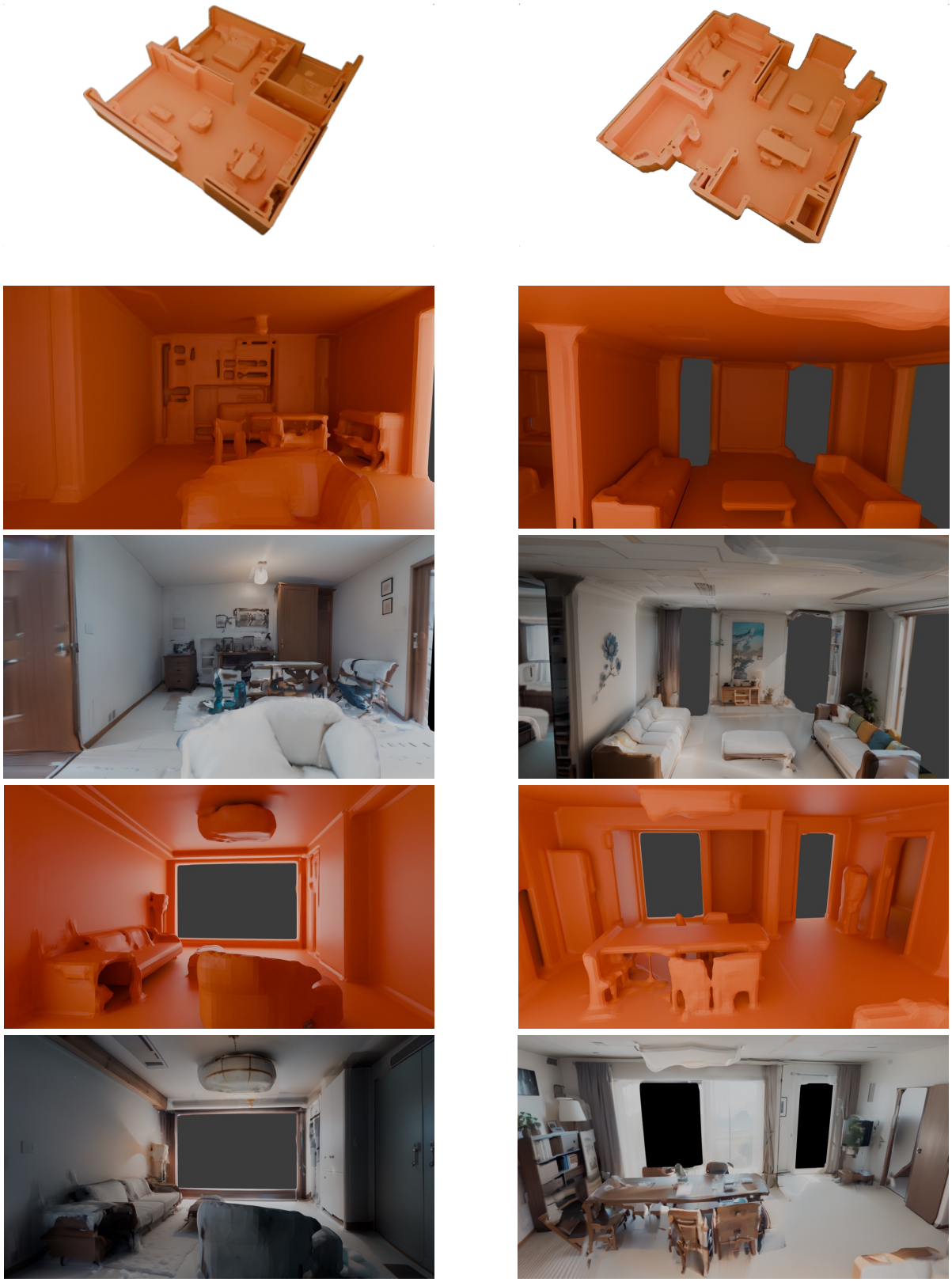
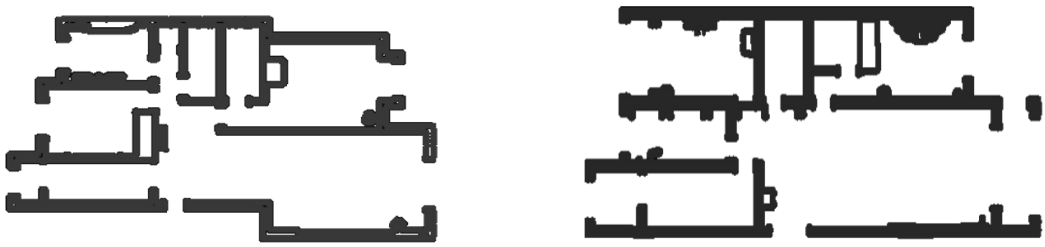


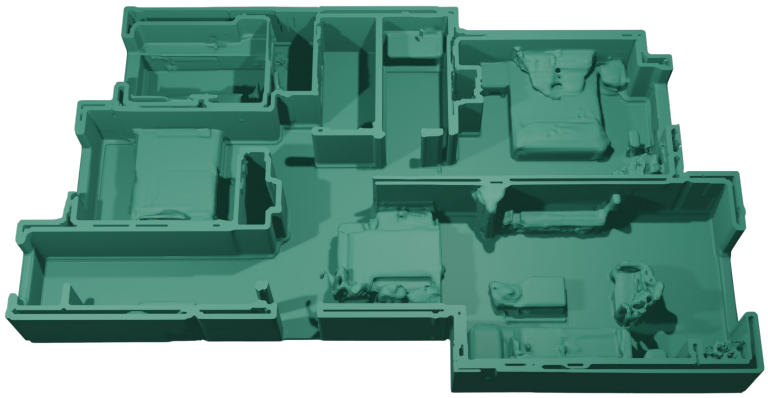
Figure 14. More generation samples in columns.



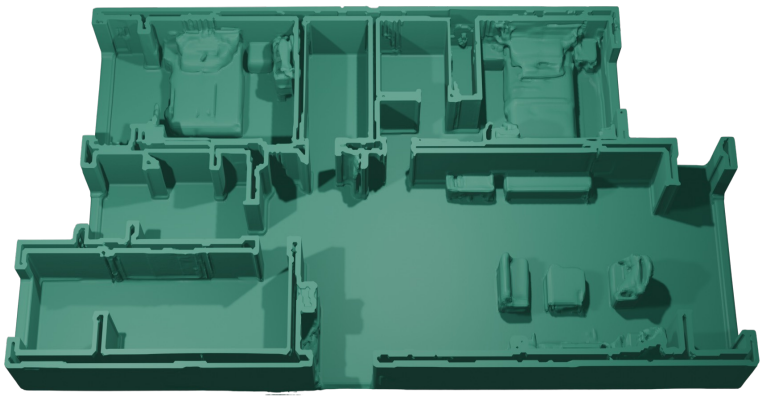
Bird-eye-view Sketch (A)

Bird-eye-view Sketch (B)

(a) Binary sketch images as condition input of generation.



Generated Sample (A)



Generated Sample (B)

(b) The corresponding generated scenes of (a).

Figure 15. Samples of Sketch-to-Scene generation.