HouseCrafter: Lifting Floorplans to 3D Scenes with 2D Diffusion Models

Supplementary Material

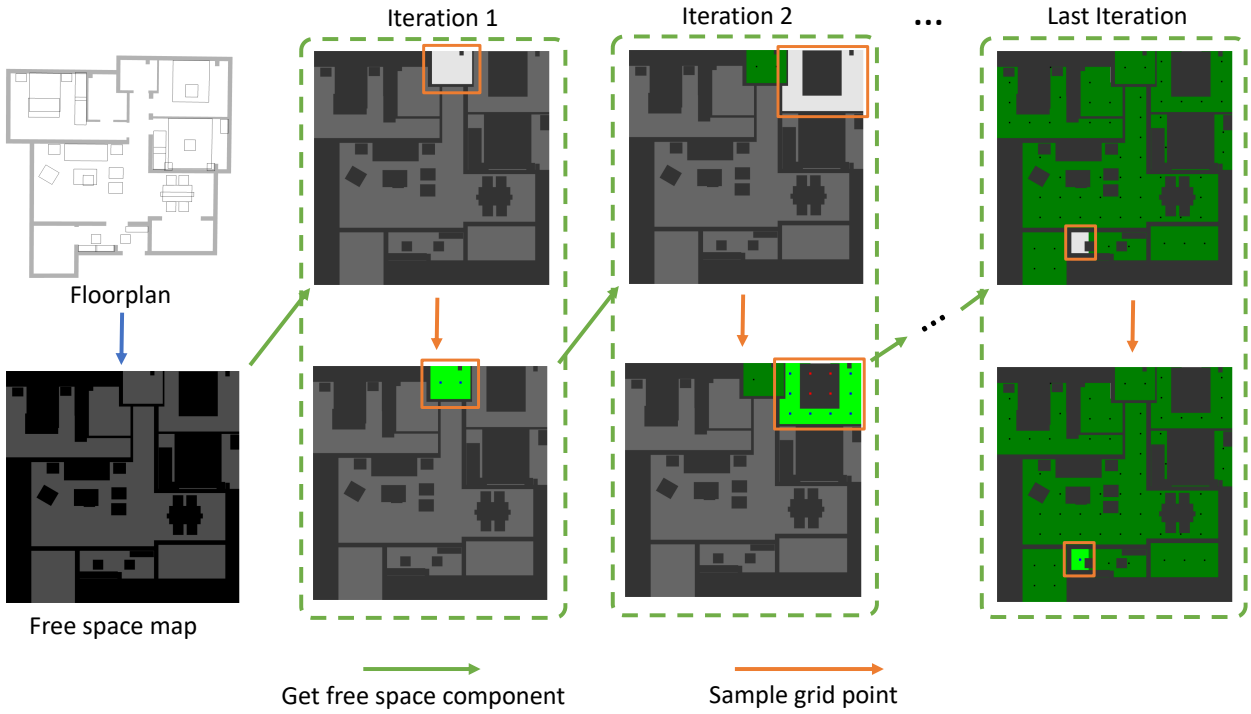

Figure 1. **Camera location sampling.** From the floorplan, we first obtain a binary free space map (black means occupied) and then iterate over the free space to sample camera locations. In each iteration, we select a free space component (highlighted in white) then sample grid points within the component's bounding box. The invalid points (occupied, marked with red) are discarded and the rest valid points (unoccupied, marked with green) are stored as possible camera locations. The loop terminates when the all the free space is processed or the remaining area is smaller than a threshold.

This supplementary document is structured as follows:

1. Camera Location Sampling and Graph Construction

The camera location sampling procedure is illustrated in Fig. 1, where we first obtain a binary free space map from the input floorplan. An iterative procedure is then applied over the free unoccupied space to sample camera locations until all the free space is processed. Based on the sampled camera locations, the graph construction is illustrated in Fig. 2, where we first construct subgraphs within each room separately. They are finally connected to form a complete graph of the entire scene.

2. Autoregressive RGB-D Image Generation via Graph Traversal

Given the location graph, the reference and novel views are selected by traversing the graph. The procedure is described in Alg. 1. To control the number of views in each generation

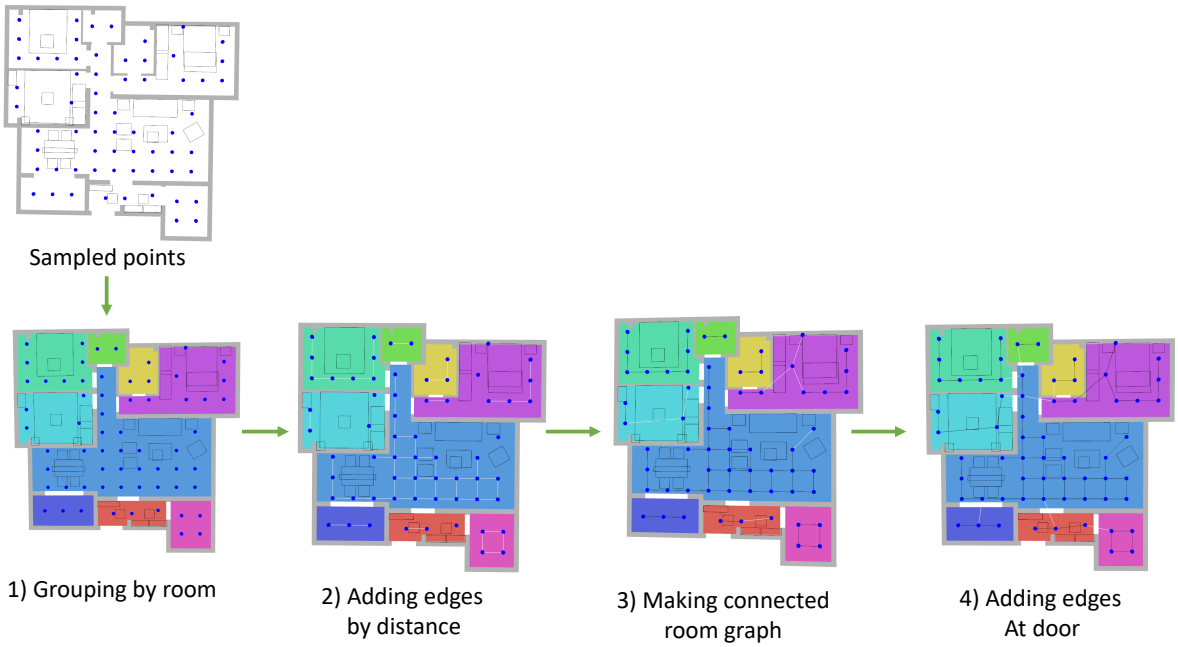

Figure 2. **Location graph construction.** From left to right: (1) Given the sampled locations, we first group the locations by room types (*e.g.*, kitchen, living room). Next, we construct a subgraph in each room in two steps: (2) adding edges between two nodes if their distances are smaller than a threshold and (3) connecting the rest isolated locations to its neighboring nodes in a partial subgraph. (4) In the last step, we use the door locations to connect the room subgraphs to form a complete graph for the entire scene. Specifically, for each door, we add an edge between the nearest locations across two adjacent rooms. By creating graphs at the room scale then connecting them using the door location, we avoid making undesirable edges where two locations are close but do not have overlap due to the wall. New edges of each step (2,3,4) are highlighted in white.

step, we use two hyper parameters δ_r and δ_n, which are the hop distance thresholds with respect to the current nodes for the reference and novel views, respectively. When visiting a location v whose images have not been generated, we choose locations that are within δ_r hops from v and have images already generated as reference views. The novel views include v and those that have not been generated and within δ_n hops from v.

3. Details of Floorplan Conditioning

For a novel view with the latent feature $\mathbf{z}_j^n \in \mathbb{R}^{C\times H\times W}$ (where C is the feature dimension and $H \times W$ the spatial dimensions), we obtain the floorplan information $\mathbf{l}_j \in \mathbb{R}^{M\times C\times H\times W}$ at the (latent) pixel-level by casting rays through the pixels and encoding semantic and geometric information at every intersection point between the projected ray and floorplan components.

Subsequently, we use cross-attention at the ray-level where each pixel feature the in $\mathbf{z}_j^n$ is the query and the floorplan features along the ray are the keys and values, meaning the attention for each ray is performed independently. To illustrate the operation, we add the batch dimension B and use `einops` [8] notation:

$$\begin{aligned}
\mathbf{z}_j^n &\leftarrow \text{rearrange}(\mathbf{z}_j^n,\ \text{B C H W} \rightarrow (\text{B H W})\ 1\ \text{C})\\
\mathbf{l}_j &\leftarrow \text{rearrange}(\mathbf{l}_j, \text{B N C H W} \rightarrow (\text{B H W})\ \text{N C})\\
\mathbf{z}_j^n &\leftarrow \text{MHA}(q=\mathbf{z}_j^n, k=\mathbf{l}_j, v=\mathbf{l}_j)\\
\mathbf{z}_j^n &\leftarrow \text{rearrange}(\mathbf{z}_j^n, (\text{B H W})\ 1\ \text{C} \rightarrow\ \text{B C H W}),
\end{aligned}$$

where `MHA()` is the multihead attention. The floorplan information is incorporated in the first block of each feature level in the UNet blocks of the image diffusion model. Since each level operates at a different resolution, this process effectively injects the encoded floorplan at multiple scales.

4. Details of Evaluation

4.1. RGB-D Image Consistency Evaluation

In this section, we describe the overlap region estimation for a pair of posed RGB-D images. Then we provide the details of the evaluation metrics.

Overlap Region Estimation. Given a pair of RGB-D images, (I_1, D_1) and (I_2, D_2), we warp images I_1, D_1 of the first view to the second view according to the relative camera pose between them, obtaining $I_{1\to2}, D_{1\to2}$. The

Algorithm 1 Autoregressive RGB-D image generation via graph traversal

Input:
$G(V, E)$: location graph
δ_n: Hop distance threshold for novel views
δ_r: Hop distance threshold for reference views
L: Floorplan

$X \leftarrow \emptyset$ ▷ Initialize the set of visited locations.
for v in $BFS(G)$ **do** ▷ traverse graph via breadth-first search.
  if $v \notin X$ **then**
    $X_r \leftarrow X \cap N(v, G, \delta_r)$ ▷ Get reference locations. $N(v, G, d)$: nodes within d hop from v
    $X_n \leftarrow \{v\} \cup N(v, G, \delta_n) \backslash X$ ▷ Get novel locations.
    if $X_n \neq \emptyset$ **then**
      $Generate\_RGBD\_Images(X_r, X_n, L)$ ▷ Generate views at locations. For the first loop, X_r can be empty.
      $X \leftarrow X \cup X_n$
    end if
  end if
end for

overlap/correspondence region $\mathcal{M}$ is defined as follows such that the warped depth $D_{1\to2}$ match perfectly with D_2,

$$\mathcal{M} := \mathbb{1}(D_{1\to2} = D_2), \tag{1}$$

where $\mathbb{1}()$ is the indicator function. To account for the potential inconsistency of the generated images, we introduce a tolerance threshold τ to estimate the overlap region,

$$\hat{\mathcal{M}} := \mathbb{1}(|D_{1\to2} - D_2| < \tau). \tag{2}$$

Given the estimated overlap region $\hat{\mathcal{M}}$, the consistency metrics are computed for depth image pair $(D_{1\to2}, D_2)$ and the RGB image pair $(I_{1\to2}, I_2)$.

RGB Consistency Metrics. Given the RGB image pair $(I_{1\to2}, I_2)$ and the overlap region $\hat{\mathcal{M}}$, we compute the peak signal-to-noise ratio PSNR for color consistency,

$$PSNR := 20 \cdot \log_{10}(255) - 10 \cdot \log_{10}(MSE), \tag{3}$$

where

$$MSE := \frac{1}{\sum_k \hat{\mathcal{M}}(k)} \sum_k \hat{\mathcal{M}}(k) \cdot [I_{1\to2}(k) - I_2(k)]^2. \tag{4}$$

Here k is pixel index. Note that we omit averaging over the color channel to simplify the notation.

Depth Consistency Metrics. Given the depth image pair $(D_{1\to2}, D_2)$ and the overlap region $\hat{\mathcal{M}}$, we compute Absolute Mean Relative Error (*AbsRel*) and percentage of pixel inliers δ_i for depth consistency. *AbsRel* is calculated as:

$$AbsRel := \frac{1}{\sum_k \hat{\mathcal{M}}(k)} \sum_k \hat{\mathcal{M}}(k) \frac{|D_{1\to2}(k) - D_2(k)|}{D_2(k)}. \tag{5}$$

The percentage of pixel inliers δ_i is calculated as:

$$\delta_i := \frac{1}{\sum_k \hat{\mathcal{M}}(k)} \sum_k \hat{\mathcal{M}}(k) \cdot \tag{6}$$

$$\mathbb{1}\left(\max\left(\frac{D_{1\to2}(k)}{D_2(k)}, \frac{D_2(k)}{D_{1\to2}(k)}\right) < 1.25^i\right). \tag{7}$$

We choose $i = 0.5$ to have a tight threshold.

4.2. Floorplan Consistency Evaluation

The floorplan evaluation protocol is the "inverse" of HouseCrater where we predict the top-down 2D bounding boxes of objects in the generated scene and compare their consistency with the provided floorplan. Specifically, the detected 2D bounding boxes are compared with 2D boxes from the given floorplan using mean Average Precision at the intersection-over-union threshold of 0.25 (mAP@25). Here, we use ODIN [4], a 3D instance segmentation method that takes multi-view posed RGB-D images as input and predicts instance segmentation of the point cloud accumulated from input images. Then, top-down 2D boxes are extracted from the segmented instances. As a scene may have up to 2000 images based on its size, we cannot pass all the images to ODIN at once. Instead, these images are grouped by room types and we do segmentation per room. This strategy does not affect the evaluation results since an object in the scene do not span in more than one room. We finetune ODIN on 3D-Front dataset to make the segmentation results more reliable since both HouseCrafter and CC3D [1] are trained on this dataset.

4.3. User Study

We conduct a user study to evaluate the results produced by Text2Room, CC3D, BlockFusion, and our method. In the

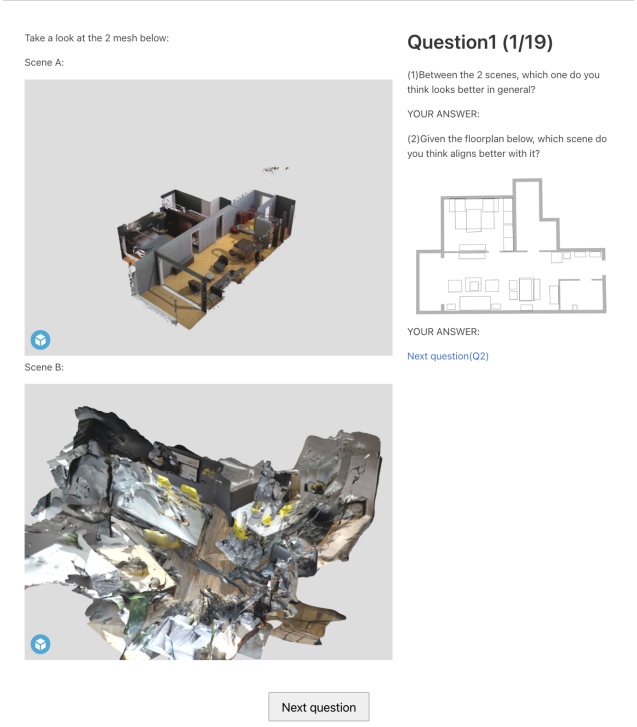

Figure 3. **User Study Interface.** We show users 2 meshes at a time, one is produced by our model and the other is produced by a baseline method. We then ask users to choose one mesh that appears "better looking in general", and one mesh that appears "align better" with the given floorplan.

study, we ask 12 participants to rate the results in a pair-wise manner. Specifically, we present the participants with two meshes at a time and ask them to choose: i) the one that appears more visually appealing; and ii) the one that is more coherent with the provided floorplan. The interface is shown in Fig. 3. We compare the entire house we generated to the results by BlockFusion. For text2room, since it does not take floorplan as a form of guidance, we do not report participants' answers to the second question if one of the meshes is produced by it.

However, we still ask the question to prevent unconscious bias. Given that CC3D generates results at the room level rather than for entire houses, we clip our results and floorplan to the specific room CC3D produces when making comparisons.

5. Implementation Details

We initialize our novel view RGB-D image generation model from `StableDiffusion v1.5` [9]. For the first layer of the UNet, we duplicate the pre-trained weights and divide the weights by two to accommodate the depth's latent and to reduce the change of the output scale. For the last layer of the UNet, we only duplicate the pre-trained weights for the RGB and depth output, respctively. The model is trained for $15,000$ iterations in 2 days with an effective batch size of 256 (4 samples per GPU $\times 8$ GPUs $\times 8$ gradient accumulation steps). Each data sample contains 3 reference views and 3 novel views with the resolution of 256. We use Adam optimizer with a learning rate of 10^{-4}. All training is conducted on a machine with 8 A6000 48GB GPUs.

Table 1. **Effectiveness of fine-tuning the novel view RGB-D generation model on noisy data for autoregressive inference.**

Variant	Autoreg.	FT	RGB Metrics			Depth Metrics	
		w/ noise	FID↓	IS↑	PSNR↑	AbsRel↓	$\delta_{0.5}$↑
⑤	✗	✗	16.70	4.74	25.0	7.06	92.43
⑤	✓	✗	34.98	4.24	19.64	12.74	86.35
⑥	✓	✓	22.30	4.37	21.76	11.09	88.13

However, during inference, instead of using ground-truth reference images, we condition the model on its previously generated outputs in an autoregressive manner. Since novel view images are inherently imperfect, small discrepancies in generated images can compound over multiple iterations, leading to error accumulation and progressive degradation in image quality.

To mitigate this issue, we introduce a noisy reference finetuning strategy to bridge the domain gap between training and inference. To this end, we construct a noisy dataset by recording the model's generated outputs from the previous epoch. During finetuning, a subset of reference images is sampled from this noisy dataset instead of the ground-truth images. Note the model is still trained to produce outputs that match the ground-truth novel view RGB-D images. This approach improves robustness to accumulated errors and enhances the model's performance in long-horizon autoregressive generation.

Table 1 demonstrates that, compared to single-batch inference, images generated in an autoregressive manner exhibit decreased consistency and visual quality due to accumulated errors over iterations . Finetuning the model on noisy reference inputs enhances its robustness, leading to improved stability and coherence in long-horizon generation.

6. Running Time Comparison with Other Methods

We measure the total time to generate a scene on an A6000 GPU. We also provide the average number of images/blocks per scene. Note that while Text2Room, BlockFusion, and HouseCrafter produce meshes as final output, CC3D generates volumetric latent as scene representation, and requires neural rendering to get any view. Hence we follow their codebase to generate a room then render 40 images and report the total time of generation and rendering. As shown in

Table 2. **Running time comparison.** * denotes the number of blocks

Method	#Images/Blocks	Total time (min)
Text2Room	$217_{\pm 5}$	$50_{\pm 1}$
CC3D	40	< 1
BlockFusion	$*23_{\pm 10}$	$30_{\pm 12}$
HouseCrafter (Ours)	$1000_{\pm 400}$	$24_{\pm 10}$

Table 3. **Ablation study of generating RGB and depth images at the same time.** The results are obtained over *rendered* images from the generated scenes.

Method	Floorplan Consistency
	mAP@25↑
Monocular metric depth	30.13
Generated depth **(proposed)**	**45.77**
GT, 3D-FRONT	54.51

Table 2, CC3D is the fastest method. Among the rest, which are diffusion-based methods, our model has the smallest running time.

7. Ablation on Simultaneous RGB and Depth Generation

To further show the effectiveness of the RGB-D generation over RGB-only generation in the reconstructed scene, we do an ablation by replacing the generated depth with the estimated depth from an off-the-shelf *metric* monocular depth estimation model [7]. Specifically, in each generation batch, we use the monocular estimated depth of the reference RGB images as the reference depths to generate the novel view RGB then estimate the depth for the novel views. As the estimated monocular depth may have an incorrect scale, we use visual cues such as wall or floor to calibrate the scale when possible. The quantitative evaluation in terms of floorplan consistency of the reconstructed scene shows that generating RGB-D images together achieves better results (Table 3). Visualization of the reconstructed scenes shown in Fig. 4 further confirms the design choice of simultaneous generation of RGB and depth images.

8. Ablation Study on Floorplan Encoding

We discuss and ablate an alternative design for floorplan encoding, which does not explicitly use the geometry information. Here, each object in the scene is represented by a vector (encoded with object category and 2D bounding box). Each image token/feature at a novel view will then cross-attend to features of all the objects that are within the camera's frustum. Compared with this alternative design choice, which models *all* the candidate objects within each view, the design presented in the main paper directly considers **objects on optical rays only** and is able to model the occlusion (*e.g.*, because of walls) better, and thus reduces the number of objects to consider and simplifies the model training.

As shown in Table 4, compared to this baseline, the proposed method in the main paper has higher floorplan consistency (in terms of mAP@25) w.r.t. the input floorplan and higher quality in terms of generated RGB images, validating the effectiveness of our proposed method. This baseline design achieves better consistency for depth images. We hypothesize that this is due to extra information provided in each object's bounding box instead of the coordinate information of the intersection point. Nevertheless, considering the overall quality, we choose the design choice reported in the main paper.

9. Limitations and Future Directions

We show promising results on a challenging task to convert a 2D floorplan into a complete textured 3D scene. Here we briefly discuss the limitations of our approach and dicuss future directions.

First, we seperate the RGB-D image generation and scene reconstruction using TSDF fusion without considering their synergies. In the future, we aim to explore combining them together, where a single model is sufficient, unifying the generation and reconstruction tasks.

Second, we adopt the TSDF fusion to produce reasonably good results in fusing generated RGB-D images as explicit scene representations are desired to support interactions with the scene. However, the fused scene mesh does not always produce high-quality rendering results. We also experimented with other methods, such as SuGaR [3], which however did not yield better results. An appealing future direction would be to investigate high-quality mesh generation based on NeRF [6] and 3D Gaussian Splatting [5].

Finally, in our proposed method of injecting the floorplan guidance to the generation process, only the geometry and semantics of the object are leveraged, while the information about the object instance is omitted. We believe that instance-awareness can give better scene understanding thus generating more faithful 3D scenes to the floorplans.

10. Additional Visual Comparisons

We show additional visual comparisons with other baseline methods in Fig. 7 and Fig. 8. We can see our model produces better scene generation results. We also refer readers to the supplmentary video for more visual results.

Table 4. **Ablation studies for layout encoding.** The better results are highlighted with **bold**.

Variant	RGB Metrics				Depth Metrics				Floor. Const.
	FID ↓	IS ↑	PSNR ↑		AbsRel ↓		$\delta_{0.5}$↑		mAP@25 ↑
			R-N	N-N	R-N	N-N	R-N	N-N	
baseline	27.15	4.20	25.01	**25.27**	**4.59**	**6.89**	**96.62**	**93.23**	38.16
proposed	**16.70**	**4.74**	**25.31**	24.69	6.79	7.37	92.20	92.65	**52.26**
GT	-	-	-	-	-	-	-	-	54.51

Table 5. **Quantitative measurement of our result on ArkitScenes data** Similar to the main paper, the scene quality is measured in Inception Score(IS), Floorplan Consistency, and 3D quality

Method	Visual Quality	Floorplan Consistency	3D quality
	IS ↑	mAP@25↑	artifacts↓
Text2Room	**5.35**	-	302.5
HouseCrafter(realworld)	5.57	**49.76**	224.0

11. Generalization to Real-World Data

Our primary experiments are conducted on the synthetic 3D-FRONT dataset, which allows us to generate high-quality multi-view training data and to define generation trajectories with full control. However, extending to real-world data poses additional challenges due to noisy depth measurements and constrained, predefined camera trajectories.

To evaluate the model's generalization capabilities in real-world scenarios, we adopt the recently released Stable Virtual Camera (SEVA) framework [10] as the backbone and integrate it with our proposed depth and layout conditioning modules.

For RGBD generation, we follow the same channel expansion strategy applied in our Stable Diffusion 1.5 experiments, modifying SEVA's input and output layers accordingly. Additionally, we revise the Plücker ray encoding scheme to incorporate both depth-projected 3D positions and ray directions more effectively.

For layout conditioning, we insert our layout attention modules at the end of each ResNet block within the UNet architecture, enabling the model to incorporate high-level spatial priors during generation.

We fine-tune the resulting model on the ARKitScenes [2] dataset and present qualitative results in Fig.9 to demonstrate its performance and generalization potential in real-world settings.

12. Ablation on the number of reference and target views.

In average, we select 30 reference and 10 target views, respectively, at each location for a single house generation. Here, we analyze how the performance varies with different numbers of reference and target views. In this experiment, we re-grouped the location and views, so that as the reference and target views in each location grow, the locations we need to traverse decrease. As can be seen in the Fig.10 above, the generation quality improves when more reference and target views are involved in a location (thus fewer locations) and starts saturating at a certain threshold. Due to the GPU memory constraints, we can not use significantly large number of reference and target views.

References

[1] Sherwin Bahmani, Jeong Joon Park, Despoina Paschalidou, Xingguang Yan, Gordon Wetzstein, Leonidas Guibas, and Andrea Tagliasacchi. Cc3d: Layout-conditioned generation of compositional 3d scenes. In *ICCV*, 2023. 3

[2] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. ARKitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile RGB-d data. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 6

[3] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *CVPR*, 2024. 5

[4] Ayush Jain, Pushkal Katara, Nikolaos Gkanatsios, Adam W Harley, Gabriel Sarch, Kriti Aggarwal, Vishrav Chaudhary, and Katerina Fragkiadaki. Odin: A single model for 2d and 3d perception. In *CVPR*, 2024. 3

[5] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 5

[6] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 5

[7] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *CVPR*, 2024. 5

[8] Alex Rogozhnikov. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *ICLR*, 2022. 2

[9] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 4

[10] Jensen (Jinghao) Zhou, Hang Gao, Vikram Voleti, Aaryaman Vasishta, Chun-Han Yao, Mark Boss, Philip Torr, Christian Rupprecht, and Varun Jampani. Stable virtual camera: Generative view synthesis with diffusion models. *arXiv preprint arXiv:2503.14489*, 2025. 6

Figure 4. **Comparison of generated depth with monocular estimated depth.**

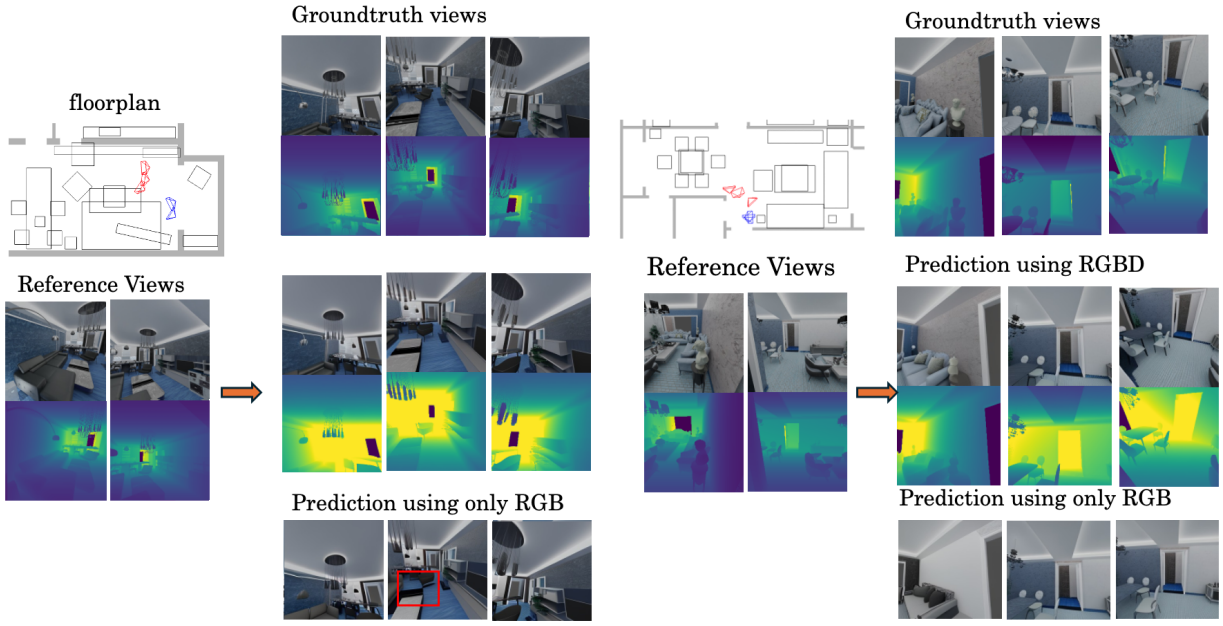

Figure 5. **Additional visual results of novel view RGB-D image generation.**

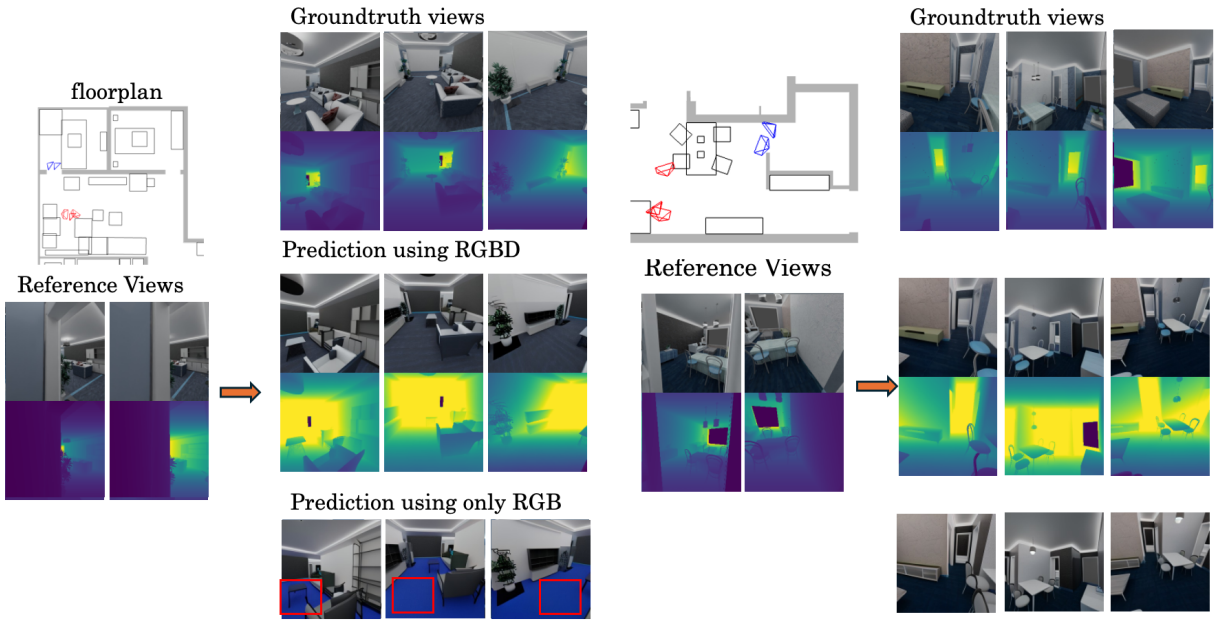

Figure 6. **Additional visual results of novel view RGB-D image generation.**

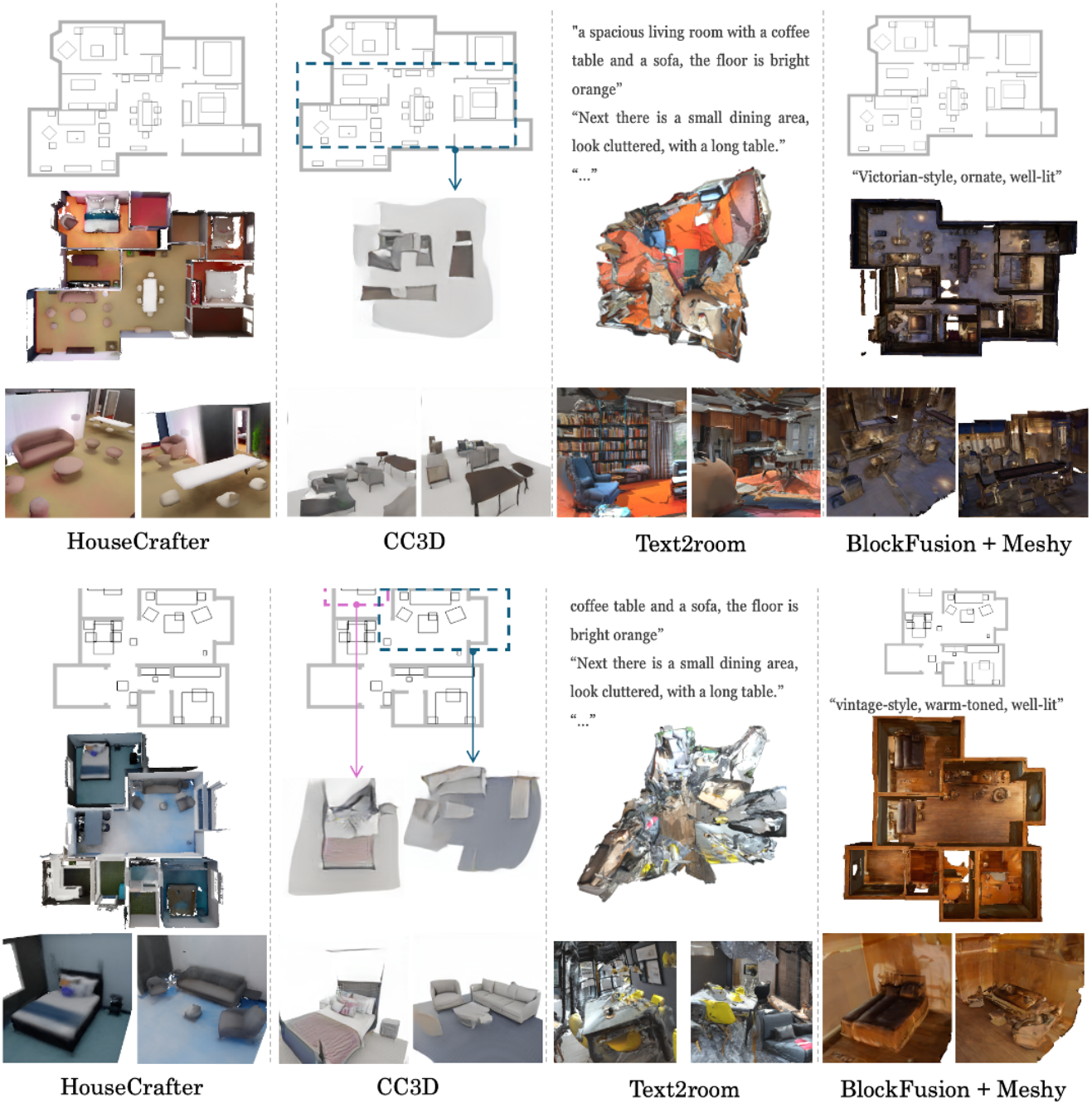

Figure 7. **Additional comparisons with baseline methods.**

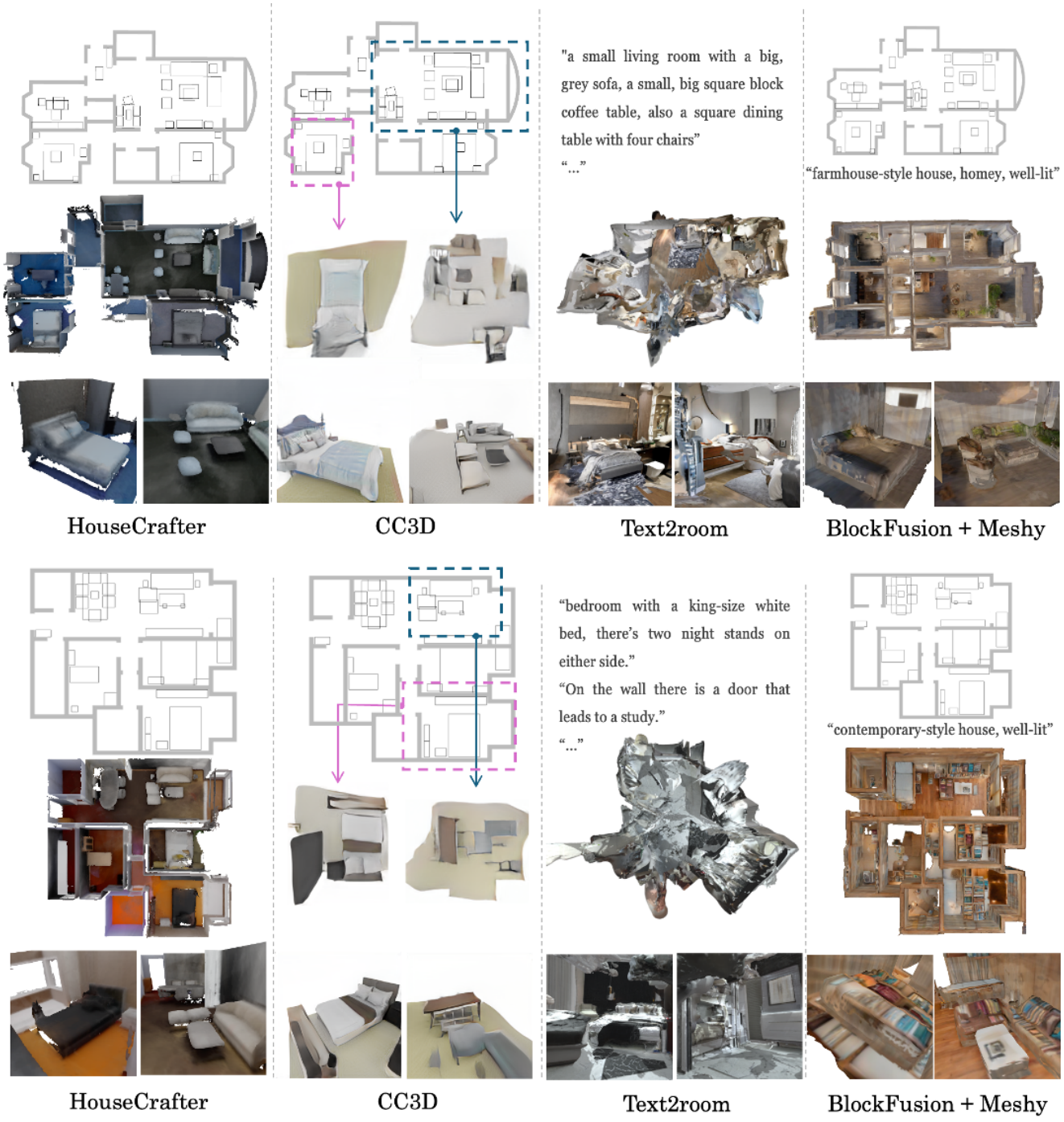

Figure 8. **Additional comparisons with baseline methods.**

Figure 9. **Generation results of our model on ArkitScenes.**

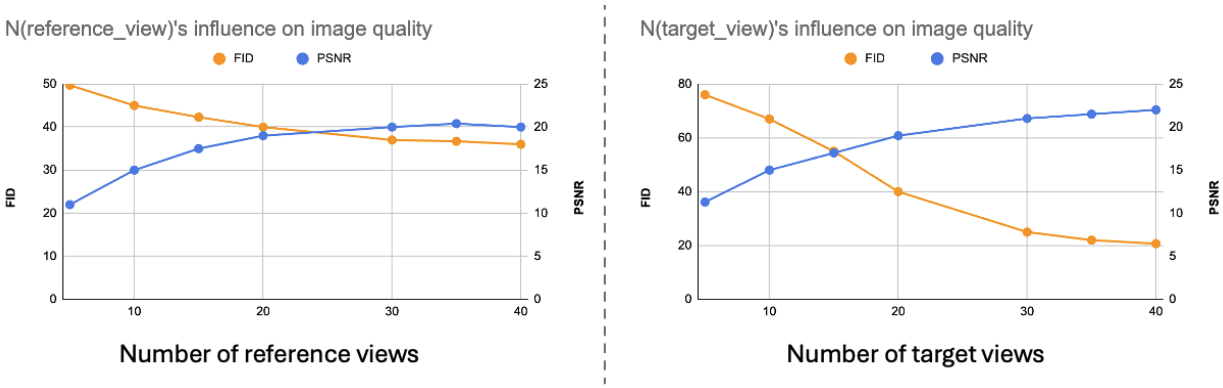

Figure 10. **Influence of selected reference view and target view numbers, respectively.**