Ctrl-Room: Controllable Text-to-3D Room Meshes Generation with Layout Constraints

Supplementary Material

In the supplementary file, we first present more details about our scene code diffusion model in Sec. 1, then we elaborate the layout-guided PeRF module and mask-guided editing method in Sec. 2 and Sec. 3, respectively. Next, we provide our dataset pre-processing, text prompt generation, and implementation details in Sec. 4 and Sec. 5 respectively. Additional experiment results are also illustrated, including panorama generation comparisons in Sec. 6, room layout generations and room mesh comparisons in Sec. 7 and user studies in Sec. 8. Furthermore, we demonstrate that our scene code diffusion model can be trained with free-style text prompts in Sec. 9.

1. Scene Code Denoising Network

In the Layout Generation Stage, we use a holistic scene code to parameterize the indoor scene and design a diffusion model to learn its distribution. Specifically, given a 3D scene $\mathcal{S}$ with N objects, we represent the scene layout as a holistic scene code $\mathbf{x_0} = \{\mathbf{o_i}\}_{i=1}^{N}$. We encode each object o_i as a node with various attributes, i.e., center location $l_i \in \mathbb{R}^3$, size $s_i \in \mathbb{R}^3$, orientation $r_i \in \mathbb{R}$, class label $c_i \in \mathbb{R}^C$. Each node is characterized by the concatenation of these attributes as $\mathbf{o}_i = [c_i, l_i, s_i, r_i]$. As shown in Fig. 1, our scene code denoising network of the layout diffusion model is built upon IDDPM [4]. The whole architecture of the layout diffusion model is similar to IDDPM, while we replace the upsample and downsample blocks with 1D-convolution network in the U-Net, and insert attention blocks after each residual block to capture both the global context among objects and the semantic context from the input text prompt. The input encoding head processes different encoding of the node attributes, e.g., semantic class labels, box centroid, and box orientation. After adding noise, the input encoding is fed into the U-Net to obtain a denoised scene code. The training objectives includes the denoising objective $\mathcal{L}_{\text{denoise}}$ and a regularization term $\mathcal{L}_{\text{physical}}$ to penalize the penetration among objects and walls as follows,

$$\mathcal{L} = \mathcal{L}_{\text{denoise}} + \mathcal{L}_{\text{physical}}, \tag{1}$$

$$\mathcal{L}_{\text{denoise}} = \mathbf{E}_{\mathbf{x_0},t,y,\epsilon}\|\epsilon - \epsilon_\theta(x_t, t, y)\|^2, \tag{2}$$

$$\mathcal{L}_{\text{physical}} = \sum_{t=1}^{T} w_t * (\mathcal{L}_{\text{w-o}} + \mathcal{L}_{\text{o-o}}). \tag{3}$$

where ϵ_θ is the noise estimator which aims to find the noise ϵ added into the input x_0. Here, y is the text embedding of the input text prompts. The hyperparamter w_t is set to

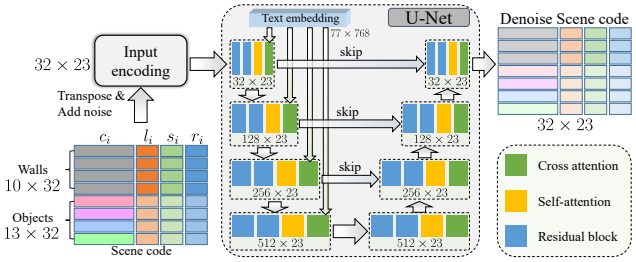

Figure 1. The detailed structure of the scene code denoising network. We here take the bedroom for example to demonstrate the dataflow of the scene code denoiser. The scene code tensor $\mathbf{x_0} \in \mathbb{R}^{N \times D}$, where $N = 23, D = 32$.

$\bar{\alpha}_\mathbf{t} * 0.1$. $\mathcal{L}_{\text{w-o}}$ is the physical violation loss between walls and objects. It is defined as follows,

$$\mathcal{L}_{\mathbf{w-o}} = \sum_{i=1}^{K_{\text{wall}}} \sum_{j=1}^{K_{\text{object}}} \sum_{p=1}^{8} Relu[-(a_i x_{jp} + b_i y_{jp} + c_i z_{jp} + d_i)] * \mathbb{1}(\prod_{\mathbf{w}_i}(x_{jp}, y_{jp}, z_{jp}) \text{ in } \mathbf{w}_i). \tag{4}$$

Here, (a_i, b_i, c_i) is the normal vector of wall $\mathbf{w}_i$ that points towards the room center. $\prod_{\mathbf{w}_i}$ is the operator projecting a point onto the plane defined by $\mathbf{w}_i$. The plane equation of i-th wall is $a_i x + b_i y + c_i z + d = 0$ and $\mathbb{1}(\prod_{\mathbf{w}_i}(x_{jp}, y_{jp}, z_{jp}) \text{ in } \mathbf{w}_i)$ indicates whether the projection of bounding box vertices (x_{jp}, y_{jp}, z_{jp}) of j-th object is inside $\mathbf{w}_i$. We skip some objects such as windows and doors since they can intersect with walls. We adopt the 3D IoU loss $\mathcal{L}_{\text{o-o}}$ in DiffuScene [8] as follows,

$$\mathcal{L}_{\mathbf{o-o}} = \sum_{\mathbf{o}_i, \mathbf{o}_j}^{K_{\text{object}}} \mathbf{IoU}(\mathbf{o_i}, \mathbf{o_j}). \tag{5}$$

During the forward phase, as in IDDPM, we iteratively perform the denoising process and generate a scene code from a partial scene textual description.

We further investigate how the physical regularization term impacts the final 3D scene layout. In Fig. 2, we use two text prompts for layout generation our layout diffusion model trained with and without our physical regularization term, respectively. As can be seen, the diffusion model trained with the physical violation loss can effectively reduce the occurrence of furniture penetrating walls, and also help to

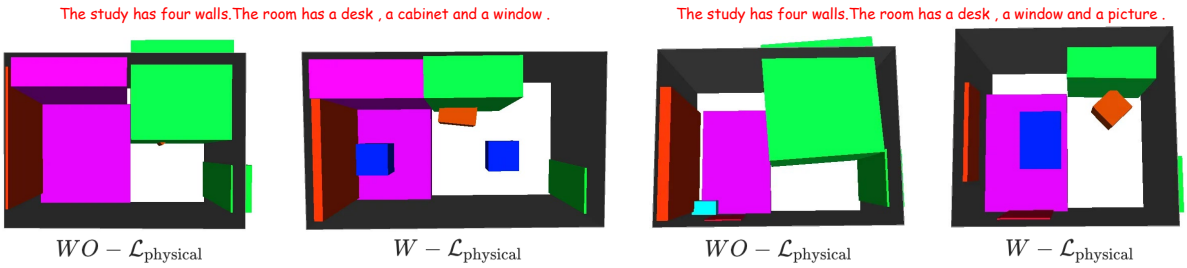

Figure 2. Ablation study of the physical violation loss. Two text prompts of study are used for layout generation using our diffusion model trained without $\mathcal{L}_{\text{physical}}$ (left) and with $\mathcal{L}_{\text{physical}}$ (right), respectively. As a result, in the left sample, diffusion model without $\mathcal{L}_{\text{physical}}$ generates a green desk that penetrates the wall. In the right sample, this phenomenon is alleviated and regulated after using the physical violation loss. Note that the sampling results of these two versions of diffusion models are slightly different since the denoise distribution is different even given the same text prompt.

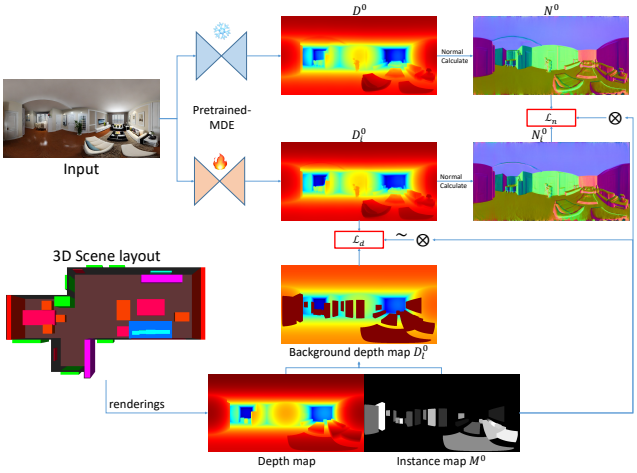

Figure 3. Our Layout-guided depth estimation. To align the estimated depth map with the 3D scene layout, we render depth map and instance map from the 3D scene layout at current view. Then we take the background depth (wall, ceiling, floor) as reference to align the depth prediction by optimizing the pretrained MDE. Avoiding degrade the MDE's performance at object surface, we ensure the normal consistency of each object during the optimization process.

regulate the orientation of the sampled furniture, resulting in more reasonable layouts than the model without the physical regularization term.

2. Layout-Guided Panoramic NeRF

Since the generated panorama is a partial observation of the room subject to occlusions, lifting the single view into a 3D room becomes a complex problem. Here we adopt the Warp and Inpainting scheme to complete the 3D room progressively. After generating the panorama I^0, we recover its depth map D^0 using the state-of-the-art monocular depth estimator (MDE) [14]. However, problems such as scale ambiguity and large occlusions may lead to incomplete 3D room reconstruction. Additionally, ensuring consistency in inpainted panoramas at new viewpoints poses another challenge. Fortunately, the scene layout generated in the first stage offers crucial geometric and semantic guidance, which can help correct biased depth predictions and guide the inpainting process to generate novel view panoramas. In this paper, we employ PeRF [11] as the 3D room model and progressively generate novel viewpoint panoramas with layout guidance to reconstruct the PeRF model.

Layout-Guided Depth Estimation.

To align the estimated panoramic depth map with the scene layout, a naive approach would be to directly compute scale and bias coefficients for the initial depth map D^0. However, as the scene layout consists of object bounding boxes and can not provide pixel-level perfect depth supervision, this method may lead to degraded depth predictions. To address this problem, we propose the panoramic geometry alignment module as depicted in Fig. 3. We use the instance labels of furniture items to exclude the rendered depths within the furniture areas, retaining only the background depth map (e.g., wall, ceiling, floor) denoted as D_l^0. We incorporate a consistency loss $\mathcal{L}_{\text{align}}$ to optimize the pre-trained monocular depth estimator (MDE). This consistency loss is formulated as follows,

$$\mathcal{L}_{\text{align}} = \mathcal{L}_{\text{d}} * w_d + \mathcal{L}_{\text{n}} * w_n, \tag{6}$$

$$\mathcal{L}_{\text{d}} = L_1(D_i^0, D_l^0) * (\sim M^0), \tag{7}$$

$$\mathcal{L}_{\text{n}} = \left|N^0 - N_i^0\right| * M^0. \tag{8}$$

where $\mathcal{L}_{\text{d}}$ represents the smooth L1 loss of depth, $\mathcal{L}_{\text{n}}$ denotes the absolute loss of normal, and w_d, w_n are weighting coefficients. D_i^0 stands for the predicted depth of the MDE at i-th iteration. M^0 is the instance map denoting the foreground, such that we only correct the predicted depth in the background region while preserving normals in the furniture regions. After alignment, we obtain the optimal depth map D^* and normal map N^*.

Layout-Guided Novel View Generation. PeRF [11] trains a panoramic neural radiance field using a single panorama. To render novel view panoramas, it employs the Stable Diffusion model [5] to inpaint 60 perspective images and stitch

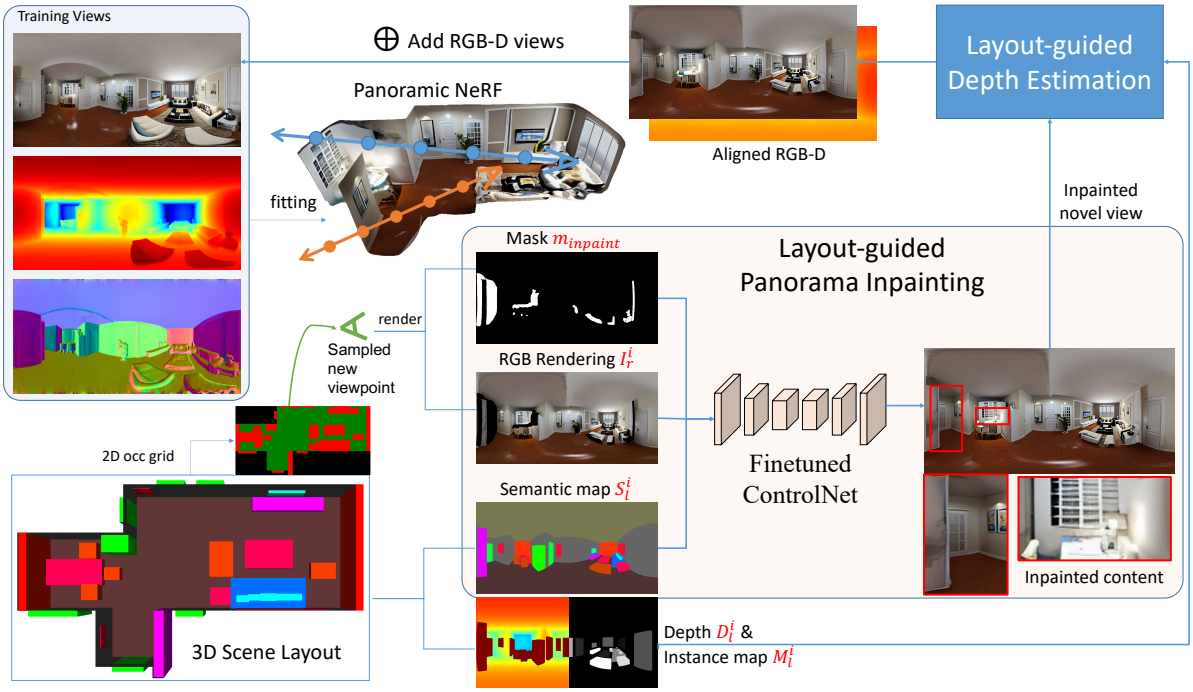

Figure 4. The Layout-guided PeRF takes the input panorama, aligned depth map and normal map as initialization. Then a progressive inpainting module is introduced to generate consistent panoramic images at the sampled novel views. The progressive inpainting module consists of the layout-guided panorama inpainting and the layout-guided depth estimation module. The final RGB-D panoramic pairs are included as training views to finetune PeRF [11].

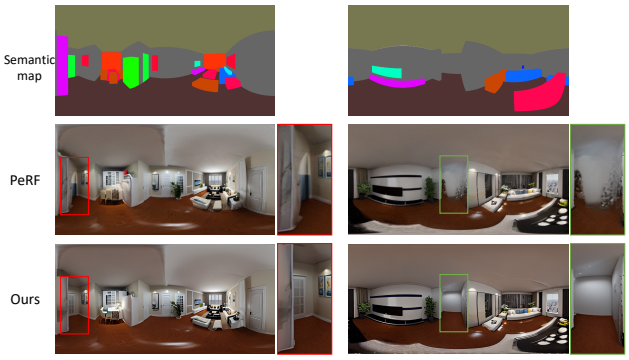

Figure 5. After the layout-guided panorama inpainting, Our generated panoramas at novel viewpoints adhere to the semantic layout and seamlessly integrate with the visible regions, while PeRF [11] fails to synthesize plausible content at those large-size occlusion areas.

them into a panorama. Although it produces consistent renderings at nearby viewpoints, it struggles to synthesize viewpoints that are far apart and involve large unoccluded regions. To address this limitation, we rely on the scene layout to guide the panorama inpainting to maintain cross-view consistency.

As illustrated in Fig. 4, we initialize the scene NeRF with the panorama I^0, the aligned depth map D^* and normal map N^*. We sample new viewpoints in the green area of the occupancy grid that do not conflict with the initial furniture arrangement. At the i-th novel view, we render semantic map S_l^i, depth map D_l^i and instance map M_l^i from the scene layout, these are then combined with the panoramic rendering I_r^i and inpainting mask $\mathbf{m}_{\text{inpaint}}$ obtained from the NeRF and fed to the layout-guided panorama inpainting module to generate the novel view panorama. Using our fine-tuned ControlNet, it achieves training-free panoramic inpainting, which replaces pixels outside the inpainting mask $\mathbf{m}_{\text{inpaint}}$ with I_r^i and fill $\mathbf{m}_{\text{inpaint}}$ based on the semantic map S_l^i. As demonstrated in Fig. 5, our resulting RGB panorama adheres to the semantic layout and seamlessly integrates with the visible regions, while that of PeRF [11] fails to generate reasonable content in the large occlusion areas. Subsequently, after generating the novel view panorama, we apply the layout-guided depth estimation and include it as training views for PeRF following their framework [11].

3. Mask-Guided Editing

To achieve consistent and seamless 3D scene editing, it should achieve two goals, i.e. altering the content according to the user's input, and maintaining appearance consistency for scene objects. We propose a mask-guided image editing as illustrated in Fig. 6, where a chair's position is moved. In the following, we will explain our method with this example. We denote the semantic panorama from

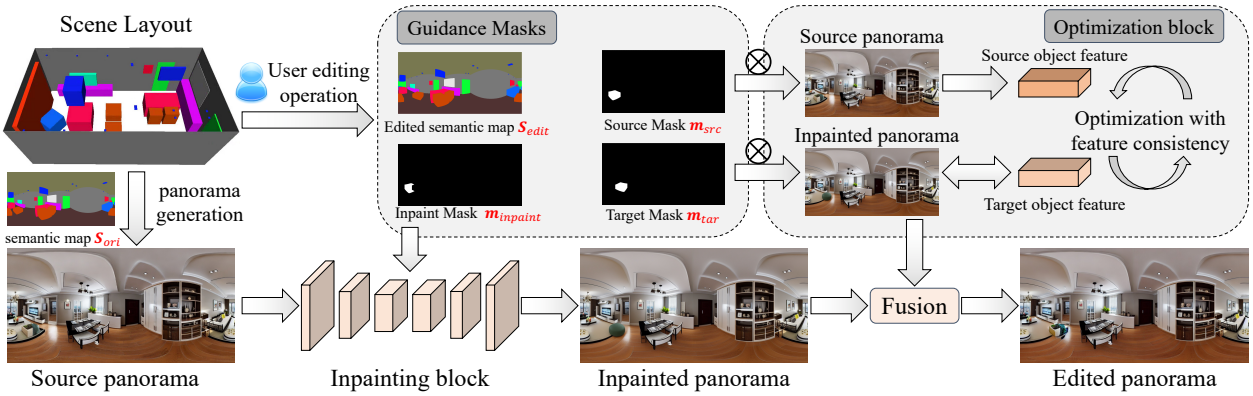

Figure 6. **Mask-guided Editing.** After editing the scene bounding box, we derive guidance masks from the changes in the semantic layout panoramas. We fill in unoccluded regions and optimize the DIFT [9] features to keep the identity of moved objects unchanged.

the edited scene as S_{edited}, then we derive the guidance masks based on its difference from the original one S_{ori}. The source mask $\mathbf{m}_{\text{src}}$ shows the position of the original chair, and the target mask $\mathbf{m}_{\text{tar}}$ indicates the location of the moved chair, and the inpainting mask $\mathbf{m}_{\text{inpaint}} = \{m | m \in \mathbf{m}_{\text{src}} \text{ and } m \notin \mathbf{m}_{\text{tar}}\}$ is the unoccluded region. Given these guidance masks, our method includes two steps: the inpainting step and the optimization step. We first fill in the inpaint area by feeding the inpaint mask $\mathbf{m}_{\text{inpaint}}$ and edited semantic panorama S_{edited} to the inpainting step. Then, in our optimization step, we optimize the DIFT [9] feature to maintain the visual consistency of relocated objects.

Inpainting Step. Denoting the original image as $\mathbf{x}_0^{\text{ori}}$, we replace pixels outside the inpainting mask $\mathbf{m}_{\text{inpaint}}$ with $\mathbf{x}_t^{\text{ori}}$ during the diffusion process. This simple strategy keeps the outside region unchanged. At each reverse diffusion step, we compute:

$$\mathbf{x}_t^{\text{ori}} \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0^{\text{ori}}(1-\bar{\alpha}_t\mathbf{I})), \tag{9}$$

$$\mathbf{x}_t^{\text{new}} \sim \mathcal{N}(\mu_\theta(x_t, t, y, S_{\text{edited}}), \Sigma_\theta(x_t, t, y, S_{\text{edited}})) \tag{10}$$

$$\hat{\mathbf{x}}_{t-1}^{\text{new}} = \mathbf{m}_{\text{inpaint}} \odot \mathbf{x}_t^{\text{new}} + (1 - \mathbf{m}_{\text{inpaint}}) \odot \mathbf{x}_t^{\text{ori}} \tag{11}$$

where $\mathbf{x}_t^{\text{ori}}$ is obtained through propagating $\mathbf{x}_0^{\text{ori}}$ in diffusion process, and $\mathbf{x}_t^{\text{new}}$ is sampled from the fine-tuned ControlNet model, which takes the edited semantic layout panorama S_{edited} and text prompt y as input. As the propagated $\mathbf{x}_t^{\text{ori}}$ is unaware of the new content $\mathbf{x}_t^{\text{new}}$, this may result in distracting boundaries of the inpainted area. To better blend the new content $\mathbf{x}_t^{\text{new}}$ and its surrounding background $\mathbf{x}_t^{\text{ori}}$ in the inpainted area, we update the computation of $\hat{\mathbf{x}}_{t-1}^{\text{new}}$ to,

$$\begin{aligned}\hat{\mathbf{x}}_{t-1}^{\text{new}} = \mathbf{m}_{\text{inpaint}} \odot \mathbf{x}_t^{\text{new}} \\ + (1 - \mathbf{m}_{\text{inpaint}}) \odot (\mathbf{x}_t^{\text{ori}} \cdot \lambda_{\text{ori}} + \mathbf{x}_{t+1}^{\text{new}} \cdot \lambda_{\text{new}})\end{aligned} \tag{12}$$

where λ_{ori} and λ_{new} are hyper-parameters to adjust the weight for fusing the inpainted area and unchanged area. The final result of inpainting is $\hat{\mathbf{x}}_0^{\text{new}}$.

Optimization Step. When the user moves the position of a furniture item, we need to keep its appearance unchanged before and after the movement. The recent work, DIFT [9], finds the learned features from the diffusion network allow for strong semantic correspondence. Thus, we maintain the consistency between the original and moved furniture by requiring their latent features to be consistent. In particular, we extract latent features F_t^l of the layer l in the denoising U-Net network, at timestep t. Then we construct a loss function using the latent features from source area $\mathbf{m}_{\text{src}}$ in source panorama $\mathbf{x}_0^{\text{ori}}$ and target area $\mathbf{m}_{\text{tar}}$ in inpainted panorama $\hat{\mathbf{x}}_0^{\text{new}}$.

For conciseness, we denote the target image $\hat{\mathbf{x}}_0^{\text{edit}}$ initialized by $\hat{\mathbf{x}}_0^{\text{new}}$. We first propagate the original image $\mathbf{x}_0^{\text{ori}}$ and $\hat{\mathbf{x}}_0^{\text{edit}}$ to get $\mathbf{x}_t^{\text{ori}}$ and $\hat{\mathbf{x}}_t^{\text{edit}}$ at timestep t by diffusion process, respectively. At each iteration, we use the same ControlNet model to denoise both $\mathbf{x}_t^{\text{ori}}$ and $\hat{\mathbf{x}}_t^{\text{edit}}$ and extract the latent features of them, denoted as F_t^{ori} and F_t^{edit}, respectively. Based on the strong correspondence between the features, the source mask area $\mathbf{m}_{\text{src}}$ and the target area $\mathbf{m}_{\text{tar}}$ in F_t^{ori} and F_t^{edit} need to have high similarity. Here, we utilize the cosine embedding loss to measure the similarity, and define the optimization loss function as follows:

$$\mathcal{L}_{\text{obj}} = -cos(\text{sg}(F_t^{ori} \odot \mathbf{m}_{\text{src}}), F_t^{edit} \odot \mathbf{m}_{\text{tar}}) \tag{13}$$

Here, sg is the stop gradient operator, the gradient will not be back-propagated for the term $\text{sg}(F_t^{ori} \odot \mathbf{m}_{\text{src}})$. Then we minimize the loss iteratively. At each iteration, $\hat{\mathbf{x}}_t^{\text{edit}}$ is updated by taking one gradient descent step with a learning rate η to minimize the loss $\mathcal{L}_{\text{obj}}$ as,

$$\hat{\mathbf{x}}_t^{k+1} = \hat{\mathbf{x}}_t^k - \eta \cdot \frac{\partial \mathcal{L}_{\text{obj}}}{\partial \hat{\mathbf{x}}_t^k} \tag{14}$$

After M steps optimization, we apply the standard denoising process to get the final result $\hat{\mathbf{x}}_0^{\text{edit}}$.

Figure 7. Example of object bounding box annotation.

4. Dataset

Structured3D dataset preprocessing Structured3D consists of $3,500$ houses with $21,773$ rooms, where each room is designed by professional designers with rich 3D structure annotations, including the room planes, lines, junctions, and orientated bounding box of most furniture, and photo-realistic 2D renderings of the room. In our work, we use the 3D orientated bounding boxes of furniture, 2D RGB panorama, and 3D lines and planes of each room. While the original dataset lacks semantic class labels for each furniture bounding box. The dataset preprocessing aims to produce clean ground truth data for our layout generation module and appearance generation module.

- **Orientated Object Bounding Box Annotation.** As the original dataset lacks semantic label for each orientated object bounding box, we first unproject the RGB panorama and depth map into a point cloud of the room, then manually annotate the object semantic class and add more accurate object bounding boxes based on the noisy annotation of the original version. As shown in Fig. 7, by using labelCloud [6], three data annotators worked for 1200 hours to annotate 5,064 bedrooms, 3,064 livingrooms, 2,289 kitchens, 698 studies, and 1,500 bathrooms, getting nearly 150K accurate orientated 3D bounding boxes across 25 object categories.
- **Scene Node Encoding.** We define our holistic scene code based on a unified encoding of walls and object bounding box. Each object o_j is treated as a node with various attributes, i.e., center location $l_i \in \mathbb{R}^3$, size $s_i \in \mathbb{R}^3$, orientation $r_i \in \mathbb{R}$, class label $c_i \in \mathbb{R}^C$. The orientated bounding box is off-the-shelf, we extract the inner walls based on the line junctions and corners of the 3D room. Then we put the orientated object bounding boxes and walls into a compact scene code. Concretely, we define an additional 'empty' object and pad it into scenes to have a fixed number of object across scenes. Each object rotation angle is parametrized by a 2-d vector of cosine and sine values. Finally, each node is characterized by the concatenation of these attributes as $\mathbf{o}_i = [c_i, l_i, s_i, \cos r_i, \sin r_i]$.
- **data filtering.** We start by filtering out those problematic scenes such as rooms with wall number less than 4 or larger than 20. We also remove those scenes with too few or too many objects. The number of walls of valid bedrooms is between 4 and 10, and that of objects is between 3 and 13. As for living rooms, the minimum and maximum numbers of walls are set to 4 and 20, and that of objects are set to 3 and 24 respectively. The number of walls for valid kitchens, studies, and bathrooms is the same as for bedrooms, while the objects number is between 3 and 24. Thus, the number of scene nodes is $N = 23$ in bedrooms, $N = 44$ in living rooms, and $N = 34$ in kitchens, studies, and bathrooms. After filtering, we get 4,961 bedrooms, 3,039 living rooms, 1,848 kitchens, 638 studies, and 1,500 bathrooms.

Text Prompt Generation We follow the SceneFormer [12] to generate text prompts describing partial scene configurations. Each text prompt contains two to four sentences. The first sentence describes how many walls are in the room, then the second sentence describes two or three existing furniture in the room. The following sentences mainly describe the spatial relations among the furniture, please refer to SceneFormer [12] and DiffuScene [8] for more detailed explanation of relation-describing sentences. In this way,

we can get some relation-describing sentences to depict the partial scene. Finally, we randomly sampled zero to two relation-describing sentences to form the text prompt for 3D room generation.

5. Implementation details

Training and inference details.

- In the layout generation stage, We train the scene code diffusion model on our processed typical indoor rooms data of Structured3D [16] for $200,000$ steps. The frozen text encoder we adopted is the same as Stable Diffusion [5]. The training is performed using the AdamW optimizer with a batch size of 128 and a learning rate of $1e-4$, utilizing 2 A6000 GPUs. During the inference process, we utilize the DDIM [7] sampler with a step size of 200 to perform scene code denoising.
- In the appearance generation stage, we fine-tune the segmentation-conditional ControlNet model based on the pairwise semantic and RGB panorama of Structured3D. The fine-tuning process is implemented on two A6000 GPUs for 150 epochs(about 3 days). In the inference phase, we generate high-fidelity and loop-consistent RGB panorama through DDIM sampler with 100 steps, rotating both semantic layout panorama and the denoised image for $\gamma = 90°$ at each step.
- As for the layout-guided panoramic NeRF module, we set $w_d = 0.6$ and $w_n = 0.4$ for depth alignment loss. During the NeRF fitting process, we randomly select 8 viewpoints for living room scenarios and 4 viewpoints for other room types. The NeRF training settings are the same as PeRF [11].
- As for the mask-guided editing module, we utilize the fine-tuned Control-Seg model to inpaint the background content and optimize the latents of the edited panorama. In inpainting step, the weights used too fuse the unpainted area and unchanged area are set $\lambda_{\text{ori}} = 0.8, \lambda_{\text{new}} = 0.2$. In the optimization step, the maximum iteration is $M = 50$, the learning rate η for optimization is initialized to 0.1 and then gradually decreases to 0.01.

5.1. Baseline Implementations

We provide implementation details for baseline methods in the following:

- MVDiffusion [10]: To get a high-resolution photo realistic panorama, MVDiffusion employs 8 branches of SD [5] model and correspondence-aware attention mechanism to generate multi-view images simultaneously. We first fine-tune the pre-trained model of MVDiffusion on Structured3D for 10 epochs(about 3 days). Since each generated subview image of MVDiffusion is at 512×512 resolution, the final panorama is pretty large. We resize the generated panorama of MVDiffusion from 4096×2048 to 1024×512. Then the 8 subview perspective images are extracted from the post-processed panorama using the same camera settings (FOV=90°,rotation=45°). The same operation is adopted on our generated panoramic images. Finally, we combine the panorama from MVDiffusion with the depth estimation [14] and Poisson reconstruction [3] module to create a 3D mesh.
- Text2Light [1]: Text2Light creates HDR panoramic images from text using a multi-stage auto-regressive generative model. We choose Text2Light as one of the baseline for our panorama generation and 3D room mesh generation. We first generate RGB panoramas from the input text using Text2Light, then lift it into 3D mesh using the same panoramic reconstruction module as MVDiffusion. When evaluating the panoramic image quality, we adopt the same processing as MVDiffusion to get multi-view perspective images of Text2Light.
- Text2Room [2]: Text2Room is the current state-of-the-art and off-the-shelf method for 3D room mesh generation. It utilizes 20 camera spots of a pre-defined trajectory to expand new areas as much as possible by generating 10 images at each spot. Here We use its final fused poison mesh for 3D mesh comparison. For a fair comparison of 2D renderings evaluation, we only use the renderings at the origin of the final mesh.
- Text2NeRF [15] generates 3D scenes from a text prompt using NeRF as the 3D representation and leverages a pre-trained text-to-image diffusion model and monocular depth estimation to constrain the 3D reconstruction. However, we found it fails to reconstruct 360° scenes. We present some NeRF reconstructions from Text2NeRF stitched into panorama images in Fig.15. Note that only ∼154° horizontal field of view (FOV) and ∼113° vertical FOV is shown since the rest of the scene is not reconstructed by the method. Thus we skip the comparison with this method.

To ensure a fair comparison, we render 60 perspective images at the origin using the final textured meshes of all methods. The camera field of view is set to 140° to capture scene layout information for evaluating the CS and IS scores. Additionally, we render corresponding geometric images in Fig. 12 to showcase the geometry quality.

6. Panorama Generation Comparison

In Fig. 8, we study the performance of our panorama generation module with and without loop-consistent sampling mechanism, the ablation indicates the loop-consistent sampling helps the generated panorama obtain better texture consistency. Fig. 9 presents additional results for panorama generation. Given a simple partial-scene text prompt, our approach obtains better RGB panorama than that of Text2Light [1] and MVDiffusion [10], which demonstrates the effectiveness of our well-designed framework. While Text2Light suffers from the inconsistent loop and unex-

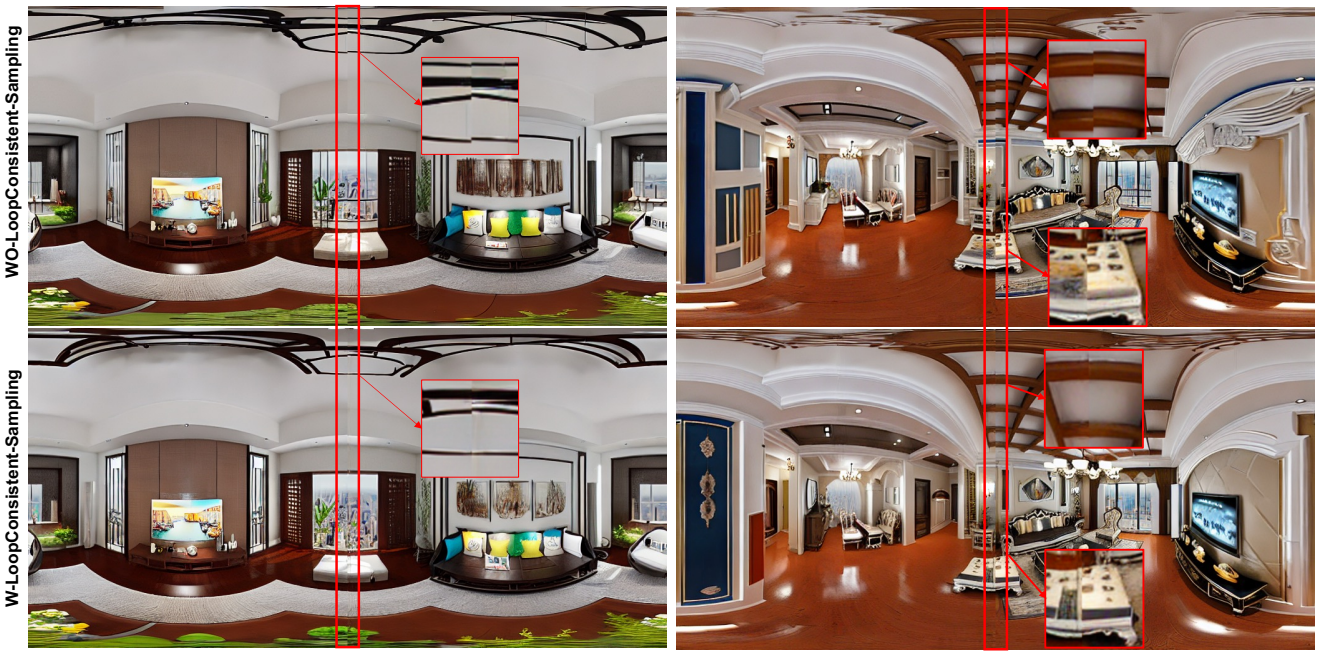

Figure 8. Ablation of loop-consistent sampling examples. We rotate the generated panorama by 180° to better visualize the leftmost and rightmost content consistency.

pected content of the generated panorama, MVDiffusion fails to recover a reasonable room layout from the text prompt.

7. Additional Qualitative Results

In Fig. 10, we first visualize more generated room layouts generation of typical rooms in the format of semantic 3D bounding boxes. Then, we show additional qualitative comparison results between our method and baselines in Fig. 12. We demonstrate more scene editing results of our method in Fig. 11.

8. User Study

Follow Text2Room [2], we conduct a user study and ask $n = 61$ ordinary users to score the Perceptual Quality(PQ) and 3D Structure Completeness(3DS) of the generated room on a scale of $1-5$. Different from Text2room which only demonstrates the perspective renderings of the 3D room, we directly show users the generated mesh to get a global evaluation of the whole generated 3D room. We show an example of the presented interface of the user study in Fig. 13. In total, we presented 40 top-down views from 10 scenes and report averaged results for each method. Users favor our approach, which emphasizes the superiority of our more plausible geometry, along with the vivid texture.

9. Free style prompts

We show the adaptability of our method by utilizing Large Language Model (LLM) GPT-4 Vision (GPT-4V) [13] to generate text captions from panorama images of Structured3D [16] bedroom scenes. The prompt used for the LLM is as shown in Table 1.

We train and test with the LLM generated captions as conditioning for layout generation. Fig. 14 shows some results from the test set and corroborates our ability to produce plausible 3D room layout following free-style test prompts.

References

[1] Zhaoxi Chen, Guangcong Wang, and Ziwei Liu. Text2light: Zero-shot text-driven hdr panorama generation. *ACM TOG*, 41(6):1–16, 2022. 6, 8

[2] 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. 6, 7

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

[4] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 1

[5] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 2, 6

[6] Christoph Sager, Patrick Zschech, and Niklas Kuhl. labelCloud: A lightweight labeling tool for domain-agnostic 3d

Figure 9. Qualitative comparison for panorama generation. Generated panorama is visualized in a panoramic image viewer to facilitate the user to check the global content of panorama. The left side of each column is two zoom-in views, and the right side is the fisheye view. Text2Light [1] exists serious inconsistent problem on the border of the generated panorama, it also shows a lot of unexpected stuff in the image. MVDiffusion [10] fails to synthesize reasonable content for the target room type. In contrast, our method obtains layout plausible and vivid panorama from the given text prompt of partial scene.

object detection in point clouds. *Computer-Aided Design and Applications*, 19(6):1191–1206, 2022. 5

[7] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 6

[8] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. Diffuscene: Scene graph denoising diffusion probabilistic model for generative indoor scene synthesis. *arXiv preprint arXiv:2303.14207*, 2023. 1, 5

[9] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *arXiv preprint arXiv:2306.03881*, 2023. 4

[10] Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. Mvdiffusion: Enabling holistic multi-

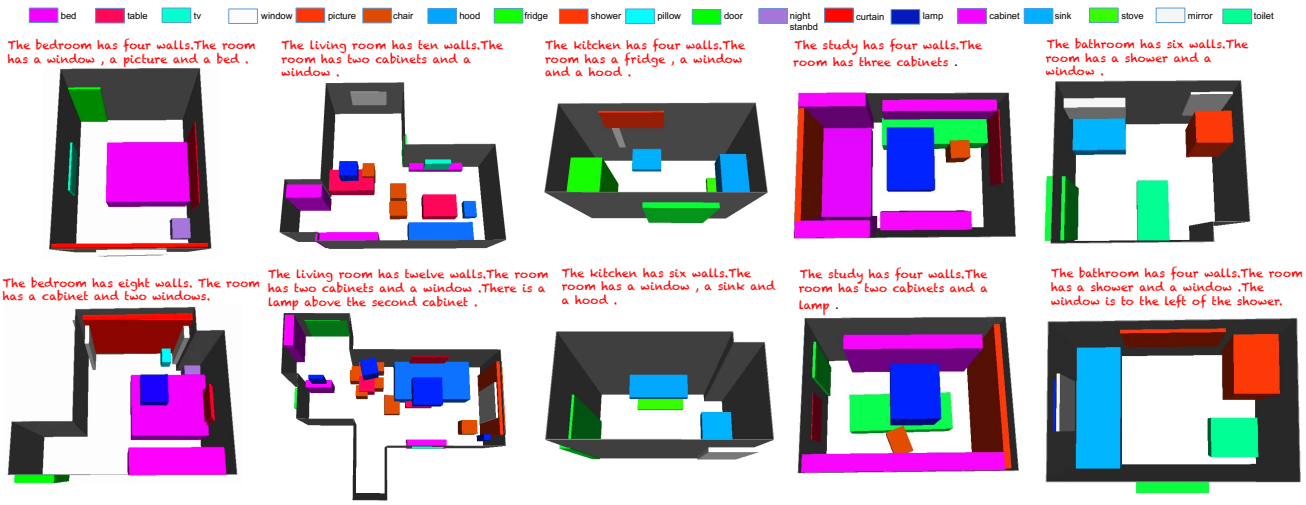

Figure 10. Additional room layout generations. In the bedroom, the bed is often attached to the wall, with a picture above it and a television in front of it. In the living room, there is often a double-seat sofa accompanied by a table and a single-seat sofa. The dining table is usually placed in a separate area of the living room, along with cabinets and chairs. In the kitchen, common furniture includes a stove, sink, fridge, and hood, which are all well-placed in the room. In the study, there is typically a desk accompanied by a chair and one or more bookshelves, and sometimes there is also a bed in the room. In the bathroom, there is usually a sink with a mirror, a toilet, and a shower.

(a) Resize the sofa (b) Remove the chair

Figure 11. Additional scene editing results. In each sub-figure, the left part is the original 3D room, the right part shows the final mesh after users' interactive editing.

view image generation with correspondence-aware diffusion. *arXiv preprint arXiv:2307.01097*, 2023. 6, 8

[11] Guangcong Wang, Peng Wang, Zhaoxi Chen, Wenping Wang, Chen Change Loy, and Ziwei Liu. Perf: Panoramic neural radiance field from a single panorama. *arXiv preprint arXiv:2310.16831*, 2023. 2, 3, 6

[12] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *2021 International Conference on 3D Vision (3DV)*, pages 106–115. IEEE, 2021. 5

[13] Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of lmms: Preliminary explorations with gpt-4v(ision), 2023. 7

[14] Ilwi Yun, Chanyong Shin, Hyunku Lee, Hyuk-Jae Lee, and Chae Eun Rhee. Egformer: Equirectangular geometry-biased transformer for 360 depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6101–6112, 2023. 2, 6

[15] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Text2nerf: Text-driven 3d scene generation with neural radiance fields. *arXiv preprint arXiv:2305.11588*, 2023. 6

[16] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *ECCV*, pages 519–535. Springer, 2020. 6, 7

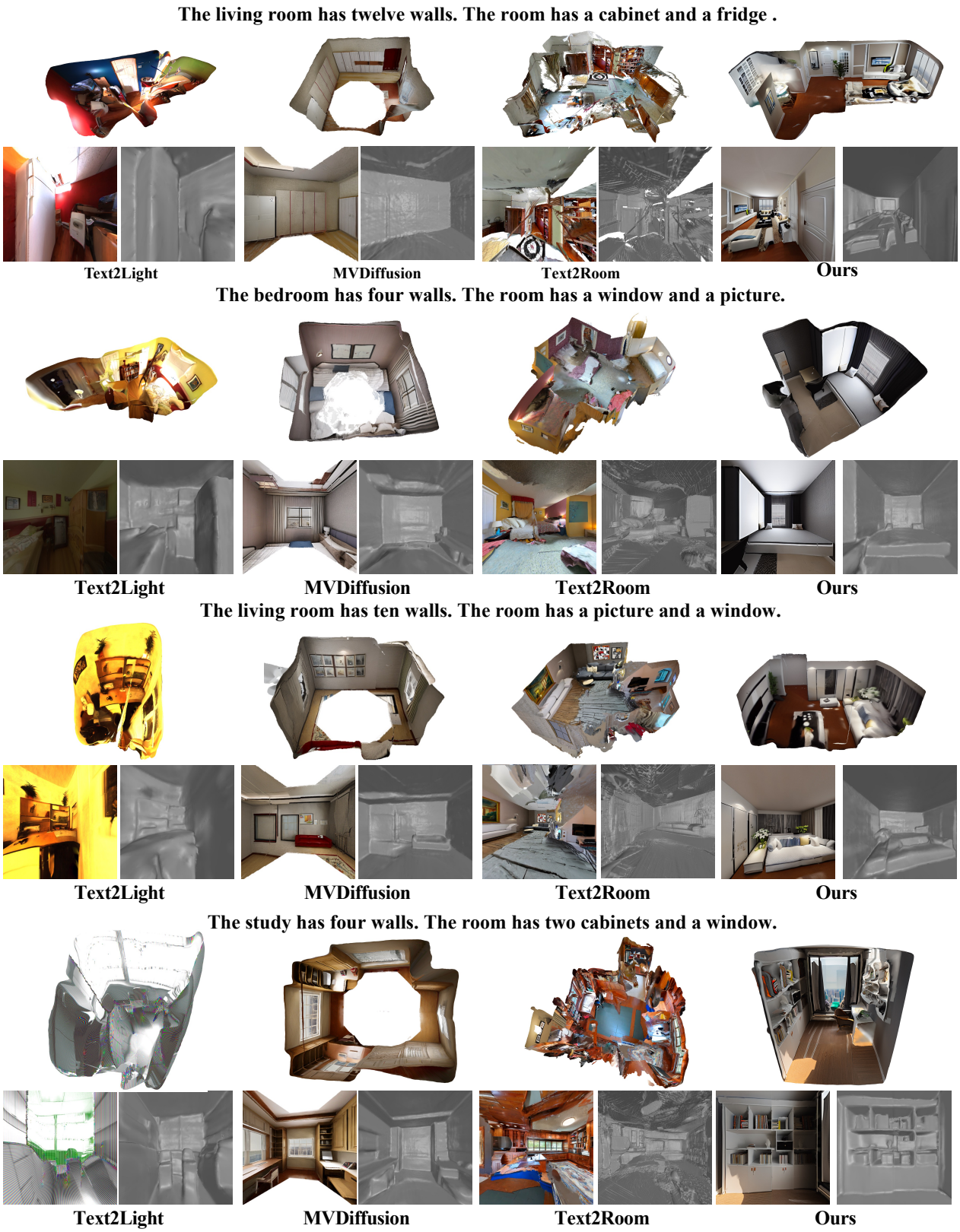

Figure 12. Additional qualitative comparison with previous works.The first row shows a textured 3D room model, and the second row shows perspective colored renderings and geometric renderings from the room model.

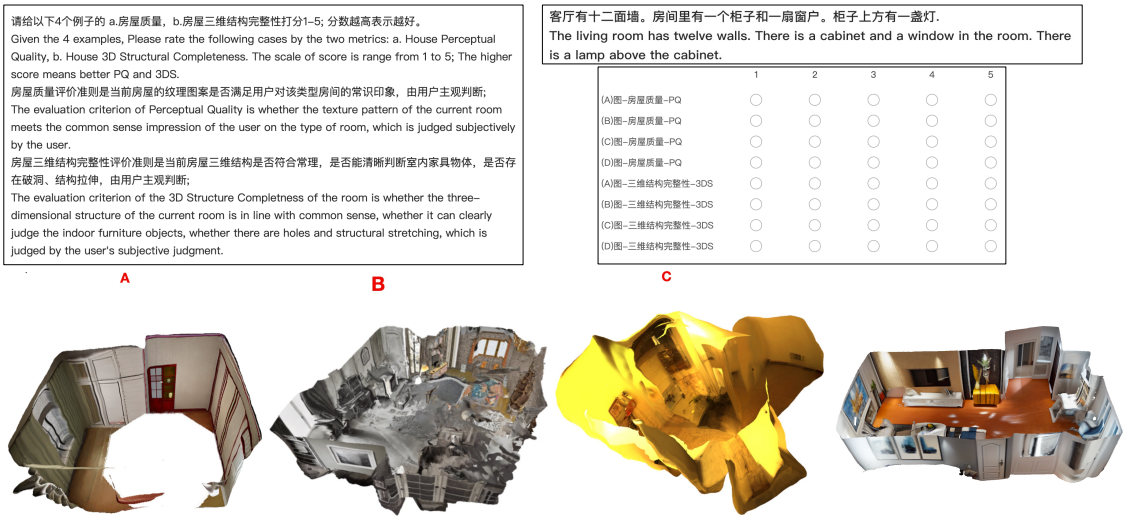

Figure 13. User study interface. We provide users with multiple top-down images from different methods and ask users to rate the given 3D meshes on a scale from 1 to 5, according to the criteria of Perceptual Quality and 3D Structure Completeness.

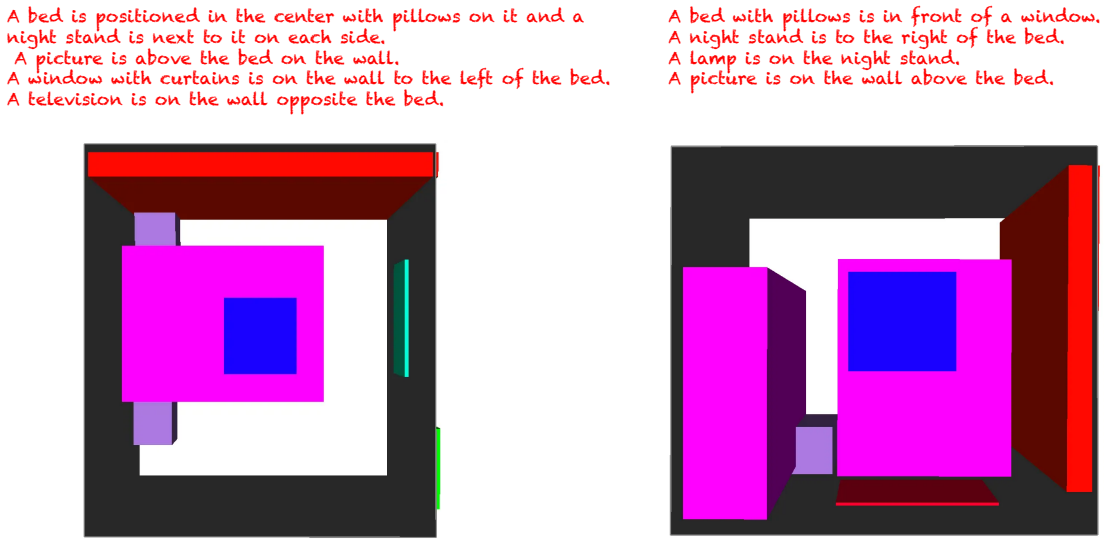

Figure 14. Text-conditioned layout generation on Structured3D using GPT-4V text prompts. Our method synthesizes a plausible scene layout that matches the description.

Table 1. Prompt for GPT-4V to generate captions from panorama images

Describe what is displayed in the panoramic image succinctly in 3 or 4 sentences encoded in ASCII. Do not use lengthy or compound sentences. Do not mention that it is an image or a panoramic image. Do not describe the background, lighting, color palette or count the number of objects. Do not describe size like “small”, “large”, etc. Describe the relative positions of each objects in the scene using only these relationships: “on”, “above”, “surrounding”, “inside”, “left touching”, “right of”, “front touching”, ‘in front of”, “right touching”, “left of”, “behind touching”, “behind”, “next to”, “left of”, “right of”. Optionally, describe the object attributes (color, texture etc). In the description only use these objects: table, night stand, picture, door, cabinet, curtain, bathtub, bed, sink, fridge, shelves, window, lamp, chair, pillow, dresser, bookshelf, sofa, counter, desk, mirror, television, wall

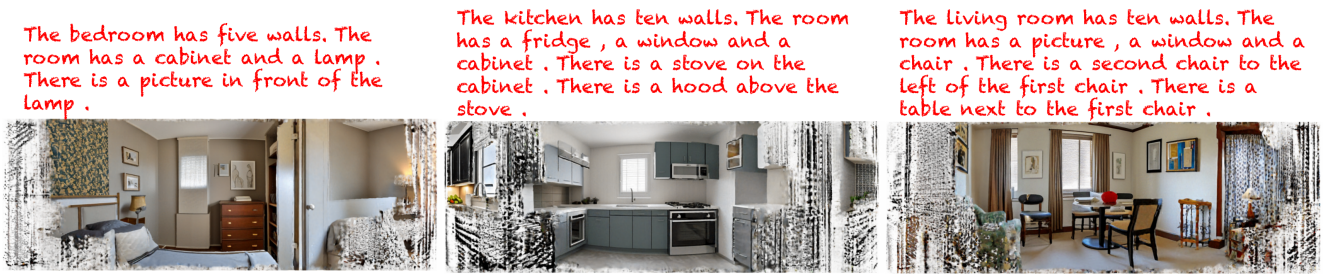

Figure 15. Text2NeRF results. The NeRF reconstructions are stitched into panorama images. Only 154° horizontal FOV and 113° vertical FOV is shown since the method was not able to reconstruct the rest of the scene.