RealmDreamer: Text-Driven 3D Scene Generation with Inpainting and Depth Diffusion

Jaidev Shriram[1*] Alex Trevithick[1*] Lingjie Liu[2] Ravi Ramamoorthi[1]

[1]University of California, San Diego
{jkariyatt, atrevithick, ravir}@ucsd.edu

[2]University of Pennsylvania
lingjie.liu@seas.upenn.edu

https://realmdreamer.github.io/

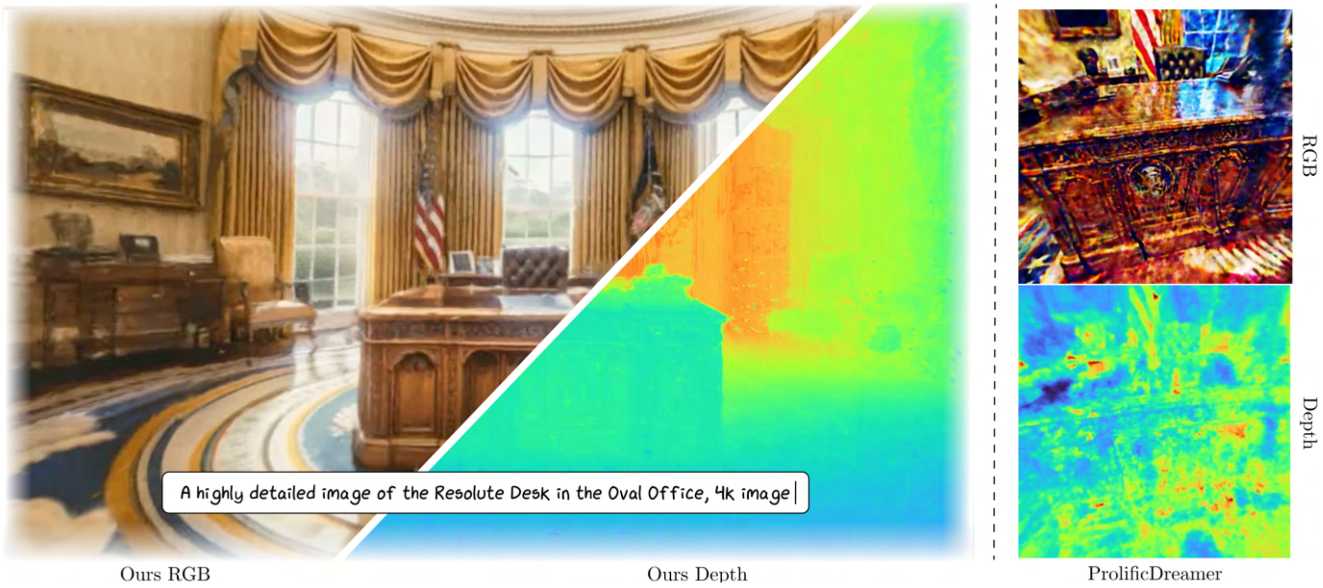

A highly detailed image of the Resolute Desk in the Oval Office, 4k image
Ours RGB
Ours Depth
ProlificDreamer
RGB
Depth

Figure 1. **A scene created by our technique on the left with prompt above compared to baseline ProlificDreamer [66] on the right.** RealmDreamer generates 3D scenes from text prompts such as the one above. Using pretrained 2D inpainting and depth diffusion models, our method achieves state-of-the-art results with parallax, detailed appearance, and high-fidelity geometry.

Abstract

*We introduce **RealmDreamer**, a technique for generation of general forward-facing 3D scenes from text descriptions. Our technique optimizes a 3D Gaussian Splatting representation to match complex text prompts. We initialize these splats by utilizing the state-of-the-art text-to-image generators, lifting their samples into 3D, and computing the occlusion volume. We then optimize this representation across multiple views as a 3D inpainting task with image-conditional diffusion models. To learn correct geometric structure, we incorporate a depth diffusion model by conditioning on the samples from the inpainting model, giving rich geometric structure. Finally, we finetune the model using sharpened samples from image generators. Notably, our technique does not require video or multi-view data and can synthesize a variety of high-quality 3D scenes in different styles, consisting of multiple objects. Its generality additionally allows 3D synthesis from a single image.*

1. Introduction

Text-based 3D scene synthesis has the potential to revolutionize 3D content creation by democratizing synthesis of high-fidelity 3D environments. With the increasing popularity of virtual reality and other spatial computing interfaces, this task has recently become even more important. Creating novice-friendly systems is made difficult by the long iteration times required for existing text-to-3D methods and restriction to simple object-level data or panoramas. An ideal 3D generation tool should offer users ownership over more aspects of the generation process, and fundamen-

[*] equal contribution

tally, the ability to create 3D *scenes* for immersive content creation and experiences (see Fig. 1).

The scarcity of scene-level 3D data makes generating 3D representations highly challenging. With virtually no scene-level 3D priors available, inconsistent 2D priors, such as CLIP [47] and Stable Diffusion [51], become crucial. Consequently, creating consistent gradients across multiple views for cohesive scene generation remains significantly difficult.

Trained on a large corpus of data, text-conditioned 2D diffusion models have shown excellent use as general purpose priors for a variety of tasks in computer vision, such as generation [72], editing [38, 43], classification [13], and segmentation [68]. Despite being a 2D prior over images, these models have also found application in 3D settings, such as enabling view synthesis [15, 15, 75] or editing [19] in the absence of any 3D training data. Most relevant to our task, some methods create 3D representations from text [46, 66] by optimizing the diffusion prior with respect to a 3D representation. However, due to the high complexity of 3D scenes, these methods have thus far been restricted to object-level data or panoramas with incorrect learned geometry (see Fig. 2).

To address this problem, we introduce **RealmDreamer**, a method for high-fidelity generation of 3D scenes from text prompts. Our key insight is to use pretrained inpainting and depth priors with a robust initialization of a 3D Gaussian Splatting model [30], to produce scenes that can be rendered from a wide baseline. Concretely, our contributions are the following:

1. A robust scene initialization for scene-level 3DGS with 2D priors (Sec. 4.1)
2. A framework for learning consistent 3D with 2D inpainting priors, within an *occluded volume* (Sec. 4.2)
3. A method for distillation from diffusion-based depth estimators (Sec. 4.3)
4. State-of-the-art results in text-based generation of 3D scenes (see Fig. 7)

2. Related Work

Text-to-3D. The first methods for text to 3D generation were based on retrieval from large databases of 3D assets [7, 8, 14]. Subsequently, learning-based methods have dominated. Using GANs, Text2Shape [9] learned to create textured 3D representations from text descriptions with GANs [1]. However, due to the dearth of diverse paired text and 3D data, more recent methods leverage 2D priors. Methods like DreamFields [25] and CLIP-Forge [54] leveraged CLIP [47] priors for text-to-3D. The most successful methods, however, have recently relied on the large-scale priors learned from text-to-image diffusion models [51, 53]. The seminal DreamFusion [46] and SJC [65] proposed to distill knowledge from such models into a 3D representation through score distillation sampling (SDS). There have been many follow-up works on this [10, 32, 36, 39], such as ProlificDreamer [66] which improves SDS by training an additional view-dependent Low-Rank Adaptation [24] for better 3D consistency.

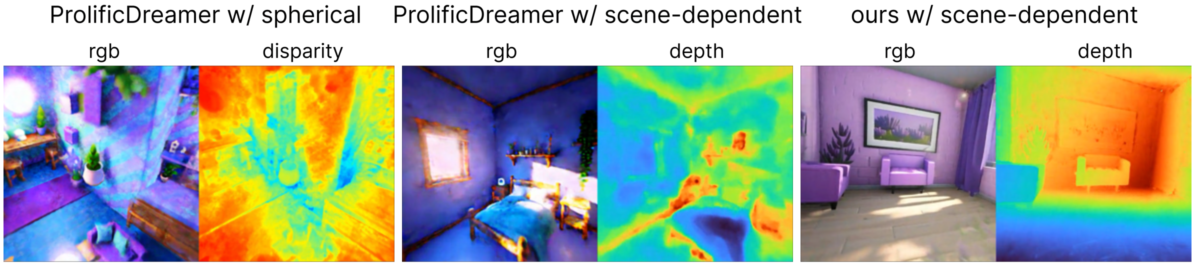
View synthesis with diffusion and 3D inpainting. Since the introduction of SDS, many papers have leveraged the capabilities of 2D diffusion models for 3D tasks. View synthesis from a single image may be accomplished with similar losses to SDS [15, 75]. 3DiM [67] learns to directly sample views conditioned on relative camera poses and an input image. Recent methods have pushed this direction further by increasing dataset size and exploring better conditioning architectures [34, 35, 55, 56]. Other methods perform diffusion in 3D space [44] or in a 3D latent space [26]. Finally, some methods condition denoising directly on renderings from 3D consistent models [6, 17] for inpainting in a multi-view consistent manner. However, all of the methods mentioned rely on object-level data, making scene-level synthesis infeasible. Separately, some methods focus on inpainting in 3D [41, 42], yet their results are limited to small baselines and inpainting masks. iNVS [27] also leverages inpainting diffusion models for view synthesis of objects from a single image.

Concurrent work. There are numerous concurrent works in the fast-moving text-to-3D space. Due to their number, we note only the most relevant concurrent works and clarify our differences from them. Some recent methods [11, 31, 63, 70] propose to utilize 3DGS representations as a backbone for text-to-3D, focusing on object-centric data. Text2NeRF [71] optimizes a NeRF for scene-level view synthesis, but its results are limited to narrow baselines with little parallax. Scenewiz3D [73] improves ProlificDreamer with the incorporation of RGBD priors, but the results lack realism. LucidDreamer [12] additionally attempts to tackle the task of 3D scene generation, but its results are mostly limited to simple camera trajectories such as those exhibiting strictly rotation or translation. In contrast, our method leverages inpainting and depth priors for scene synthesis with non-trivial camera trajectories.

3. Preliminaries

3.1. 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [30] has recently emerged as an explicit alternative to NeRF [40], offering extremely fast rendering speeds and a memory-efficient backwards pass. In 3DGS, a set of splats are optimized from a set of posed images. The soft geometry of each splat is represented by a mean $\mu \in \mathbb{R}^3$, scale vector $s \in \mathbb{R}^3$, and rotation R parameterized by quaternion $q \in \mathbb{R}^4$, so that the covariance of the Gaussian is given by $\Sigma = RSS^TR^T$ where $S = \text{Diag}(s)$.


ProlificDreamer w/ spherical
rgb
disparity
ProlificDreamer w/ scene-dependent
rgb
depth
ours w/ scene-dependent
rgb
depth

"Small lavender room, soft lighting, unreal engine render, voxels."

Figure 2. Our method, compared to the state-of-the-art ProlificDreamer [66], shows significant improvements. ProlificDreamer's public results, with outward-looking cameras from a scene-independent sphere, result in poor depth quality. Even with a complex training trajectory, ProlificDreamer yields unsatisfactory geometry and oversaturated renders. In contrast, our approach produces visually appealing images with realistic geometry.

Additionally, each splat has a corresponding opacity $\sigma \in \mathbb{R}$ and color $c \in \mathbb{R}^3$.

The splats $\{\Theta_i\}_{i=1}^N = \{\mu_i, s_i, q_i, \sigma_i, c_i\}_{i=1}^N$ are projected to the image plane where their contribution α_i is computed from the projected Gaussian (see [77]) and its opacity. For a given pixel, the Gaussians are sorted by depth to compute the pixel color by α-blending:

$$C = \sum_{i=1}^{N} \alpha_i c_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{1}$$

A significant drawback of 3DGS-based approaches is the necessity of a good initialization. State-of-the-art results are only achieved with means μ_i initialized by the sparse depth of Structure-from-Motion [57]. Additionally, heuristics to grow the number of Gaussians to recover under-reconstructed regions are essential. 3DGS initializes new splats based on the gradients of the loss with respect to the camera space means. To address the initialization issues, we generate a prototype of our 3D scene using a text prompt, which we then optimize (Sec. 4.1).

3.2. Diffusion for text-to-3D generation

Diffusion models [21, 28, 58–61] are generative models which learn to map noise to data by iterative denoising over a family of increasingly smooth distributions p_t, corresponding to the noise level t. From a data point x and noise $\epsilon \sim \mathcal{N}(0, I)$, we can sample a noisy image as $x_t = \alpha_t x_0 + \sigma_t \epsilon$. The distributions p_t are parameterized by an increasing schedule σ_t for $t \in [0, 1]$ chosen such that p_0 approximates the data and p_1 is approximately Gaussian. By marginalizing over the data, we define $p_t(x_t) = \int p_t(x_t|x)p(x)dx$ where $p_t(x_t|x) = \mathcal{N}(\alpha_t x, \sigma_t^2 I)$ and $\alpha_t^2 = 1 - \sigma_t^2$. Given t, a diffusion model ϵ_θ is trained to predict the noise ϵ added to the image so that after training we have $\epsilon_\theta(x_t, t) \approx -\sigma_t \nabla_{x_t} \log p_t(x_t)$ approximates the scaled score of the smoothed data distribution [64]. We can then sample a pure noise image $x_1 \sim \mathcal{N}(0, I)$ and guide it backwards in time towards the data distribution using non-deterministic DDPM [21] or deterministic DDIM sampling [59], among others [28, 60, 61].

Often, the data distribution is conditional on quantities such as text T and images I, so the denoiser takes the form $\epsilon_\theta(x_t, I, T)$ where we absorb the timestep into x_t. In the conditional case, adherence to conditioning is ensured through Classifier-Free Guidance (CFG), whereby the predicted noise is parameterized as the difference between conditional and unconditional predictions [4, 22]:

$$\begin{aligned} \tilde{e}_\theta(x_t, I, T) = \ & e_\theta(x_t, \emptyset, \emptyset) \\ & + S_I \cdot (e_\theta(x_t, I, \emptyset) - e_\theta(x_t, \emptyset, \emptyset)) \\ & + S_T \cdot (e_\theta(x_t, I, T) - e_\theta(z_t, I, \emptyset)) \end{aligned} \tag{2}$$

where $\emptyset$ indicates no conditioning, and the values S_I and S_T are the guidance weights for image and text, dictating fidelity towards the respective conditions.

Distilling text-to-image diffusion models for text-to-3D generation of object-level data has enjoyed great success since the introduction of Score Distillation Sampling (SDS) [46, 65]. Given a text prompt T and a text-conditioned denoiser $\epsilon_\theta(x_t, T)$, SDS optimizes a 3D model by denoising noised renderings. Given a rendering from a 3D model x, we sample a timestep and corresponding x_t. Considering $\hat{x} = \frac{1}{\alpha_t}(x_t - \sigma_t \epsilon_\theta(x_t, T))$ as the detached one-step prediction of the denoiser, SDS is equivalent to minimizing [76]:

$$L_{\text{sds}} = \mathbb{E}_{t,\epsilon} \left[w(t) \left\| x - \hat{x} \right\|_2^2 \right] \tag{3}$$

where $w(t)$ is a time-dependent weight over all cameras with respect to the parameters of the 3D representation, and the distribution of t determines the strength of added noise.

4. Method

We now describe our technique in detail, which broadly consists of three stages: **initialization** (Fig. 3, Sec. 4.1); **inpainting** (Fig. 4, Sec. 4.2) with **depth distillation** (Fig. 4,

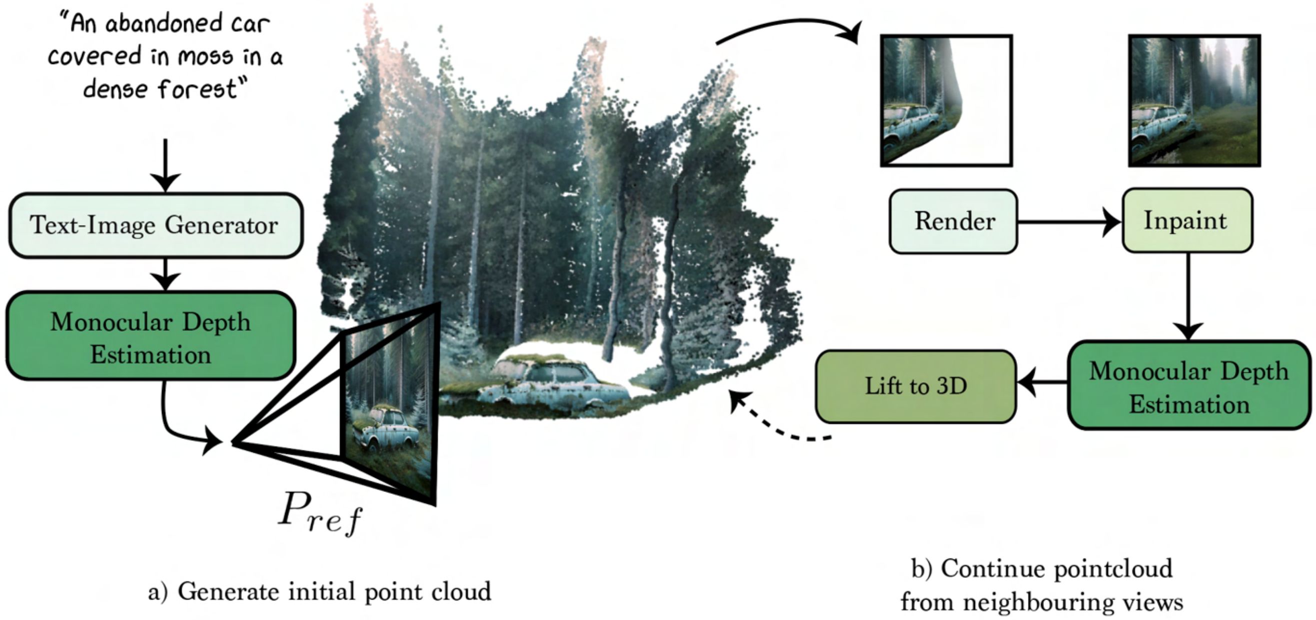

"An abandoned car covered in moss in a dense forest"
Text-Image Generator
Monocular Depth Estimation
P_{ref}
a) Generate initial point cloud
Render
Inpaint
Monocular Depth Estimation
Lift to 3D
b) Continue pointcloud from neighbouring views

Figure 3. **Point Cloud for 3D Gaussian Splatting Initalization.** We initialize our 3D Gaussian Splatting optimization by leveraging 2D diffusion priors. As shown in a), we first create a reference image I_{ref} from the text prompt T_{ref} at a pre-defined pose P_{ref}, and use a monocular depth estimator to convert it to 3D. b) The point cloud is then rendered from neighboring views and expanded to fill in missing regions, as described in Sec. 4.1.

Sec. 4.3); and **finetuning** (Fig. 4, Sec. 4.4). Given a text-prompt T_{ref} and camera poses, we initialize the scene-level 3DGS representation $\{\Theta_i\}_{i=1}^{N}$ leveraging 2D diffusion models and monocular depth priors, along with the computed *occlusion volume* (Sec. 4.1). With this robust initialization, we leverage inpainting diffusion models to fill in disoccluded regions and make a complete 3D scene (Sec. 4.2). In this stage, we also incorporate diffusion depth prior models for better geometry and faster convergence (Sec. 4.3). Finally, we finetune the model with a sharpness filter on sampled images to obtain cohesive 3D samples (Sec. 4.4).

4.1. Initializing a scene-level 3D representation

Our technique utilizes a text-conditioned optimization procedure to obtain 3DGS scenes, making good initializations essential. A common strategy in this setting is to initialize with a sphere [33, 46] or a low fidelity 3D model provided either by the user [10] or generated via pretrained models [26, 44]. However, such initializations are far from the spatial complexity expected of scenes. Hence, we leverage pretrained 2D priors to create a good initialization, as shown in Fig. 3.

Concretely, we follow an iterative process consisting of image generation, and monocular depth estimation, to build our initial point cloud. We first generate a reference image of the scene I_{ref} from the text prompt T_{ref} with state-of-the-art text-to-image-models. We then use a monocular depth model [29] $\mathcal{D}$ to lift this image to a 3D pointcloud $\mathcal{P}$ from a pre-defined camera pose P_{ref}. However, depending on the generated image, the extent of the pointcloud can vary widely. To make the initialization more robust, we *outpaint* I_{ref} by moving the camera in a small neighborhood of P_{ref} to poses P_{aux}, and add points to newly visible areas, similar to [23, 50]. We use an inpainting diffusion model [51] to fill-in these regions and lift this image to 3D using $\mathcal{D}$, while updating $\mathcal{P}$.

Given this initial point cloud $\mathcal{P}$, we then precompute the *occlusion volume* $\mathcal{O}$, which is the set of voxels within the scene bounding box which are occluded by the existing points in $\mathcal{P}$ from P_{ref}. We then define the initialization of our 3DGS means as

$$\{\mu_i\}_{i=1}^{N} = \mathcal{P} \cup \mathcal{O}. \tag{4}$$

More details about this process can be found in the supplement.

4.2. Inpainting diffusion models for text-to-3D

Since the pointcloud $\mathcal{P}$ is generated from a a single view and its depth, viewing it from other viewpoints exposes large holes in disoccluded regions. This motivates our formulation of scene generation as an *inpainting* problem,

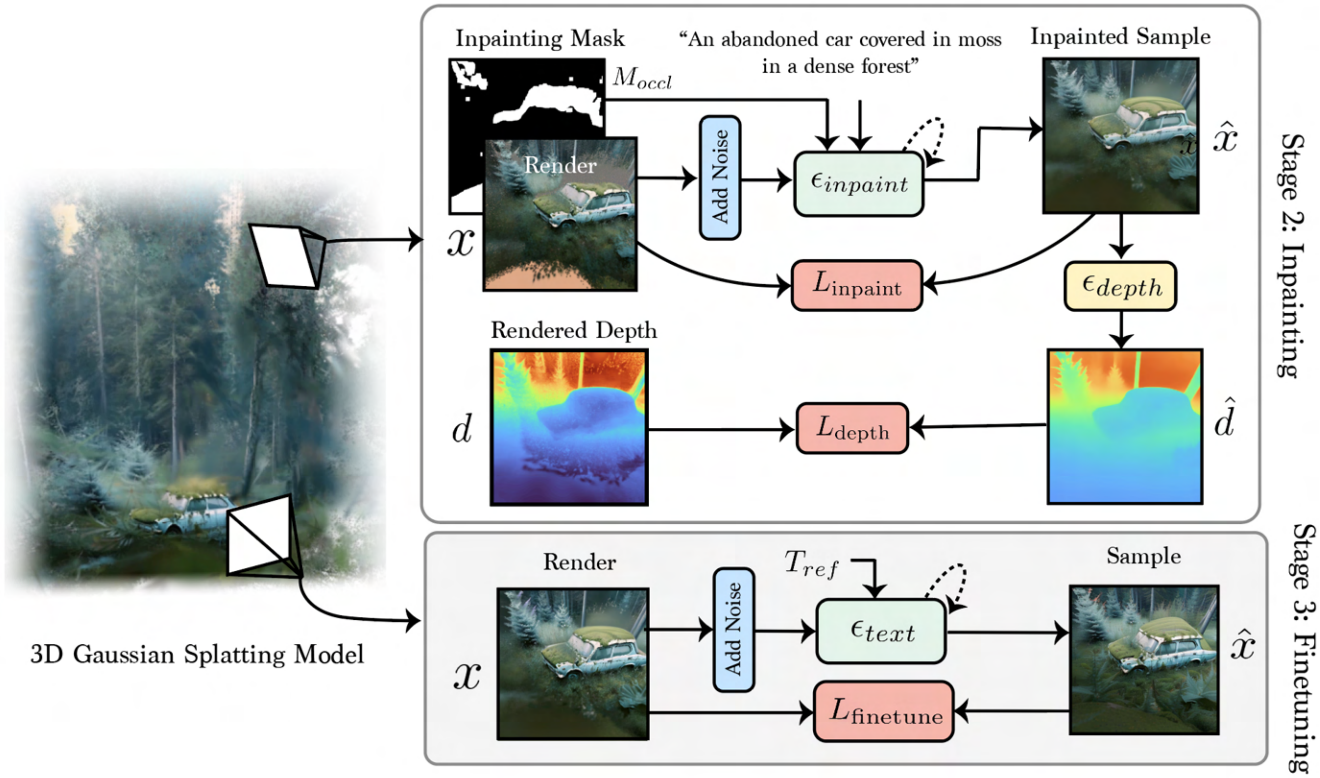


Figure 4. **Overview of our technique.** Our technique first uses a text prompt and an image to build a point cloud (Sec. 4.1) as shown in Fig. 3, which is then completed during the inpainting stage (Sec. 4.2) with an additional depth diffusion prior (Sec. 4.3), and finally a fine-tuning stage (Sec. 4.4) to improve the scene's coherence.

which we tackle using 2D inpainting diffusion models. Our 2D inpainting diffusion model $\epsilon_{\text{inpaint}}$ [51] takes as input a noisy 3DGS rendering x_t, the text prompt T_{ref}, an occlusion mask M_{occl}, and the original render x. Sampling from this model results in images $\hat{x}$ which fill in the holes in the point cloud rendering while preserving the general structure of the 3D scene, as shown in Fig. 4.

Creating the Inpainting Mask. To compute the conditioning quantities for $\epsilon_{\text{inpaint}}$ from a target camera, we leverage the point cloud $\mathcal{P}$ and the precomputed occlusion volume. Specifically, we render the point cloud and occlusion volume and set all components of M_{occl} for which the occlusion volume is visible from the target to 0, and 1 otherwise. Note that this handles cases such as the point cloud occluding itself. We also compute the masked point cloud render I_{pc} and a mask M_{val} which is set to 1 for those pixels with corresponding geometry and 0 otherwise.

Computing the Inpainting Loss. We use a 2D inpainting diffusion model which operates in latent space to construct an inpainting loss that provides 3D guidance. The latent diffusion model is defined by its encoder $\mathcal{E}$, decoder $\mathcal{D}$, and the denoising diffusion model $\epsilon_{\text{inpaint}}$. We render an image x with the initialized 3DGS model, and encode it to obtain a latent z, where $z = \mathcal{E}(x)$. We then add noise to this latent, yielding z_t, corresponding to a randomly sampled timestep t from the diffusion model's noise schedule. Using these quantities, we take multiple DDIM [59] steps from z_t to compute a clean latent $\hat{z}$ corresponding to the inpainted image.

We define our inpainting loss in both latent space and image space, by additionally decoding the predicted latent to obtain $\hat{x} = \mathcal{D}(\hat{z})$. We compute the L2 loss between the latents of the render and sample, as well as an L2 and LPIPS perceptual [74] loss between the rendered image and the decoded sample. To prevent edits outside of the inpainted region, we also add an anchor loss on the unmasked region of x, as the L2 difference between x and original point cloud render I_{pc}. Our final inpainting loss is

$$L_{\text{inpaint}} = \lambda_{\text{latent}}||z - \hat{z}||_2^2 + \lambda_{\text{image}}||x - \hat{x}||_2^2 \\ + \lambda_{\text{lpips}}\text{LPIPS}(x, \hat{x}) + \lambda_{\text{anchor}}||M_{\text{pc}}(x - I_{\text{pc}})||_2^2 \tag{5}$$

with λ weighting the different terms and $M_{\text{pc}} = (M_{\text{val}} \ \& \ M_{\text{occl}})$ is the bitwise and operation between computed masks. In practice, we also dilate M_{occl} to improve cohesion at mask boundaries. Further, we find it essential

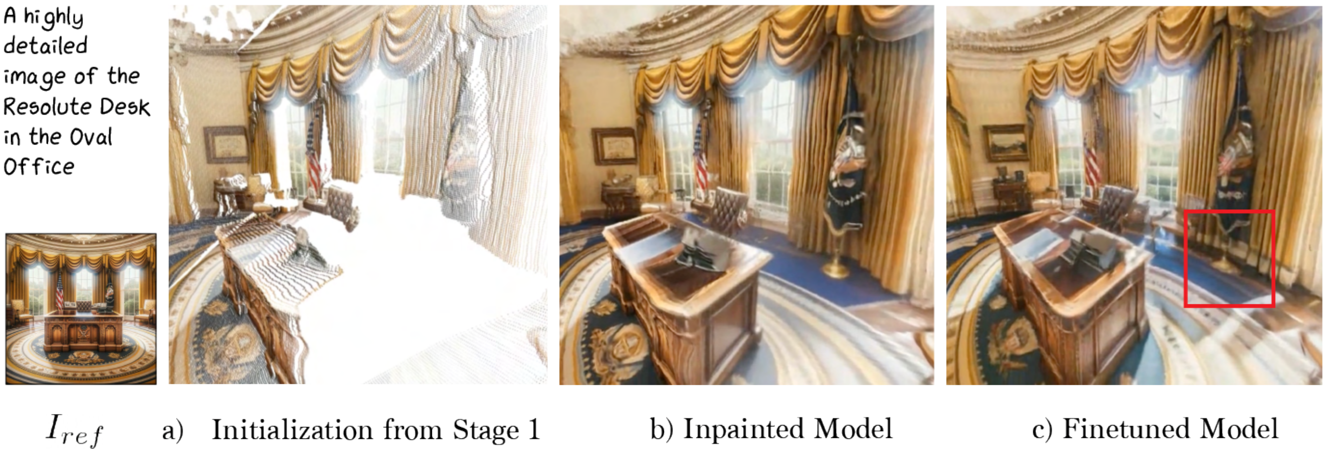

A highly detailed image of the Resolute Desk in the Oval Office
I_{ref}
a) Initialization from Stage 1
b) Inpainted Model
c) Finetuned Model

Figure 5. **Progression of 3D Model after each stage.** In this figure, we show how the 3D model changes after each stage in our pipeline. As shown in a) Stage 1 (Sec. 4.1 creates a point cloud with many empty regions. In b), we show the subsequent inpainted model from Stage 2 (Sec. 4.2). Finally, the fine-tuning stage (Sec. 4.4) refines b) to produce the final model, with greater cohesion and sharper detail.

to mask the latent-space L2 loss, to prevent unwanted gradients outside the masked region.

4.3. Depth diffusion for text-to-3D

Recently, image-conditional depth diffusion models have shown state-of-the-art results [29] for relative depth estimation by finetuning a large-scale pre-trained RGB prior [51] on depth datasets. In this section, we show how to distill the knowledge from such methods for text-to-3D synthesis. We consider a depth diffusion model ϵ_{depth} which is conditioned by images and text.

As mentioned in the previous section, during training, we can draw samples $\hat{x}$ using the inpainting diffusion model $\epsilon_{\text{inpaint}}$ applied to a noisy rendering of the current scene. Our insight is to use these clean samples as the conditioning for the depth diffusion model (shown in Fig. 4). Starting from pure noise $d_1 \sim \mathcal{N}(0, I)$, we predict the normalized depth using DDIM sampling [59]. We then compute the (negated) Pearson Correlation between the rendered depth and sampled depth:

$$L_{\text{depth}} = -\frac{\sum (d_i - \frac{1}{n}\sum d_k)(\hat{d}_i - \frac{1}{n}\sum \hat{d}_k)}{\sqrt{\sum (d_i - \frac{1}{n}\sum d_k)^2 \sum (\hat{d}_i - \frac{1}{n}\sum \hat{d}_k)^2}} \quad (6)$$

where d is the rendered depth from the 3DGS model and n is the number of pixels.

4.4. Optimization and Finetuning

The final loss for the first training stage of our pipeline is thus:

$$L_{\text{init}} = L_{\text{inpaint}} + L_{\text{depth}}. \quad (7)$$

After training with this loss, we have a 3D scene that roughly corresponds to the text prompt, but which may lack cohesiveness between the reference image I_{ref} and the inpainted regions. To remedy this, we incorporate an additional lightweight finetuning phase. In this phase, we utilize a vanilla text-to-image diffusion model ϵ_{text} personalized for the input image [15, 37, 48, 52]. We compute $\hat{x}$ using the same procedure as in Sec. 4.2, except with ϵ_{text}. The loss L_{text} is the same as Eq. (5), except with the $\hat{z}$ and $\hat{x}$ sampled with this finetuned diffusion model ϵ_{text}.

To encourage sharp details in our model, we use a lower noise strength than the inpainting stage and uniformly sample the timestep t within this limited range. We also propose a novel sharpening procedure, which improves the sharpness of our final 3D model. Instead of using $\hat{x}$ to compute the image-space diffusion loss introduced earlier, we use $\mathcal{S}(\hat{x})$, where $\mathcal{S}$ is a sharpening filter applied on samples from the diffusion model. Finally, to encourage high opacity points in our 3DGS model, we also add an opacity loss $L_{opacity}$ per point that encourages the opacity of points to reach either 0 or 1, inspired by the transmittance regularizer used in Plenoxels [16].

The combined loss for the fine-tuning stage can be written as:

$$L_{\text{finetune}} = L_{\text{text}} + \lambda_{opacity} L_{\text{opacity}}, \quad (8)$$

where $\lambda_{opacity}$ is a hyperparameter controlling the effect of the opacity loss.

4.5. Implementation Details

Point Cloud Initialization. We implement the point cloud initialization stage (Sec. 4.1) in Pytorch3D [49], with Stable

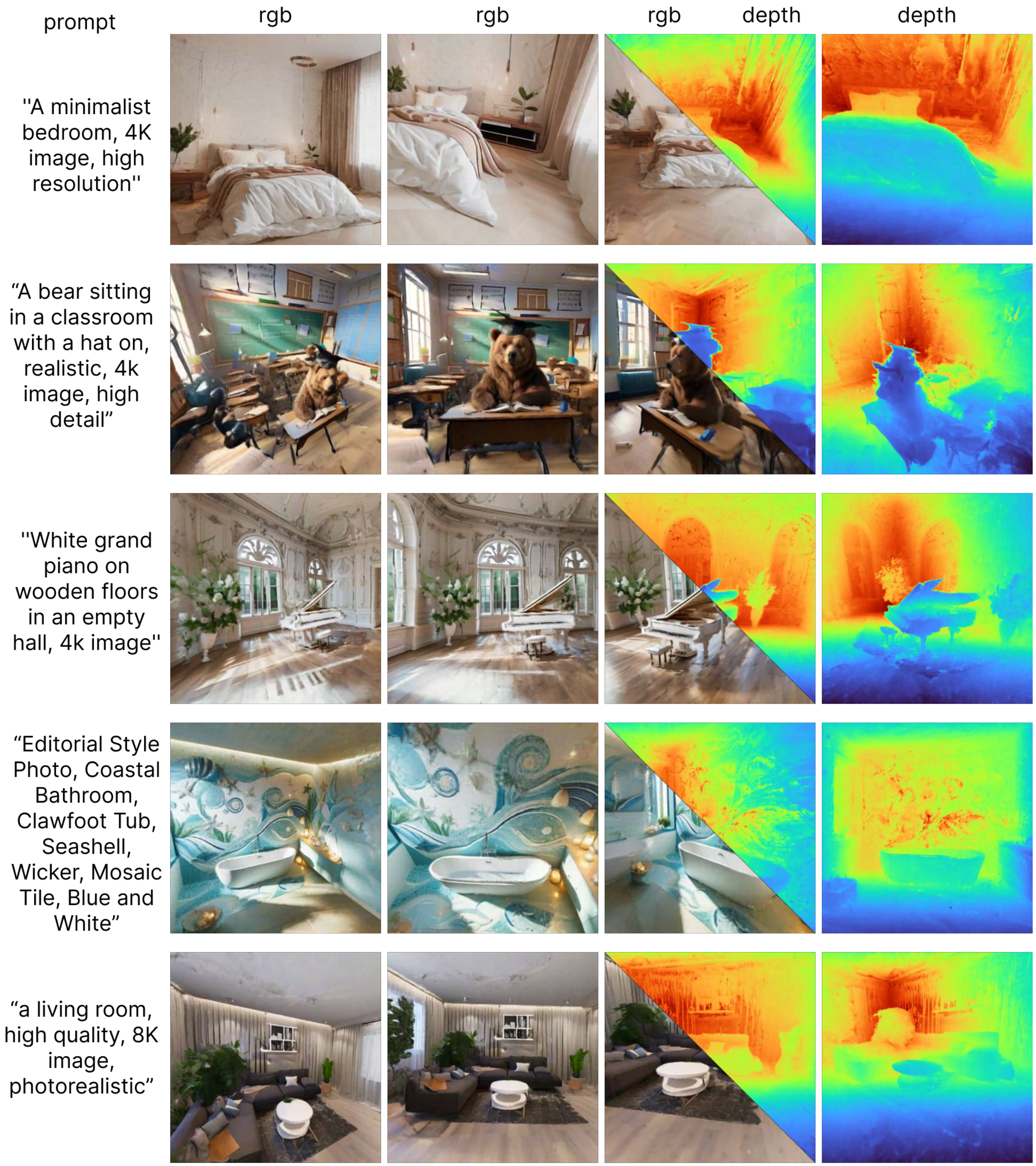

prompt
rgb
rgb
rgb
depth
depth
''A minimalist bedroom, 4K image, high resolution''
“A bear sitting in a classroom with a hat on, realistic, 4k image, high detail”
''White grand piano on wooden floors in an empty hall, 4k image''
“Editorial Style Photo, Coastal Bathroom, Clawfoot Tub, Seashell, Wicker, Mosaic Tile, Blue and White”
“a living room, high quality, 8K image, photorealistic”

Figure 6. We show the main qualitative results from our proposed method. In the left column, we show the input prompt. In the next two columns, we show the renderings from our 3D model from different viewpoints. In the fourth column, we show the level of agreement between rendering and geometry by a split view of the rendering and depth. Finally, in the last column, we show the depth map.

Diffusion [51] as our inpainting model. To lift the generated samples to 3D, we use a high quality monocular depth estimation model, Marigold [29], as a relative depth estimator. Since it lacks a defined scale, we align these predictions with the absolute metric depth predicted with DepthAnything [69].

Inpainting and Finetuning Stage. Our inpainting (Sec. 4.2) and finetuning stages (Sec. 4.4) are implemented in NeRFStudio [62] using the official implementation of Gaussian Splatting [30]. We use Stable Diffusion 2.0 as ϵ_{text} and its inpainting variant as $\epsilon_{\text{inpaint}}$, building on threestudio [18] to define our diffusion-guided losses. Further, we use Marigold [29] as our depth diffusion model. During the inpainting stage, we set the guidance weight for image and text conditioning of $\epsilon_{\text{inpaint}}$ as 1.8 and 7.5 respectively. We find that a high image guidance weight produces samples with greater overall cohesion. We also use a guidance weight of 7.5 for the text-to-image diffusion model ϵ_{text} during the finetuning stage. During the inpainting stage, we sample the timestep t from $\mathcal{U}(0.1, 0.95)$ and during the finetuning stage from $\mathcal{U}(0.1, 0.3)$. We train the model for 15000 iterations during the inpainting stage, which takes 8 hours to run on a 24GB Nvidia A10 GPU. The finetuning stage is run for 3000 iterations and takes 2.5 hours to finish on the same GPU.

5. Results

We evaluate our technique on a custom dataset of prompts, reference images, and associated camera poses P_i, selected to showcase parallax and disocclusion in the 3D scene. We built this dataset by creating a set of 11 prompts, and having a human expert manually choose camera poses using a web-viewer [62], by displaying a scene prototype obtained as in Sec. 4.1. No such dataset already exists for this problem, as existing text-3D techniques [46, 66] operate with simple camera priors, such as sampling on a sphere, with cameras looking at a central object, or outward in the case of scenes [66]. This can limit the scale of the obtained scene significantly.

5.1. Qualitative Results

Our method's results are shown on 5 representative examples in Fig. 6, demonstrating effective 3D scene sampling across various settings (indoor, outdoor) and image styles (realistic, fantasy, illustration). We highlight the significant difference between views and the consistency of rendering and geometry in split views, underscoring our method's use of inpainting and depth diffusion priors.

5.2. Comparisons

We compare our technique with the state-of-the-art baselines for text-to-3D synthesis in Fig. 7. Namely, we compare against DreamFusion [46], ProlificDreamer [66], and Text2Room [23]. We choose these baselines as they represent the state-of-the-art performance for 3D object and scene synthesis. Both ProlificDreamer and DreamFusion learn oversaturated scenes with incorrect geometry and scene structure. On the other hand, Text2Room fails to reconstruct non-room scenes, resulting in degenerate renderings which essentially forget the input prompt during its iterative mesh creation process. To ensure a fair comparison, we also incorporate prompts from the baselines Text2Room [23], ProlificDreamer [66], and Dreamfusion [46].

5.3. User Study

In line with prior work [66], we conduct a user study to compare the results from our technique with the established baselines. We conducted the study on Amazon Mechanical Turk and recruited 20 participants with a 'master' qualification to compare with each baseline, while following several guidelines outlined in [5]. All participants were provided a brief definition about "3D scenes" and asked to make preferential decisions based on three criteria: 1). the quality of the videos 2). how 3D the scene looks and 3). alignment to the associated prompt.

For comparison with ProlificDreamer [66] and DreamFusion [46], we showed participants side-by-side videos comparing our method to the baseline. For fairness, we use the same camera trajectory in all videos. The order of the videos was also randomized to prevent any biases due to the order of presentation. The user's preference was logged along with a brief explanation.

When comparing with Text2Room [23], instead of a video, we showed users side-by-side sets of three multiview images for each prompt, due to the degeneracy of the output mesh far from the starting camera pose. The users preferred triplet was logged along with their brief explanation. The images we showed looked slightly left and right of the reference pose P_{ref}.

5.3.1 Common Themes

All study participants were asked to justify their preferences for one 3D scene over the other after making their choice. Participants were not informed about the names or the nature of any technique. We also adopted method-neutral language to avoid biasing the user to prefer any particular technique. We find that their provided reasoning closely aligns with several noted limitations of the baselines, which we discuss further:

ProlificDreamer [66] can produce cloudy results. Several participants described the NeRF renders as containing "moving clouds", a "hazy atmosphere", and a "blotch of colours". This can likely be attributed to the presence of floaters in the model, which is evident in the noisy depth maps shown in Appendix A.2. In contrast, participants de-

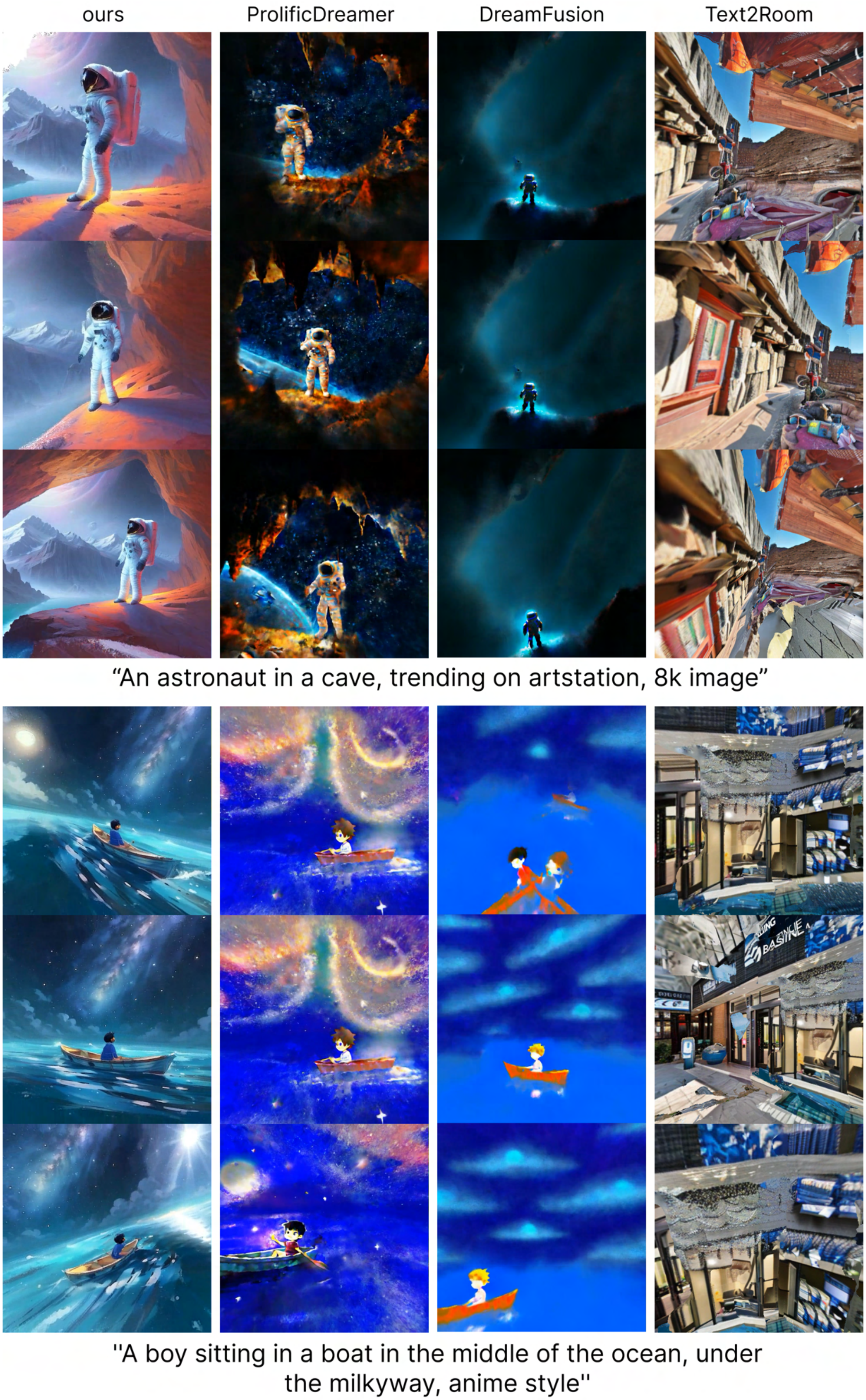

ours
ProlificDreamer
DreamFusion
Text2Room
“An astronaut in a cave, trending on artstation, 8k image”
"A boy sitting in a boat in the middle of the ocean, under the milkyway, anime style"

Figure 7. We show the main qualitative comparison to the state-of-the-art baselines ProlificDreamer [66], DreamFusion [46], and Text2Room [23] on the two prompts specified.

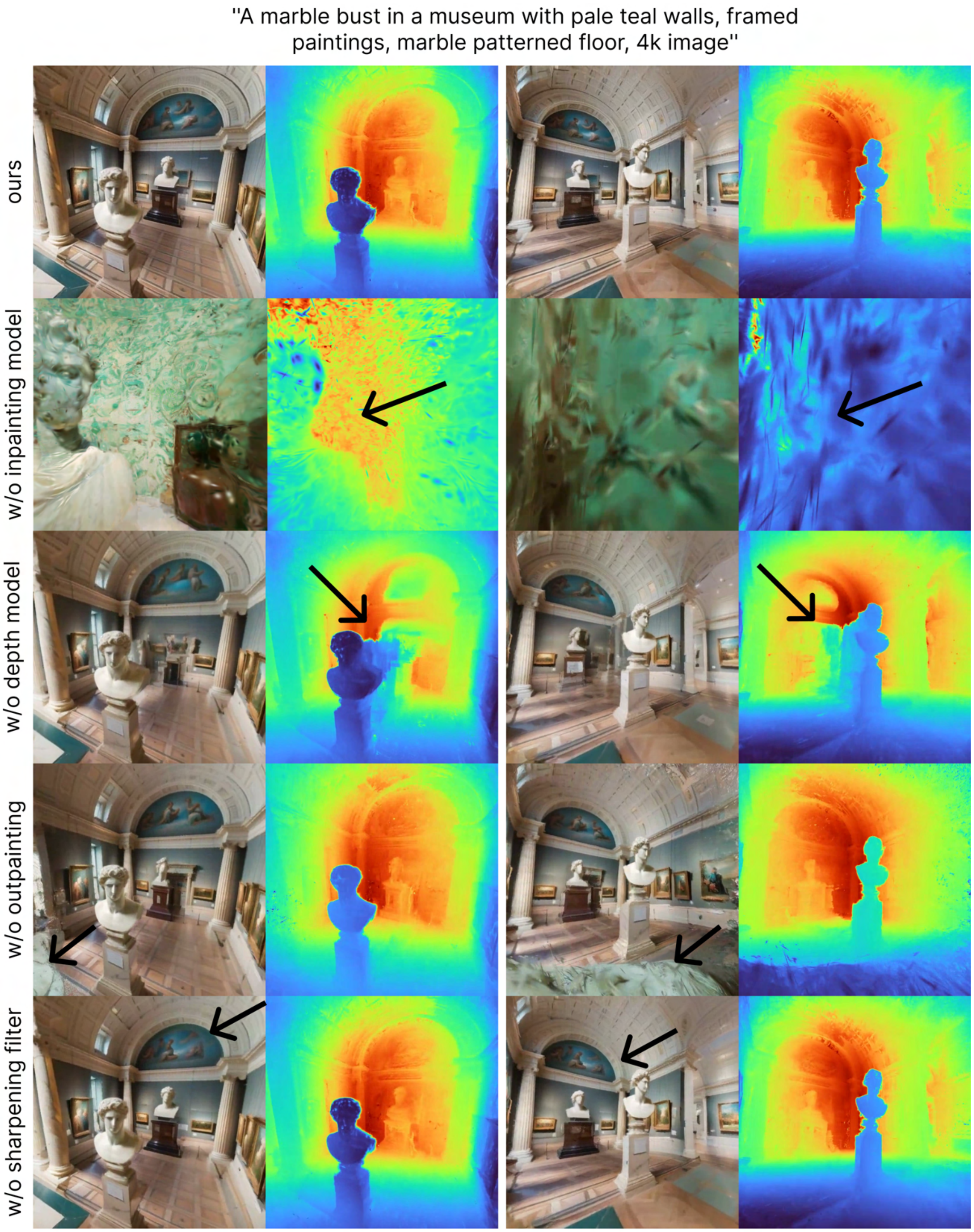

"A marble bust in a museum with pale teal walls, framed paintings, marble patterned floor, 4k image"
ours
w/o inpainting model
w/o depth model
w/o outpainting
w/o sharpening filter

Figure 8. We show the qualitative results of our model and its ablations. Arrows indicate failures in the ablated models. Please see Sec. 5.5 for a detailed discussion of the ablated components and their respective importance.

Table 1. **Results of user study.** This table shows the percentage of comparisons where our technique was preferred over the mentioned baseline. Our approach is significantly preferred over existing baselines.

Ours vs. ProlificDreamer[66]	Ours vs. DreamFusion[46]	Ours vs. Text2Room[23]
93.63%	**92.27%**	**87.72%**

scribed our method as "clean and crisp when it comes to the colors and sharpness of the pixels" and looking realistic, without the presence of over-saturated colors.

Dreamfusion [46] lacks realism and detail. Feedback from users when comparing with DreamFusion often mirrored feedback from the ProlificDreamer comparison, referencing a lack of realism and detail in the produced renders. One participant said "[Our technique] is more crisp and does a better job with the content quality.", while the Dreamfusion result can "feel disjointed". Another participant described a render as having a "distorted looking background". In contrast to these issues, our technique synthesizes realistic models with high detail and high-quality backgrounds, with minimal blurriness.

Text2Room [23] can produce messy outputs. A common theme across feedback regarding Text2Room was that it often looked like a mess, sometimes with a "strange distortion". One user writes that our result is "less busy and fits the description". Another common reason users cited when choosing our technique was the adherence to the input prompt, with Text2Room often missing key objects that are expected for an associated prompt. Our technique, however, is capable of producing highly coherent outputs that are faithful to the reference prompt and produce high-quality renderings from multiple views.

5.4. Quantitative Metrics

We also provide a quantitative comparison between our method and baselines using CLIP [47]. Specifically, for each method, we compute the CLIP alignment between the rendered image at P_{ref} and the corresponding prompt for all scenes. For Text2Room, this pose is the input pose from which the scene is generated. The results can be seen in Tab. 2, where our method shows better alignment to the text prompt versus the baselines. Note that this is due to our method's ability to inherit the high-fidelity generations of 2D diffusion models.

Ours	Text2Room	DreamFusion	ProlificDreamer
33.14	25.10	30.28	30.69

Table 2. CLIP alignment for renderings of scenes for our method and the baselines. Scores are scaled by 100.

5.5. Ablations

We verify the proposed contributions of our method by ablating the key components in Fig. 8 with the specified prompt. In the first row, we show our base method. In the second row, we then show the importance of the inpainting diffusion model from Sec. 4.2—removing this component results in high-variance gradients which cause the 3DGS representation to diverge into a meaningless cloud. In the third row, we remove the depth prior and L_{depth}; this results in a 3DGS representation with incorrect geometry which doesn't reflect the renderings. Next, in the fourth row, we initialize the point cloud with only our reference image I_{ref}, without sampling from neighbouring poses P_{aux}, which makes the regions outside of the original camera frustum underdetermined and low quality as evidenced by the images. Finally, in the last row, we show our result without applying a sharpening filter on the sampled images which results in blurrier renders with less detail.

6. Application: Single image to 3D

Our technique extends to creating 3D scenes from a single image, as shown in Fig. 9, by using a user's image as I_{ref} and a text-prompt T_{ref} obtained using an image-captioning model like GPT-4. Our pipeline remains the same and can effectively fill in occluded areas and generate realistic geometry for unseen regions.

7. Conclusion

We have proposed **RealmDreamer**, a method for generation of forward-facing 3DGS scenes leveraging inpainting and depth diffusion priors. Our method provides higher-quality results than existing baselines along with higher-fidelity geometry on more challenging camera trajectories. Still, some limitations remain; our method takes several hours, and produces blurry results for complex scenes with high occlusions. Future work may explore more efficient diffusion models for faster training, and more complex conditioning schemes for full 360-degree generations.

Ethical Considerations

While we use pretrained models for all components of our pipeline, it is important to acknowledge biases and ethical issues that stem from the training of these large-scale image generative models [51]. As these models are often trained on vast collections of internet data, they can reflect negative

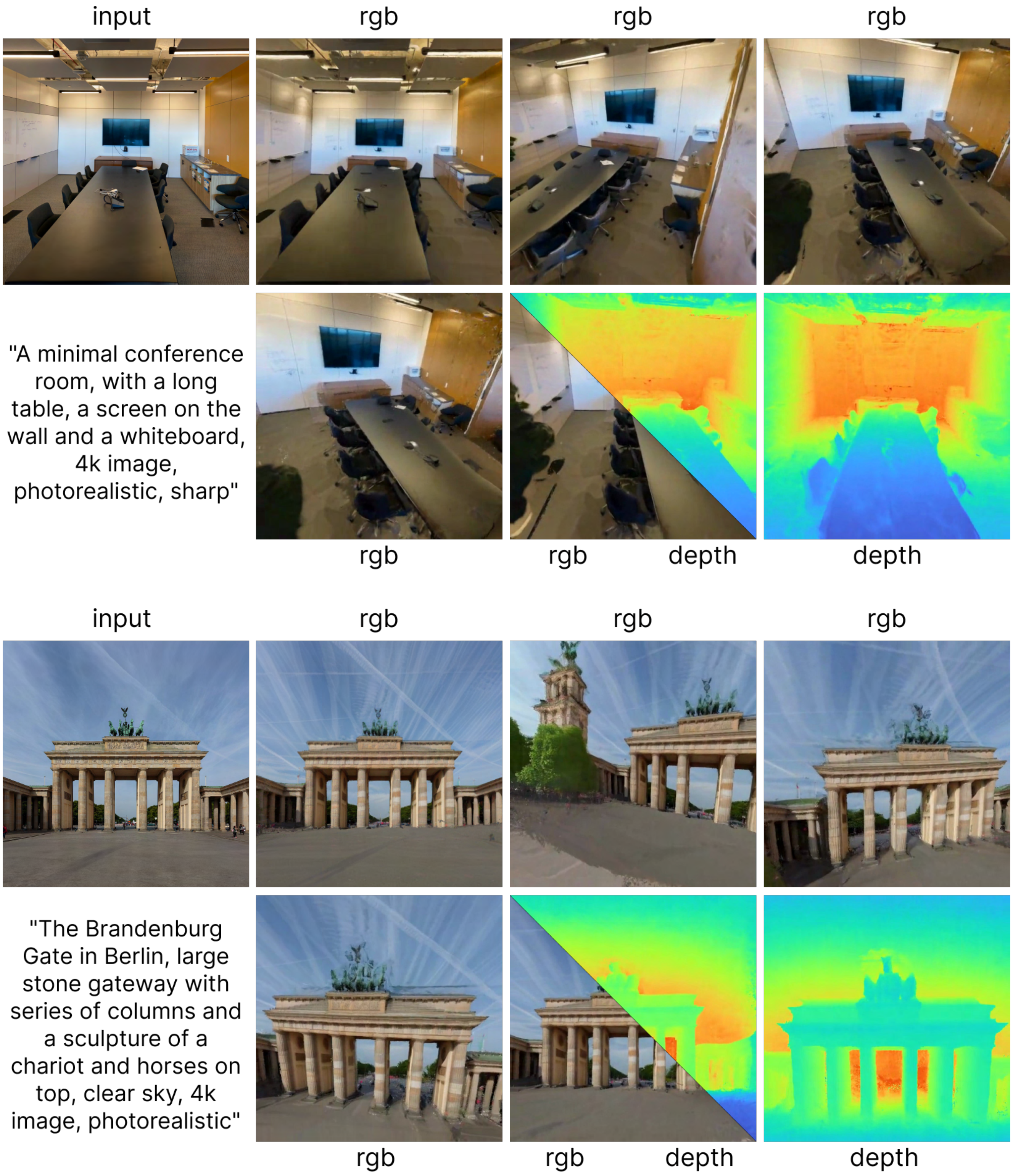

input
rgb
rgb
rgb
"A minimal conference room, with a long table, a screen on the wall and a whiteboard, 4k image, photorealistic, sharp"
rgb
rgb
depth
depth
input
rgb
rgb
rgb
"The Brandenburg Gate in Berlin, large stone gateway with series of columns and a sculpture of a chariot and horses on top, clear sky, 4k image, photorealistic"
rgb
rgb
depth
depth

Figure 9. **Result for single-image to 3D.** Using a provided image and a prompt obtained via an image captioning model, our technique can generate a 3D scene and fill in occluded regions.

biases and stereotypes against certain populations, as well as infringe on the copyright of artists and other creatives. It is essential to consider these factors when using these models and our technique broadly.

8. Acknowledgements

We thank Jiatao Gu and Kai-En Lin for early discussions, Aleksander Holynski and Ben Poole for later discussions. This work was supported in part by an NSF graduate Fellowship, ONR grant N00014-23-1-2526, NSF CHASE-CI Grants 2100237 and 2120019, gifts from Adobe, Google, Qualcomm, Meta, the Ronald L. Graham Chair, and the UC San Diego Center for Visual Computing.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017. 2

[2] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Irondepth: Iterative refinement of single-view depth using surface normal and its uncertainty. In *British Machine Vision Conference (BMVC)*, 2022. 17

[3] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023. 16

[4] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023. 3

[5] Zoya Bylinskii, Laura Mariah Herman, Aaron Hertzmann, Stefanie Hutka, and Yile Zhang. Towards better user studies in computer graphics and vision. *Found. Trends Comput. Graph. Vis.*, 15:201–252, 2022. 8

[6] Eric R Chan, Koki Nagano, Matthew A Chan, Alexander W Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. Generative novel view synthesis with 3d-aware diffusion models. *arXiv preprint arXiv:2304.02602*, 2023. 2

[7] Angel Chang, Will Monroe, Manolis Savva, Christopher Potts, and Christopher D Manning. Text to 3d scene generation with rich lexical grounding. *arXiv preprint arXiv:1505.06289*, 2015. 2

[8] Angel Chang, Manolis Savva, and Christopher D Manning. Learning spatial knowledge for text to 3d scene generation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 2028–2038, 2014. 2

[9] Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pages 100–116. Springer, 2019. 2

[10] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023. 2, 4

[11] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. *arXiv preprint arXiv:2309.16585*, 2023. 2

[12] Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. Luciddreamer: Domain-free generation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2311.13384*, 2023. 2

[13] Kevin Clark and Priyank Jaini. Text-to-image diffusion models are zero shot classifiers. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[14] Bob Coyne and Richard Sproat. Wordseye: An automatic text-to-scene conversion system. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 487–496, 2001. 2

[15] Congyue Deng, Chiyu Jiang, Charles R Qi, Xinchen Yan, Yin Zhou, Leonidas Guibas, Dragomir Anguelov, et al. Nerdi: Single-view nerf synthesis with language-guided diffusion as general image priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20637–20647, 2023. 2, 6

[16] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 6

[17] Jiatao Gu, Alex Trevithick, Kai-En Lin, Joshua M Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion. In *International Conference on Machine Learning*, pages 11808–11826. PMLR, 2023. 2

[18] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3d content generation. https://github.com/threestudio-project/threestudio, 2023. 8, 17

[19] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. *arXiv preprint arXiv:2303.12789*, 2023. 2

[20] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. 16

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 3

[22] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 3

[23] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7909–7920, October 2023. 4, 8, 9, 11, 16

[24] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen.

Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 2

[25] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 867–876, 2022. 2

[26] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 2, 4

[27] Yash Kant, Aliaksandr Siarohin, Michael Vasilkovsky, Riza Alp Guler, Jian Ren, Sergey Tulyakov, and Igor Gilitschenski. invs: Repurposing diffusion inpainters for novel view synthesis. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–12, 2023. 2

[28] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. 3

[29] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation, 2023. 4, 6, 8, 16, 17

[30] 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. 2, 8

[31] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching, 2023. 2, 16, 17

[32] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023. 2

[33] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 4

[34] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023. 2

[35] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453*, 2023. 2

[36] Jonathan Lorraine, Kevin Xie, Xiaohui Zeng, Chen-Hsuan Lin, Towaki Takikawa, Nicholas Sharp, Tsung-Yi Lin, Ming-Yu Liu, Sanja Fidler, and James Lucas. Att3d: Amortized text-to-3d object synthesis. *arXiv preprint arXiv:2306.07349*, 2023. 2

[37] Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Realfusion: 360deg reconstruction of any object from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8446–8455, 2023. 6

[38] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021. 2

[39] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12663–12673, 2023. 2

[40] 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. In *ECCV*, 2020. 2

[41] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Marcus A Brubaker, Jonathan Kelly, Alex Levinshtein, Konstantinos G Derpanis, and Igor Gilitschenski. Reference-guided controllable inpainting of neural radiance fields. *arXiv preprint arXiv:2304.09677*, 2023. 2

[42] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G Derpanis, Jonathan Kelly, Marcus A Brubaker, Igor Gilitschenski, and Alex Levinshtein. Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20669–20679, 2023. 2

[43] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6038–6047, 2023. 2

[44] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 2, 4

[45] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 16

[46] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 2, 3, 4, 8, 9, 11, 17

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2, 11

[48] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. Dreambooth3d: Subject-driven text-to-3d generation. *arXiv preprint arXiv:2303.13508*, 2023. 6

[49] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 6, 16

[50] Chris Rockwell, David F. Fouhey, and Justin Johnson. Pixelsynth: Generating a 3d-consistent experience from a single image. In *ICCV*, 2021. 4

[51] 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. 2, 4, 5, 6, 8, 11

[52] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023. 6, 16

[53] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 2

[54] Aditya Sanghi, Hang Chu, Joseph G Lambourne, Ye Wang, Chin-Yi Cheng, Marco Fumero, and Kamal Rahimi Malekshan. Clip-forge: Towards zero-shot text-to-shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18603–18613, 2022. 2

[55] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023. 2

[56] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 2

[57] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006. 3

[58] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 3

[59] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3, 5, 6, 16

[60] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 3

[61] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 3

[62] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023. 8

[63] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024. 2

[64] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011. 3

[65] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. *arXiv preprint arXiv:2212.00774*, 2022. 2, 3

[66] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 1, 2, 3, 8, 9, 11, 17

[67] Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. *arXiv preprint arXiv:2210.04628*, 2022. 2

[68] Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. Open-vocabulary panoptic segmentation with text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2955–2966, 2023. 2

[69] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 8, 16, 17

[70] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In *CVPR*, 2024. 2

[71] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Text2nerf: Text-driven 3d scene generation with neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 2024. 2

[72] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 2

[73] Qihang Zhang, Chaoyang Wang, Aliaksandr Siarohin, Peiye Zhuang, Yinghao Xu, Ceyuan Yang, Dahua Lin, Bolei Zhou, Sergey Tulyakov, and Hsin-Ying Lee. Scenewiz3d: Towards text-guided 3d scene composition. *arXiv preprint arXiv:2312.08885*, 2023. 2

[74] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 5

[75] Zhizhuo Zhou and Shubham Tulsiani. Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction. In *CVPR*, 2023. 2

[76] Junzhe Zhu, Peiye Zhuang, and Sanmi Koyejo. Hifa: High-fidelity text-to-3d generation with advanced diffusion guidance. In *The Twelfth International Conference on Learning Representations*, 2023. 3

[77] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa volume splatting. In *Proceedings Visualization, 2001. VIS'01.*, pages 29–538. IEEE, 2001. 3

A. Appendix

In this supplementary, we provide additional details concerning **RealmDreamer**, such as additional implementation details (Appendix A.1) and additional qualitative results (Appendix A.2).

A.1. Additional Implementation Details

We intend to open-source our code later on. In addition, we describe some key implementation details to assist reproducibility.

A.1.1 Point Cloud Generation

Image Generation. We generate our reference image I_{ref} using a variety of state-of-the-art text-image generation models, choosing between Stable Diffusion XL [45], Adobe Firefly, and DALLE-3 [3].

Depth Estimation. As mentioned earlier, we use Marigold [29] as our depth estimation model, with absolute depth obtained using DepthAnything [69]. We align the relative depth with this absolute depth by computing the linear translation that minimizes the least squares error between them. Since DepthAnything provides separate model weights for indoor and outdoor scenes, we use GPT-4 to decide which checkpoint to use by passing I_{ref} as input. When iteratively growing the point cloud, we follow Text2Room [23] and align the predicted depth with the ground truth depth rendered via Pytorch3D [49] for all regions with valid geometry. We additionally blur the edges of these regions to lower the appearance of seams at this intersection.

Growing the pointcloud beyond P_{ref}. After lifting the reference image I_{ref} to a pointcloud $\mathcal{P}$, we additionally create new points from neighbouring poses P_{aux}, as mentioned earlier. In practice, we notice that using the same prompt P_{ref} across all neighbouring poses P_{aux} can lead to poor results, as objects mentioned in the prompt get repeated. Hence, we use GPT-4 to compute a new suitable prompt that can represent the neighbouring views of P_{ref}. Specifically, we pass the reference image I_{ref}, the original prompt T_{ref} and ask GPT-4 to provide a new prompt T_{aux} that can be suitable for neighbouring regions. For instance, when viewing a *"car in a dense forest"*, T_{aux} may correspond to a *"dense forest"*.

A.1.2 Occlusion Volume Computation

We compute the occlusion volume $\mathcal{O}$ with Bresenham's line-drawing algorithm. First, we initialize an occupancy grid $\mathcal{G}$ using the point cloud $\mathcal{P}$ from stage 1. We also store whether any voxel is occluded with respect to P_{ref} within the same occupancy grid, initially settings all voxels as occluded. Then, we draw a line from the position of the reference camera T_{ref} to all voxels in the occupancy grid $\mathcal{G}$, iterating over the voxels covered by this line and marking all as *non-occluded* until we encounter an occupied voxel. Once the algorithm terminates, all voxels that are untouched by the line-drawing algorithm form our occlusion volume $\mathcal{O}$.

A.1.3 Optimization

Use of DDIM Inversion. During the inpainting and finetuning stage (Sec 4.2, 4.3 in the original paper), we find it helpful to obtain the noisy latent z_t using DDIM inversion [59], where $z = \mathcal{E}(x)$, x is the rendered image, and t is a timestep corresponding to the amount of noise added. This is similar to prior work on 2D/3D editing and synthesis using pre-trained diffusion models [20, 31]. We demonstrate the importance of doing so in Fig. 10, where DDIM inversion can significantly improve the detail in the optimized model. During the inpainting stage, we use 25 steps to sample an image from pure noise, and during finetuning, we use 100 steps.

Hyperparameter Weights. We set $\lambda_{\text{latent}} = 0.1, \lambda_{\text{anchor}} = 10000$ during the inpainting stage, and $\lambda_{\text{latent}} = 0.01, \lambda_{\text{anchor}} = 0$ during the finetuning stage. The other parameters are set as $\lambda_{\text{image}} = 0.01, \lambda_{\text{lpips}} = 100, \lambda_{\text{depth}} = 1000$, and $\lambda_{\text{opacity}} = 10$.

Use of Dreambooth during fine-tuning. While finetuning the output from stage 2, we use Dreambooth [52] to personalize the text-to-image diffusion model with the reference image I_{ref} and associated prompt T_{ref}. We find that this helps the final 3D model adhere closer to I_{ref} stylistically. We use the implementation of Dreambooth from HuggingFace and train at a resolution of 512x512 with a batch size of 2, with a learning rate of 1e-6 for 200 steps.

Gaussian Splatting. We initialize our gaussian splatting model during the inpainting stage, using the point cloud from stage 1, where each point is an isotropic gaussian, with the scale set based on the distance to its nearest neighbors. During the inpainting stage, we use a constant learning rate of 0.01 for rotation, 0.001 for the color, and 0.01 for opacity. The learning rate of the geometry follows an exponentially decaying scheduler, which decays to 0.00005 from 0.01 over 100000 steps, after 5000 warmup steps. Similarly, the scale is decayed to 0.0001 from 0.005 over 10000 steps, after 7000 warmup steps. During the finetuning stage, we use a constant learning rate of 0.01 for rotation, 0.001 for the color, 0.01 opacity, and 0.0001 for scale. We use an exponentially decaying scheduler for the geometry, which decays to 0.0000005 from 0.0001 over 3000 steps, after 750 warmup steps.

A.1.4 Comparison with baselines.

Text2Room [23] We use the official implementation of Text2Room on Github. To ensure a fair comparison, we es-

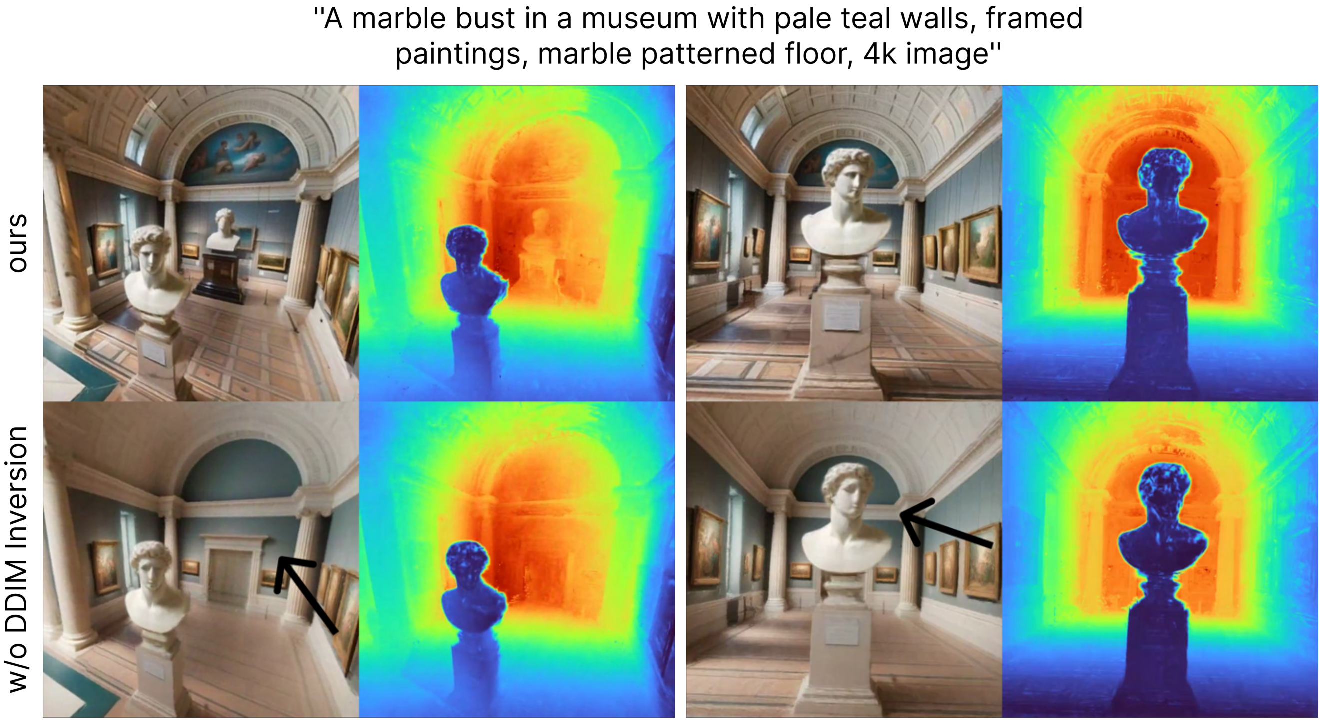

"A marble bust in a museum with pale teal walls, framed paintings, marble patterned floor, 4k image"
ours
w/o DDIM Inversion

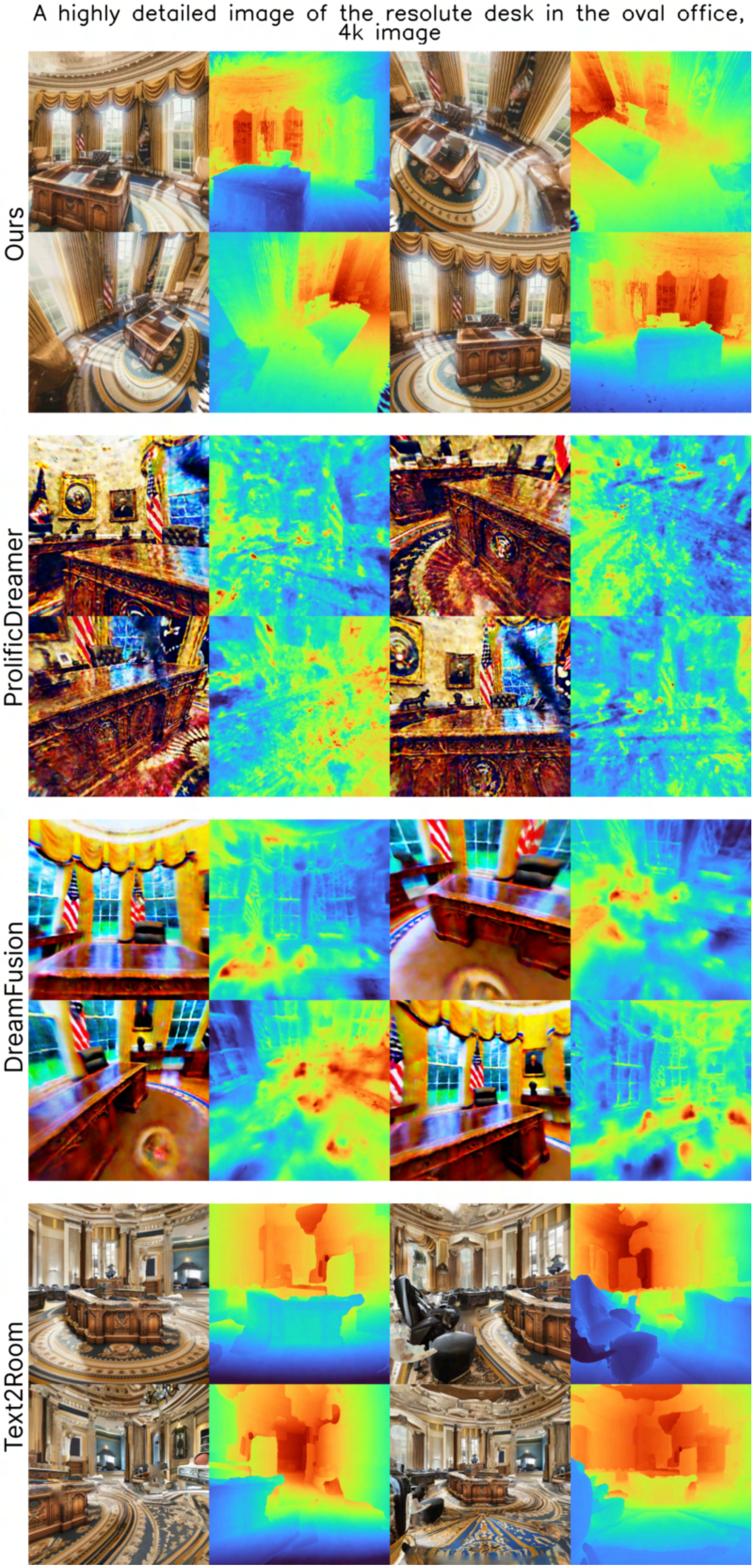
Figure 10. We ablate the importance of DDIM Inversion. Like other methods [31], we find that the inversion allows more details to be synthesized by our method.

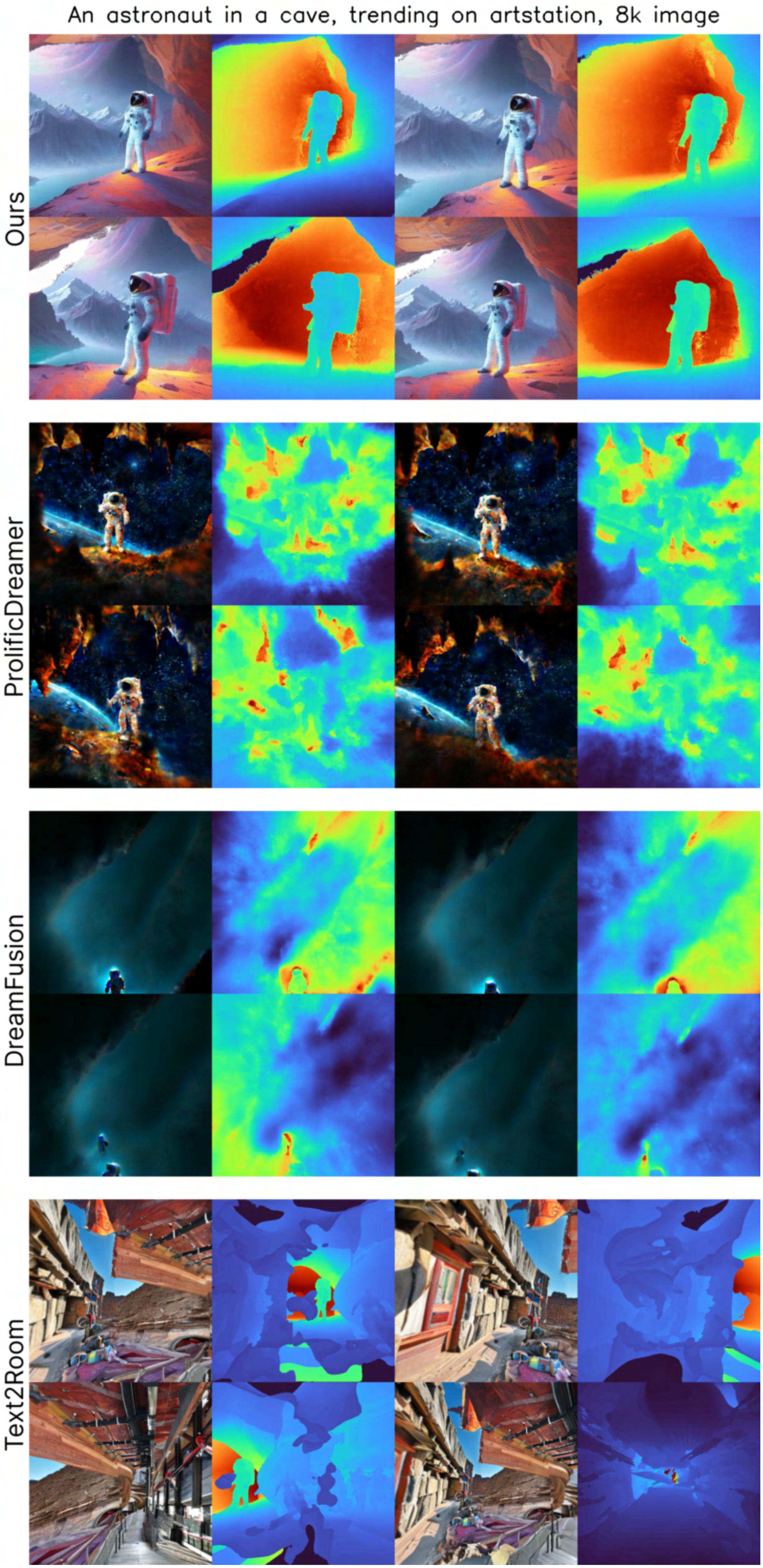
timate depth using Marigold [29] and DepthAnything [69] as in our technique, replacing the original IronDepth [2]. The rest of the pipeline is kept the same.

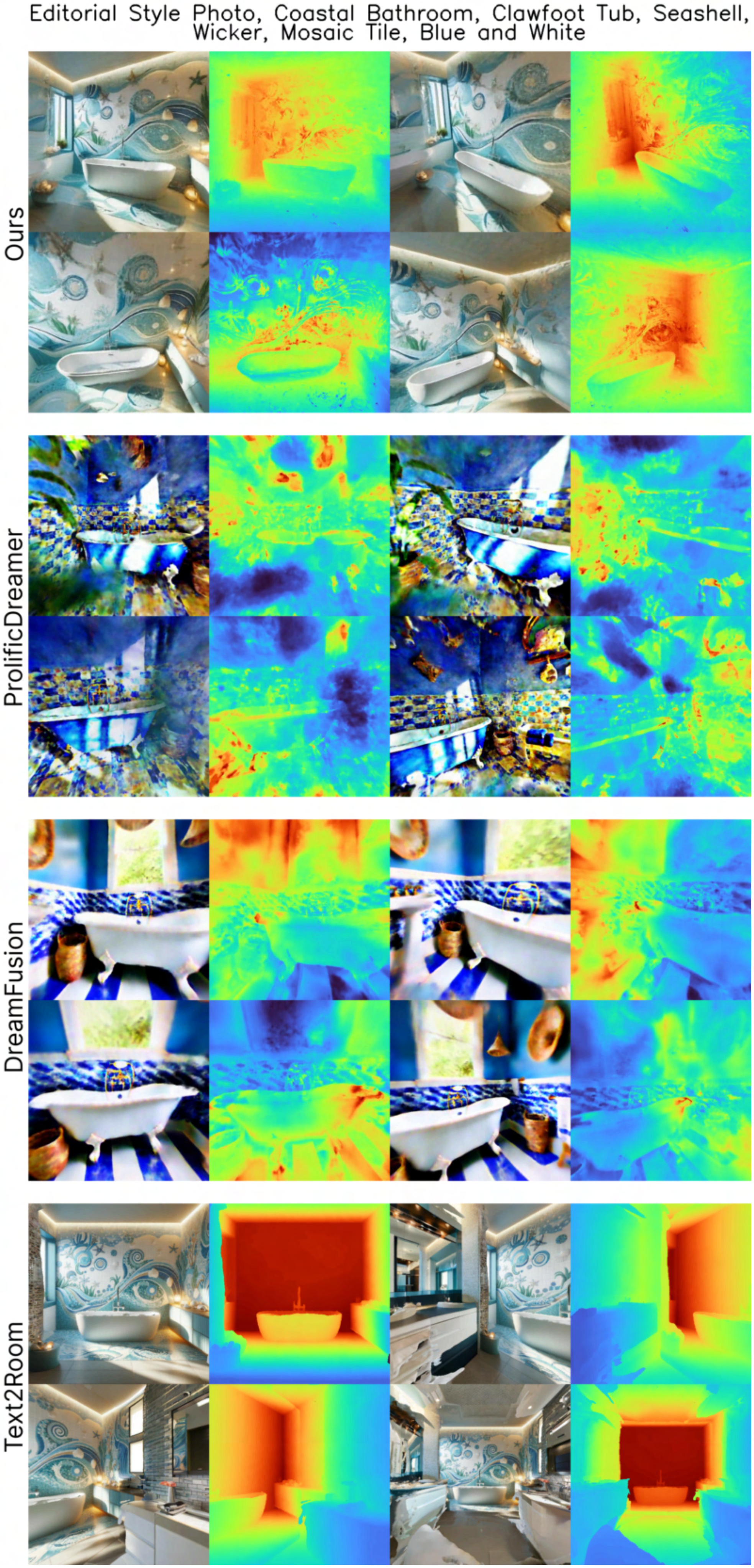
ProlificDreamer [66] and Dreamfusion [46] We use the implementation of these baselines provided in threestudio [18] and use their recommended parameters, training for 25k and 10k steps, respectively. To ensure a fair comparison, we use the same poses for these baselines as our technique.

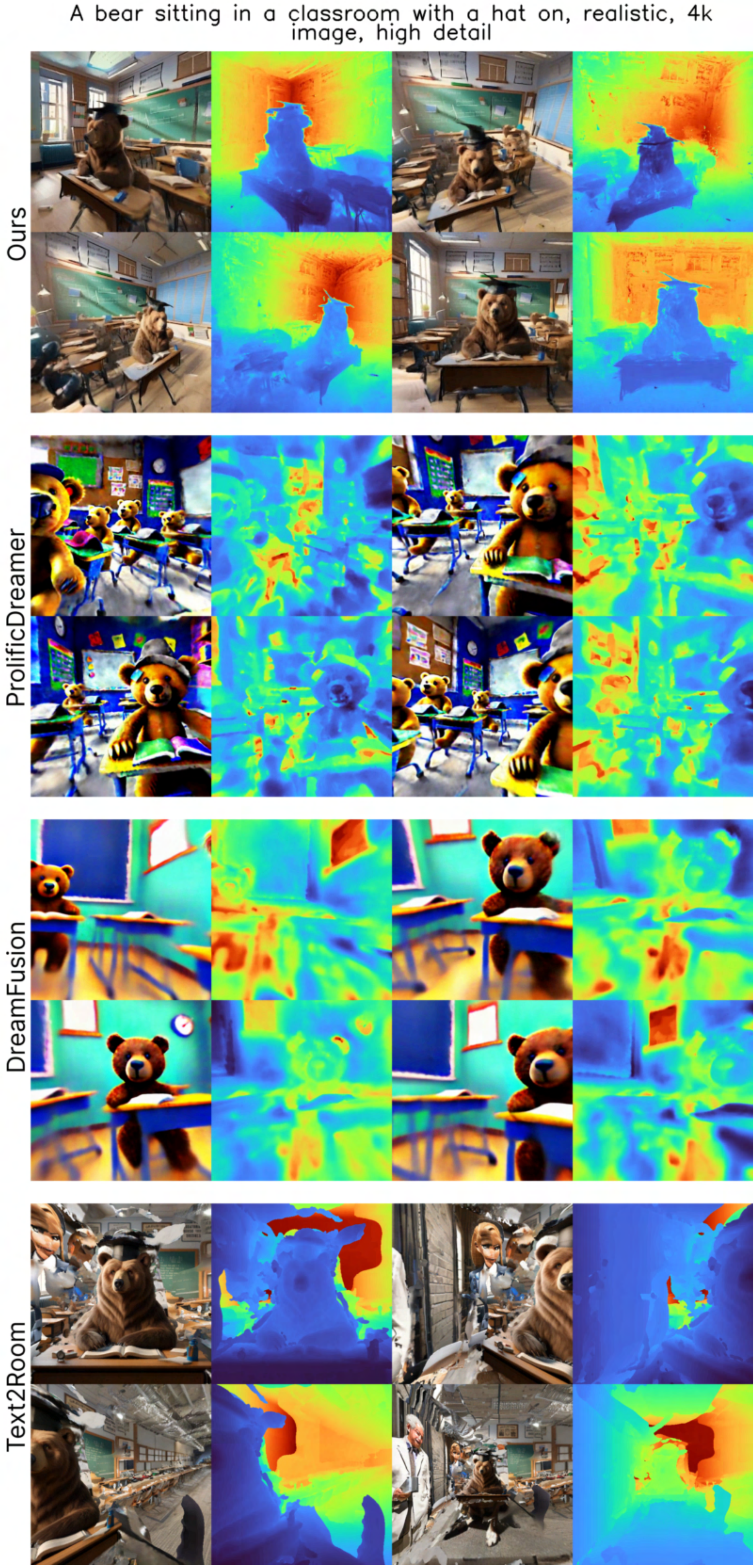
A.2. Additional Qualitative Results

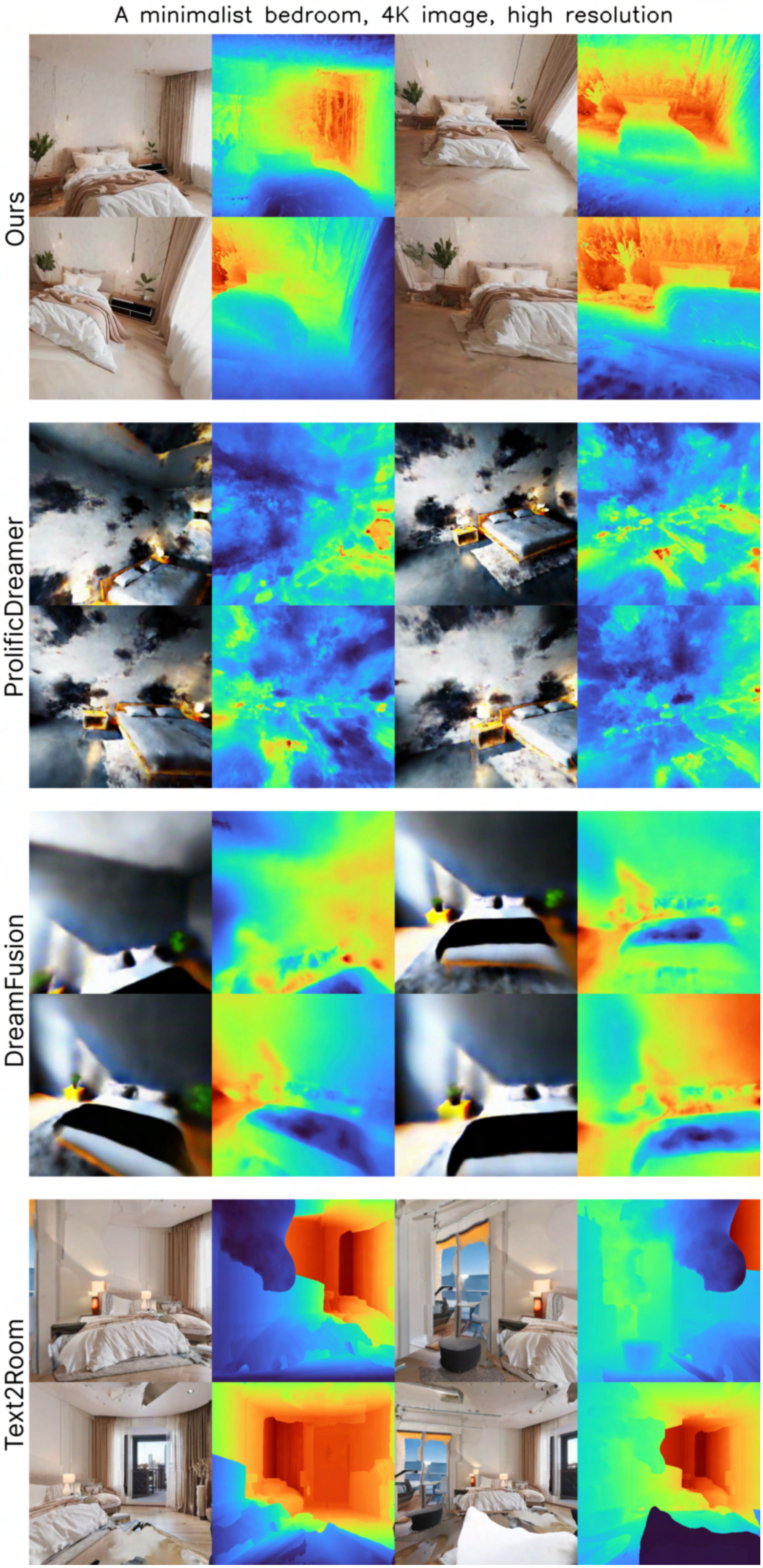
In the following pages, we show qualitative results from our technique as well as all baselines.

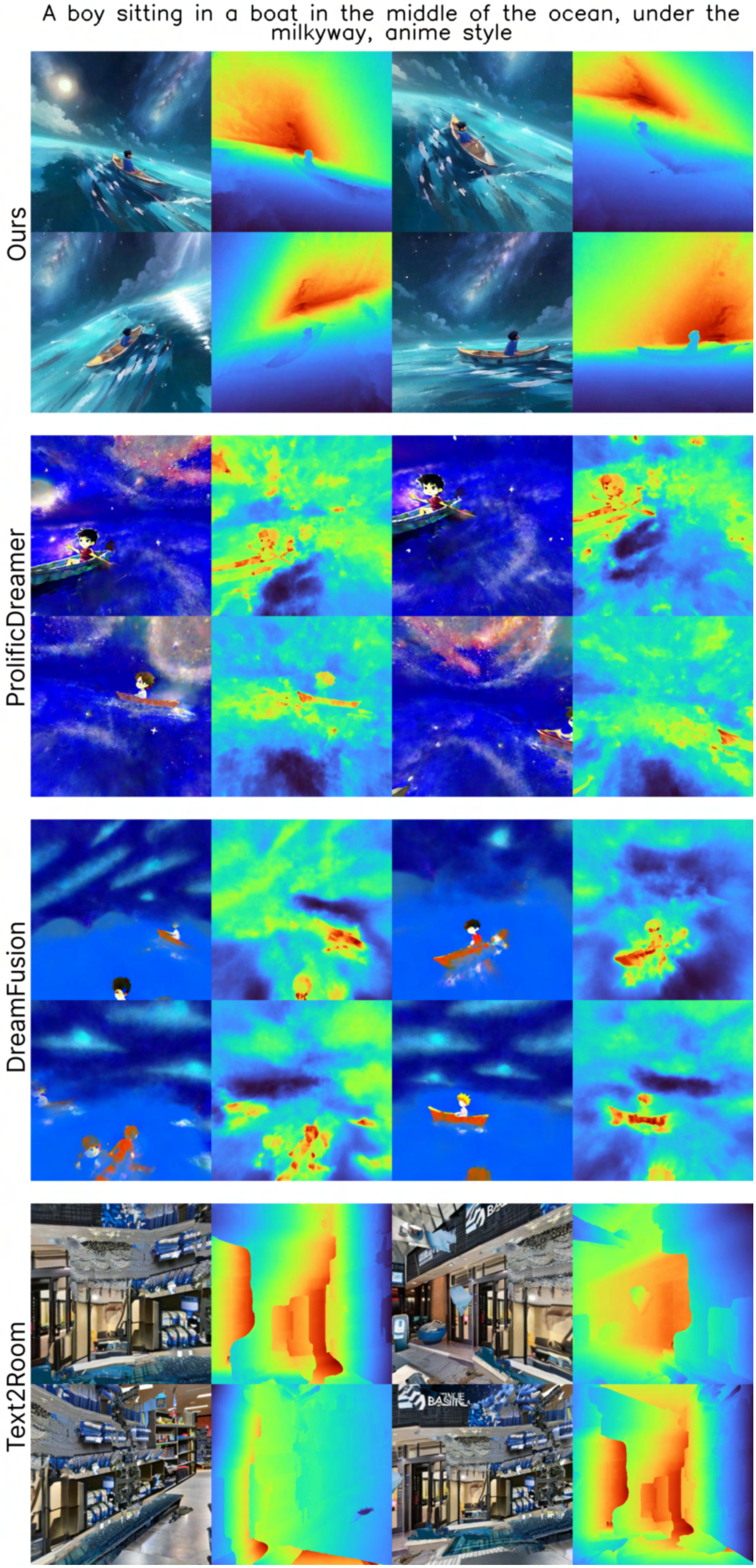

A highly detailed image of the resolute desk in the oval office, 4k image
Ours
ProlificDreamer
DreamFusion
Text2Room


An astronaut in a cave, trending on artstation, 8k image
Ours
ProlificDreamer
DreamFusion
Text2Room

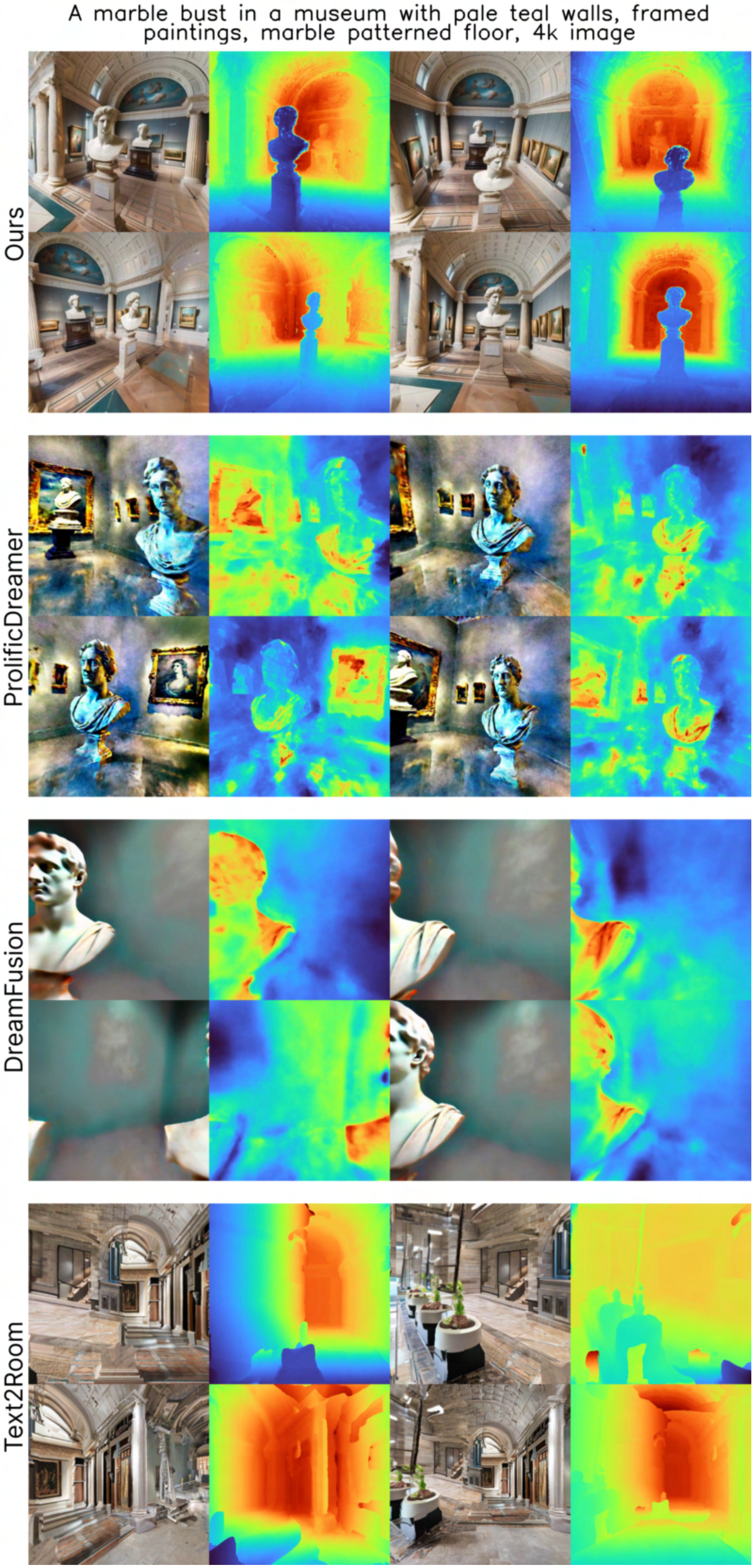

Editorial Style Photo, Coastal Bathroom, Clawfoot Tub, Seashell, Wicker, Mosaic Tile, Blue and White
Ours
ProlificDreamer
DreamFusion
Text2Room

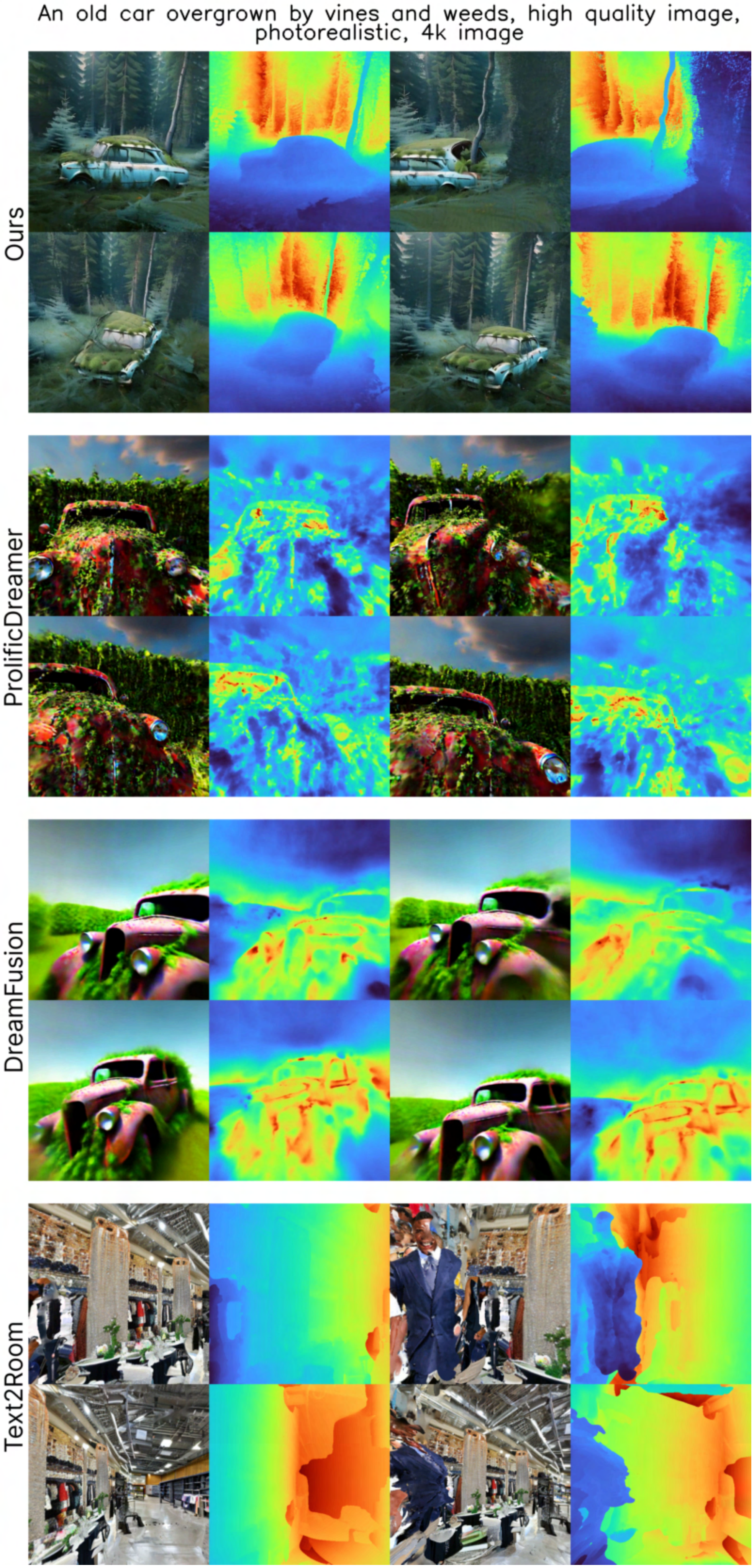

A bear sitting in a classroom with a hat on, realistic, 4k image, high detail
Ours
ProlificDreamer
DreamFusion
Text2Room

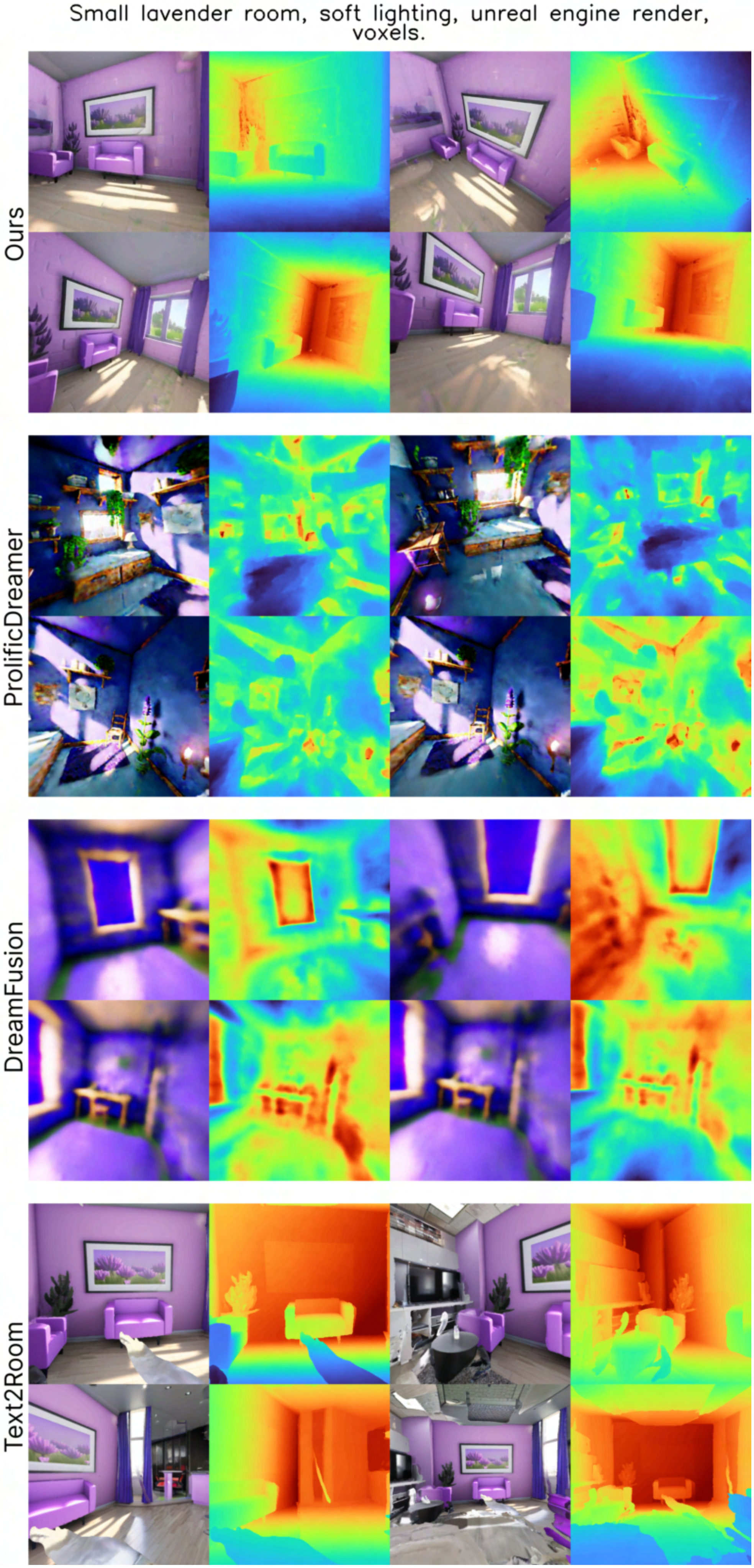

A minimalist bedroom, 4K image, high resolution
Ours
ProlificDreamer
DreamFusion
Text2Room



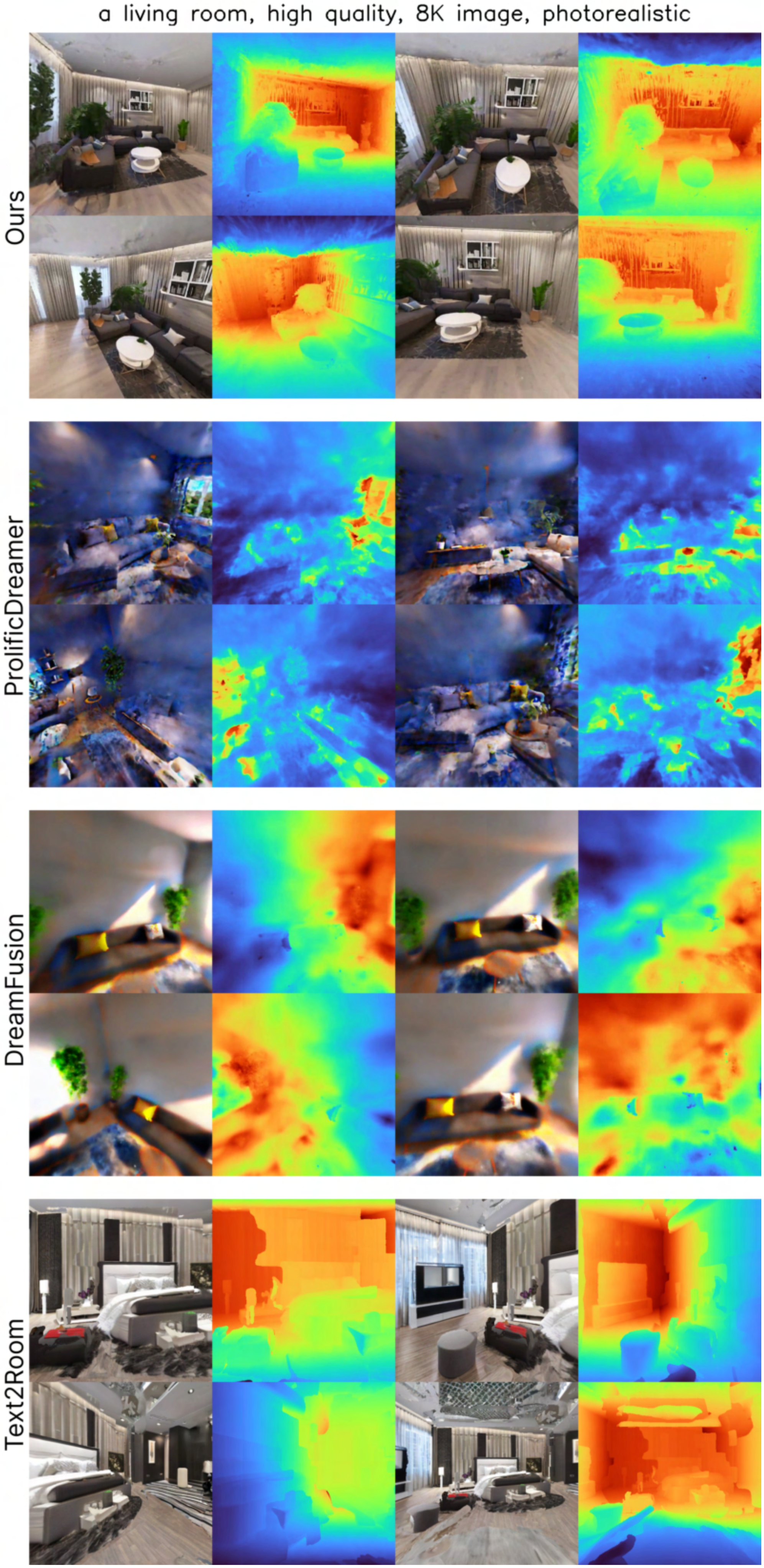
A boy sitting in a boat in the middle of the ocean, under the milkyway, anime style

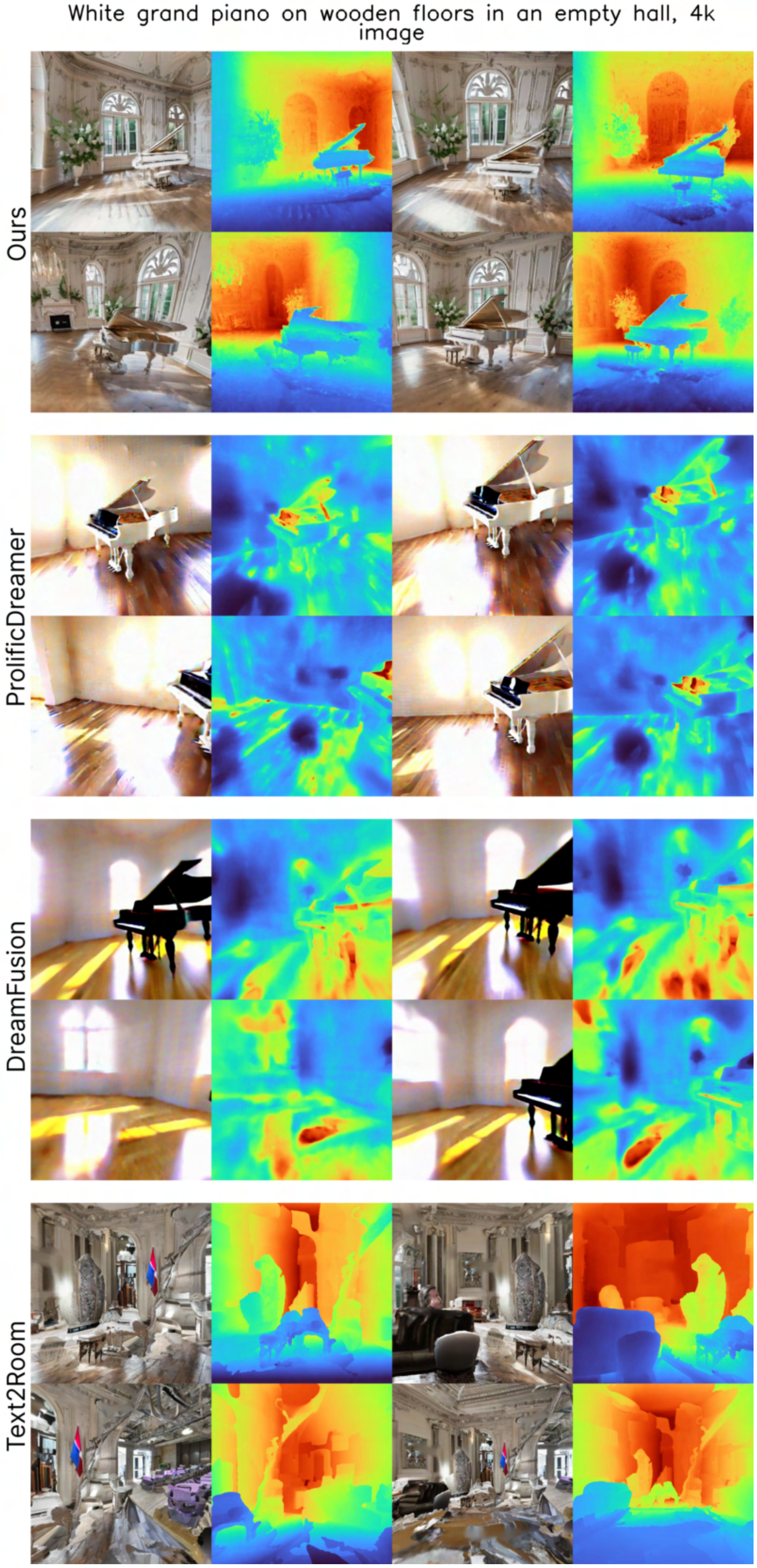

A marble bust in a museum with pale teal walls, framed paintings, marble patterned floor, 4k image
Ours
ProlificDreamer
DreamFusion
Text2Room


An old car overgrown by vines and weeds, high quality image, photorealistic, 4k image
Ours
ProlificDreamer
DreamFusion
Text2Room



Small lavender room, soft lighting, unreal engine render, voxels.


a living room, high quality, 8K image, photorealistic
Ours
ProlificDreamer
DreamFusion
Text2Room


White grand piano on wooden floors in an empty hall, 4k image
Ours
ProlificDreamer
DreamFusion
Text2Room