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 [2]University of Pennsylvania

https://realmdreamer.github.io/

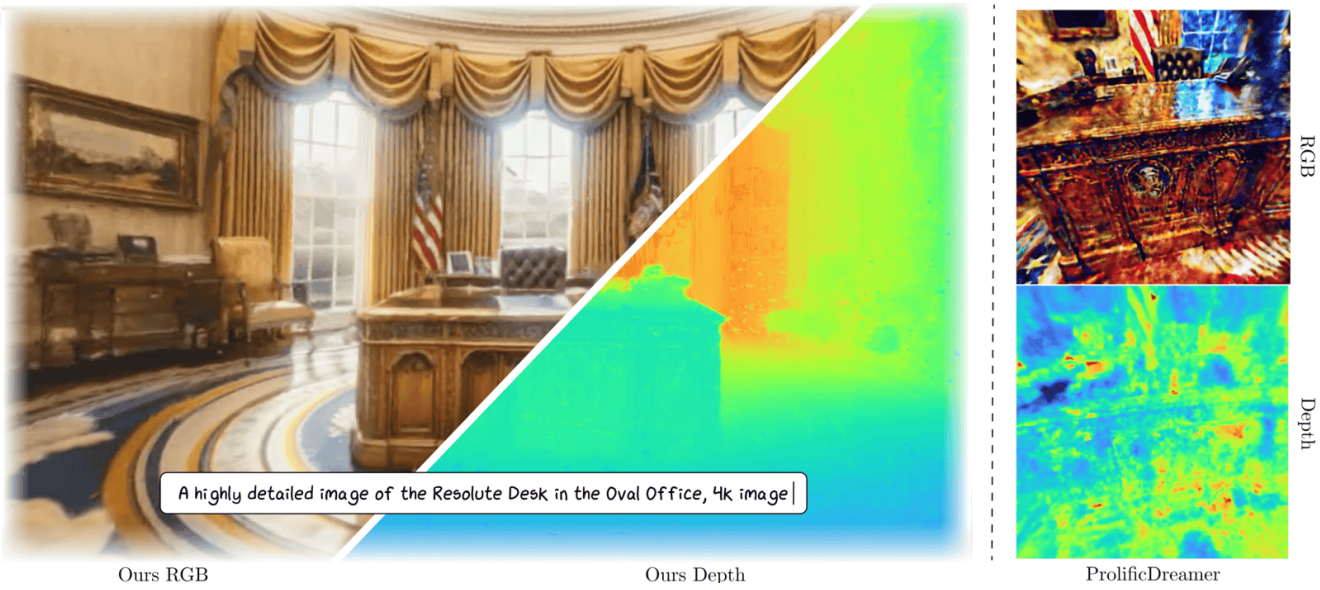


Figure 1. **A scene created by our method on the left compared to baseline ProlificDreamer [64] on the right.** RealmDreamer generates 3D scenes from text prompts (as above), achieving state-of-the-art results with parallax, detailed appearance, and realistic geometry.

Abstract

*We introduce **RealmDreamer**, a technique for generating forward-facing 3D scenes from text descriptions. Our method optimizes a 3D Gaussian Splatting representation to match complex text prompts using pretrained diffusion models. Our key insight is to leverage 2D inpainting diffusion models conditioned on an initial scene estimate to provide low variance supervision for unknown regions during 3D distillation. In conjunction, we imbue high-fidelity geometry with geometric distillation from a depth diffusion model, conditioned on samples from the inpainting model. We find that the initialization of the optimization is crucial, and provide a principled methodology for doing so. Notably, our technique doesn't require video or multi-view data and can synthesize various high-quality 3D scenes in different styles with complex layouts. Further, the generality of our method allows 3D synthesis from a single image. As measured by a comprehensive user study, our method outperforms all existing approaches, preferred by 88-95%.*

1. Introduction

Text-based 3D scene generation has the potential to revolutionize 3D content creation, with broad applications in virtual reality, game development, and even robotic simulation. However, unlike text-based 2D generative models, 3D data is scarce and lacks diversity, which greatly limits the development of generative 3D techniques. Ideally, one can mitigate this by leveraging rich 2D priors for 3D generation instead. Indeed, object-generation techniques such as DreamFusion [44] and ProlificDreamer [64] do just this, by *distilling* 2D diffusion priors into a 3D representation, with the latter even demonstrating early abilities to generate scenes. Unfortunately, such distillation approaches can often have saturated results, poor geometry, and lack detail, which become very apparent in the more challenging setting of scene generation (Fig. 2). This leaves the question: *How to design a distillation technique for high-quality 3D scene generation from pretrained 2D priors?*

A common observation from distillation based object-generation techniques is that greater 3D consistency in 2D

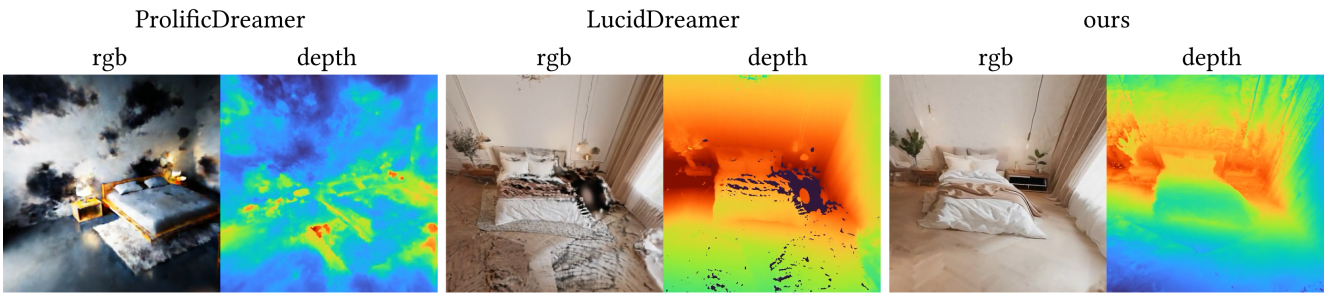


Figure 2. Our method, compared to state-of-the-art ProlificDreamer [64] and concurrent work LucidDreamer [14], shows significant improvements. ProlificDreamer yields unsatisfactory geometry and oversaturated renders. LucidDreamer, receiving the same input as our method and an updated depth model [30], displays degeneracy in disoccluded regions, such as the right side of the bed. In contrast, our approach produces visually appealing 3D scenes with realistic geometry.

diffusion models results in higher-quality distillation, as they provide lower-variance supervision during optimization. As a result, many methods use 2D diffusion models fine-tuned on 3D data [16], such as for novel-view synthesis [21, 37, 45]. Equivalent 3D scene datasets are scarce however, which limits the generalization of such techniques to scenes. Alternatively, ProlificDreamer [64] fine-tuned a diffusion model during distillation to be more 3D consistent, producing more highly-detailed textures than before. In this work, we introduce a technique to achieve these strengths *without* requiring 3D training data or fine-tuning existing 2D diffusion models.

We introduce **RealmDreamer**, a technique for high-fidelity generation of 3D scenes from text prompts (Fig. 1). Our key insight is that we can obtain a 3D scene-aware diffusion model for *free*, by simply re-appropriating 2D inpainting diffusion models. Typically, 2D inpainting models condition on a partial image to fill in the rest. Instead, we demonstrate that such models can also condition on a 3D scene and fill in unknown regions for novel view synthesis through our proposed inpainting distillation process. As a result, we obtain high-quality 3D scenes with considerably improved detail and appearance over prior distillation techniques. Further, we propose a simple initialization strategy that provides a 3D scene to use as conditioning for this distillation and serves as an initial point cloud for the 3DGS model. We evaluate our technique on several quantitative metrics and obtain significantly higher quality results than prior work, as notably shown by a user study where we are preferred over state-of-the-art ProlificDreamer [64] by 95.5%. Concretely, our contributions are the following:

1. An occlusion-aware scene initialization for 3DGS, essential for obtaining high-quality scenes (Sec. 4.1).
2. A framework for distillation from 2D inpainting diffusion models which conditions on the existing scene, providing lower variance supervision (Sec. 4.2).
3. A method for geometry distillation from diffusion-based depth estimators for higher-fidelity geometry. (Sec. 4.3).
4. State-of-the-art results in text-based generation of 3D scenes, as confirmed by several quantitative metrics and a user study (see Fig. 6, Tab. 1, Tab. 2).

2. Related Work

Text-to-3D. The first methods for text-to-3D generation were based on retrieval from large databases of 3D assets [9, 10, 15]. Subsequently, learning-based methods have dominated [1, 11, 36]. However, due to the dearth of diverse paired text and 3D data, many recent methods leverage 2D priors, such as CLIP [27, 53] or text-to-image diffusion models [13, 33, 44, 63, 64, 67]. These distill knowledge from 2D priors into a 3D representation, through variations on Dreamfusion's score distillation sampling (SDS) [44]. However, these techniques have primarily been limited to object synthesis. In contrast, there are iterative techniques that incrementally build 3D scenes [14, 26] or 3D-consistent perpetual views [18], but can struggle with high parallax. Our proposed technique builds on strengths from distillation and iterative techniques to produce large scale 3D scenes with high parallax using pretrained 2D priors.

View Synthesis with Diffusion and 3D inpainting. Motivated by the success of SDS, several techniques generate 3D objects from a single image by leveraging image-guided diffusion models to generate novel views and distill to 3D [17, 70]. When trained on larger datasets [16], with better conditioning architectures, these approaches [37, 38, 54–56] can produce higher quality novel view renders with sharper texture. Some methods also condition denoising directly on renderings from 3D consistent models [8, 21] for view synthesis in a multi-view consistent manner. Unfortunately, most techniques rely on object-level data, limiting their use for text-based scene synthesis. 3D inpainting techniques [41, 42] also leverage image-guided diffusion models to remove small objects in scenes. Other works focus on

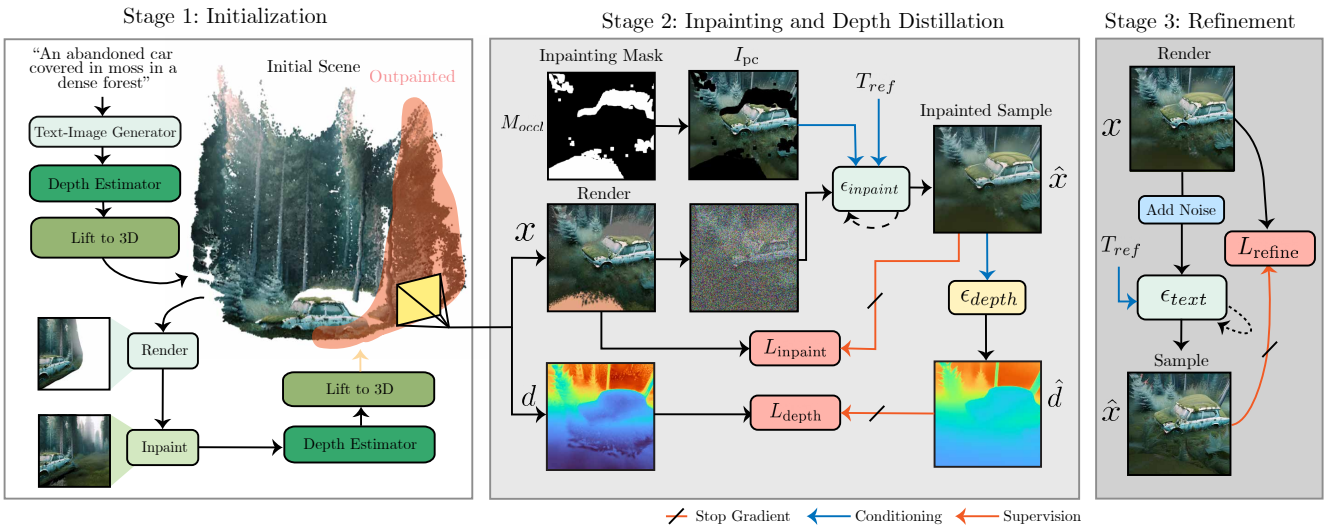


Figure 3. **Overview of our technique.** Our technique first uses a text prompt and an image to build a point cloud (Sec. 4.1), which is then completed during the inpainting stage (Sec. 4.2) with an additional depth diffusion prior (Sec. 4.3), and finally a refinement stage (Sec. 4.4) to improve the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

training custom inpainting models for indoor scenes [32] or objects [28] to generate novel views. In contrast to these, we leverage pre-trained text-guided inpainting priors and focus on generating large missing regions of diverse scenes with our novel inpainting distillation loss.

Concurrent work. In the rapidly evolving text-to-3D field, we focus on the most relevant concurrent works, highlighting our key differences. LucidDreamer [14] and Text2NeRF [68] uses an iterative approach similar to PixelSynth [49] and Text2Room [26] to generate 3D scenes but displays limited parallax. Considering LucidDreamer as the most relevant concurrent baseline, we compare it in the fairest setting possible, by using newer depth estimators [30, 65], and surpass it by 88.5% in our user study. Most recently, in follow-up work, CAT3D [20], utilizes a diffusion model finetuned on multiview datasets to generate multiple views from a single image. In contrast, our entire pipeline does not use multiview images.

3. Preliminaries

3.1. 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [31] 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 $S = \text{Diag}(s)$. 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 [72]) and σ_i. A pixel's color is obtained by α-blending Gaussians sorted by depth:

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

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], which is not applicable for scene generation. To address this challenge, we generate a prototype of our 3D scene using a text prompt, which we then optimize (Sec. 4.1).

3.2. Conditional Diffusion Models

Diffusion models [25, 29, 58–61] are generative models which learn to map noise $x_T \sim \mathcal{N}(0, I)$ to data by iteratively denoising a set of latents x_t corresponding to decreasing noise levels t using non-deterministic DDPM [25] or deterministic DDIM sampling [59], among others [29, 60, 61].

Given t, a diffusion model ϵ_θ is trained to predict the noise ϵ added to the image such that we obtain $\epsilon_\theta(x_t, t)$, which approximates the direction to a higher probability density. Often, the data distribution is conditional on quantities such as text T and images I, so the denoiser takes the form $\epsilon_\theta(x_t, I, T)$. In the conditional case, classifier-free

I_{ref} a) Initialization from Stage 1 b) Inpainted Model c) Finetuned Model

Figure 4. **Progression of 3D Model after each stage.** 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.

guidance is often used to obtain the predicted noise [6, 24]:

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

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. In the case of latent diffusion models like Stable Diffusion [50], denoising happens in a compressed latent space by encoding and decoding images with an encoder $\mathcal{E}$ and decoder D.

Score Distillation Sampling. 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) [44, 63]. 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 [71]:

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

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. In this work, we use a variation of SDS to distill from pretrained-inpainting models (Sec. 4.2)

4. Method

We now describe our technique in detail, which broadly consists of three stages: **initialization** (left of Fig. 3, Sec. 4.1); **inpainting** (middle of Fig. 3, Sec. 4.2) with **depth distillation** (middle of Fig. 3, Sec. 4.3); and **finetuning** (right of Fig. 3, 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 use 2D inpainting models to predict novel views, distilling to 3D to create a complete 3D scene (Sec. 4.2). In this stage, we also incorporate depth distillation for higher-quality geometry (Sec. 4.3). Finally, we refine the model with a sharpness filter on sampled images to obtain high-quality 3D samples (Sec. 4.4). The result from these stages are shown in Fig. 4.

4.1. Initializing a Scene-level 3D Representation

Our technique utilizes 3DGS for text-conditioned optimization, making a good initialization essential. A common strategy in this setting is to initialize with a sphere [34, 44] but the density of a scene is more complex and distributed. Hence, we leverage pretrained 2D priors to synthesize a robust initialization (left of Fig. 3).

Concretely, we first generate a reference image of the scene I_{ref} from the text prompt T_{ref} with a state-of-the-art text-to-image-model. We then employ a monocular depth model [30] $\mathcal{D}$ to lift this image to a pointcloud $\mathcal{P}$ from corresponding camera pose P_{ref}. 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 left and right of P_{ref} to poses P_{aux}. We use an inpainting diffusion model [50] to fill in the unseen regions which are lifted to 3D using $\mathcal{D}$. The union of all generated points thus becomes $\mathcal{P}$.

Determining Incomplete Regions. Given the initial point cloud $\mathcal{P}$, we then precompute the undetermined 3D region, or the *occlusion volume* $\mathcal{O}$, which is the set of voxel centers within the scene's occupancy grid which are occluded by the existing points in $\mathcal{P}$ from P_{ref}. We use $\mathcal{O}$ when computing inpainting masks later and define the initialization of our 3DGS means as

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

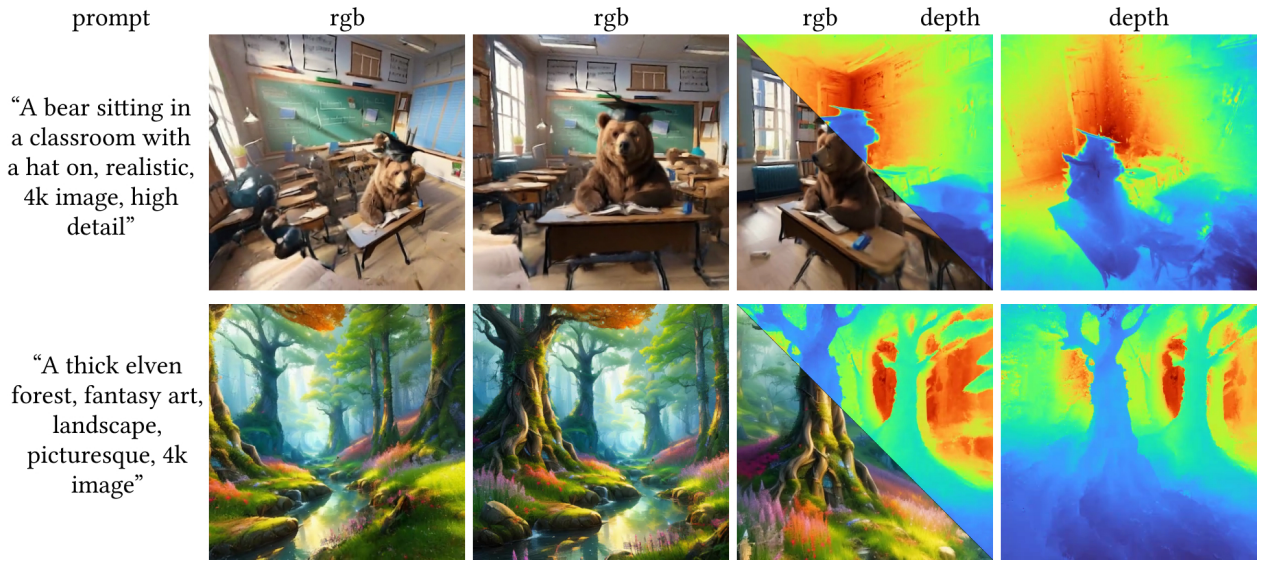


Figure 5. **Qualitative Results.** In the left column, we show the input prompt for our technique. In the next two columns, we show the renderings from our 3D model from different viewpoints. In the fourth column, we show the level of agreement between rendering and geometry by a split view of the rendering and depth. Finally, in the last column, we show the depth map.

More details can be found in the supplementary.

4.2. Inpainting Diffusion for 3D-Conditioned Distillation

Since our initialization is generated from sparse poses, viewing it from novel viewpoints exposes large holes in disoccluded regions (Fig. 4). We resolve this with a novel inpainting distillation technique, that conditions a 2D inpainting diffusion model $\epsilon_{\text{inpaint}}$ [50] on the existing scene to complete missing regions. The model takes as input a noisy rendering x_t of $\{\Theta_i\}_{i=1}^N$, and is conditioned by the text prompt T_{ref}, an occlusion mask M_{occl}, and the point cloud render I_{pc}. Sampling from this model results in novel views $\hat{x}$ which plausibly fill in the holes in the renderings while preserving the structure of the 3D scene (Fig. 3).

Conditioning the inpainting model. To compute the conditioning mask M_{occl} for $\epsilon_{\text{inpaint}}$, we render the point cloud $\mathcal{P}$ and the precomputed occlusion volume $\mathcal{O}$. We 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 (see the supplement for a visualization).

Computing the inpainting loss. Our 2D inpainting diffusion model $\epsilon_{\text{inpaint}}$ [50] operates in latent space, thus additionally parametrized by its encoder $\mathcal{E}$ and decoder D. 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 $\hat{x} = \mathsf{D}(\hat{z})$. We compute the L2 loss between the latents of the render and sample, as well as an L2 and LPIPS perceptual [69] 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

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

with λ weighting the different terms. We discuss the similarity of this loss with SDS in the supplemetary.

Discussion. In contrast to existing iterative methods which utilize inpainting (such as Text2Room and LucidDreamer), our framework does not iteratively construct a scene with inpainting. In practice, sampling from inpainting models often produces artifacts (such as due to out-of-distribution masks), which iterative approaches can amplify when generating from new poses. In contrast, due to scene-conditioned multiview optimization, we obtain cohesive 3D scenes and do not progressively accumulate errors. Moreover, in contrast to DreamFusion and ProlificDreamer, our method utilizes a scene-conditional diffusion model, providing lower variance updates for effective optimization (see row 2 of Fig. 7). This avoids the high-saturation and blurry results that are typically found (Fig. 6).

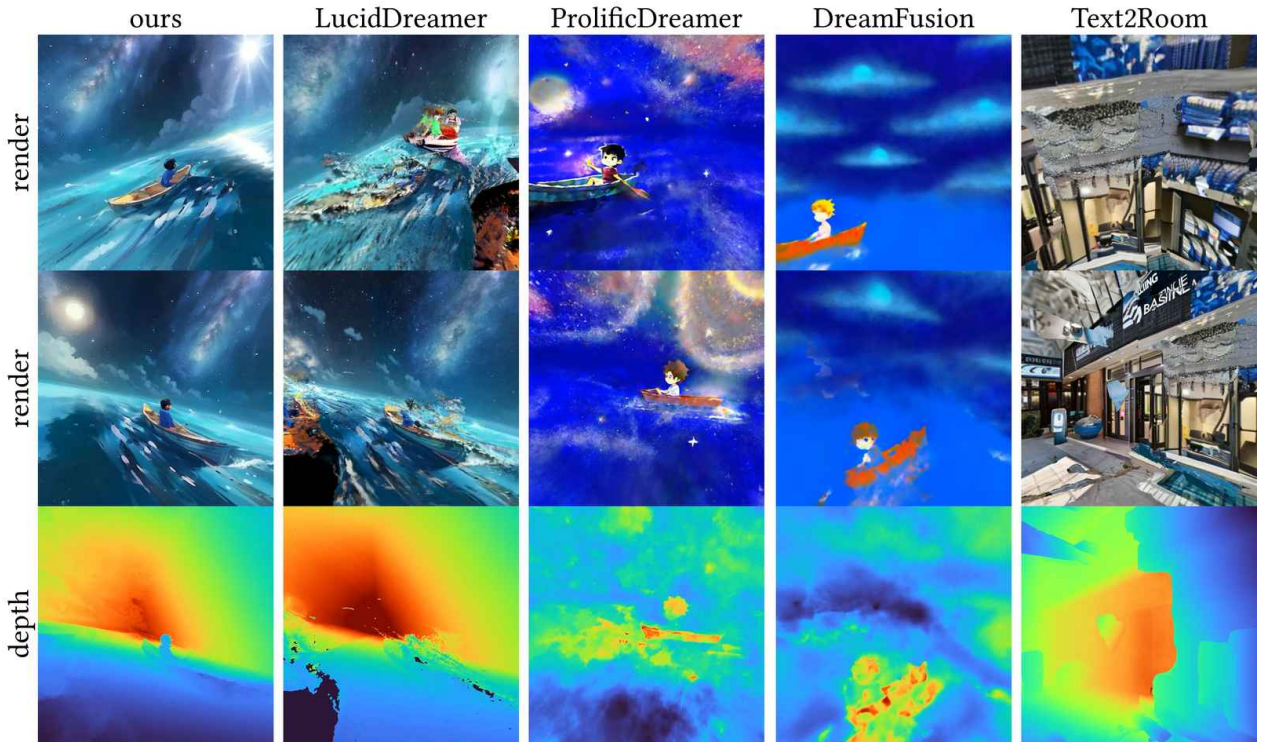


Figure 6. **Qualitative Comparisons.** Our technique shows superior quality in appearance and geometry than all baselines. Please see the supplementary for more comparisons. Prompt: "A boy sitting in a boat in the middle of the ocean, under the milkyway, anime style".

4.3. Depth Diffusion for Geometry Distillation

To improve the quality of generated geometry, we incorporate a pretrained geometric prior to avoid degenerate solutions. Here, we leverage monocular depth diffusion models and propose an additional depth distillation loss (middle of Fig. 3). Crucially, we integrate this with our inpainting distillation by conditioning the depth model ϵ_{depth} on the aforementioned samples $\hat{x}$ from $\epsilon_{\text{inpaint}}$.

Our insight is that these samples $\hat{x}$ act as suitable, in-domain, conditioning for the depth diffusion model throughout optimization, while renders x can be incoherent before convergence. Further, this ensures that predictions from ϵ_{depth} are aligned with $\epsilon_{\text{inpaint}}$ despite not using a RGBD prior. 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 and n is the number of pixels.

4.4. Optimization and Refinement

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 (see Fig. 4). To remedy this, we incorporate an additional lightweight refinement phase. In this phase, we utilize a vanilla text-to-image diffusion model ϵ_{text} personalized for the input image with Dreambooth [17, 39, 47, 51]. 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}. Note that the noise added to the renderings at this stage is smaller to combat the higher variance samples from the lack of image conditioning.

We also propose a novel sharpening procedure: 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 incorporate an opacity loss L_{opacity} per point that encourages a point's opacity to reach either 0 or 1, inspired by the transmittance regularizer used in Plenoxels [19]. The combined

loss for the fine-tuning stage is:

$$L_{\text{refine}} = L_{\text{text}} + \lambda_{\text{opacity}} L_{\text{opacity}}, \tag{8}$$

where λ_{opacity} controls the effect of the opacity loss.

4.5. Implementation Details

Point Cloud Initialization. We implement this stage (Sec. 4.1) in Pytorch3D [48], with Stable Diffusion [50] for outpainting. To lift the generated images to 3D, we use Marigold [30], a monocular depth estimation model. Since it predicts relative depth, we align its predictions with the metric depth predicted by DepthAnything [65].

Inpainting and Refinement Stage. Our inpainting (Sec. 4.2) and refinement stages (Sec. 4.4) are implemented in NeRFStudio [62] using the official implementation of Gaussian Splatting [31]. We use Stable Diffusion 2.0 as ϵ_{text} and its inpainting variant as $\epsilon_{\text{inpaint}}$, building on threestudio [22] to define our diffusion-guided losses. Further, we use Marigold [30] 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, and sample the timestep t from $\mathcal{U}(0.1, 0.95)$. 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 refinement stage, sampling noise from $\mathcal{U}(0.1, 0.3)$.

Timing. The first stage, currently unoptimized, takes 2.5 hours. The inpainting stage, trained for 15,000 iterations, runs for 8 hours on a 24GB Nvidia A10 GPU. The refinement stage, at 3,000 iterations, completes in 2.5 hours on the same GPU.

5. Results

We evaluate our technique on a custom dataset of 20 prompts, and associated camera poses P_i, selected to showcase parallax and disocclusion. We built this dataset by creating a set of 20 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-to-3D techniques [44, 64] typically operate with spherical camera priors. Please refer to the supplemental video results to see the generated scenes.

5.1. Qualitative Results

We show some qualitative results in Fig. 5 with additional results in the supplementary, demonstrating effective 3D scene synthesis across various settings (indoor, outdoor) and image styles (realistic, fantasy, illustration). We would like to highlight the rendering quality and the consistency of rendering and geometry, underscoring our method's use of inpainting and depth priors.

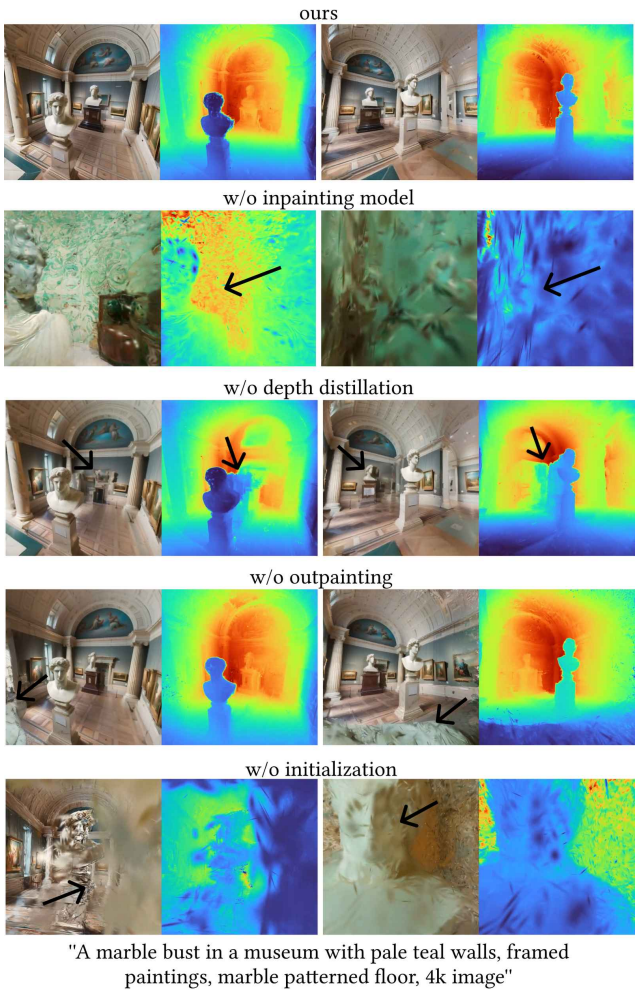


Figure 7. **Ablation Results.** We show the qualitative results of our model and its ablations. Arrows indicate failures in the ablated models. Please see Sec. 5.5 for a detailed discussion of the ablated components and their respective importance.

5.2. Comparisons

We compare our technique with state-of-the-art for text-to-3D that use either distillation or iterative approaches: DreamFusion [44], ProlificDreamer [64], Text2Room [26], and concurrent work LucidDreamer [14] (Fig. 6). Both ProlificDreamer and DreamFusion generate oversaturated scenes with incorrect geometry and scene structure. On the other hand, Text2Room fails to construct non-room scenes, as it deviates from the input prompt during generation. Similarly, LucidDreamer's [14] scenes lack cohesion, with noisy results in occluded regions. Note that LucidDreamer and Text2Room take an image as input; we gave these baselines the same input image as to ours and updated their depth models.

5.3. User Study

To validate the quality of our generated 3D scenes, we conduct a user study (Tab. 1), similar to prior work [12, 35, 64]. We conducted a study on Amazon Mechanical Turk and recruited 20 participants with a 'master' qualification to compare each baseline, while following several guidelines outlined in [7]. The details can be found in the supplementary. Participants overwhelmingly prefer results from our technique over baselines.

Table 1. **Results of user study.** We show the percentage of comparisons where our technique was preferred over baselines: PD [64], DF [44], T2R [26], and LD [14].

Ours vs. PD	Ours vs. DF	Ours vs. T2R	Ours vs. LD
95.5%	**94.5%**	**88%**	**88.5%**

Table 2. **CLIP alignment scores and additional metrics** for scene renderings of our method and the baselines. CLIP scores are scaled by 100. Higher is better for all metrics.

Method	CLIP	Depth Pearson	IS
Ours	**31.69**	**0.89**	**6.99**
Text2Room [26]	28.11	0.77	5.10
DreamFusion [44]	29.48	0.09	6.80
ProlificDreamer [64]	29.39	0.16	6.89
LucidDreamer [14]	29.97	0.80	5.73

5.4. Quantitative Metrics

We also provide a quantitative comparisons with all baselines based on alignment to the text prompt using CLIP [46], Inception Score [52] on renderings, and the quality of geometry with the pearson correlation between rendered depth and the predicted depth by DepthAnythingV2 [66]. We note that due to the lack of ground truth data, standard reconstruction metrics such as PSNR or LPIPS [69] do not apply. We compute these scores for renderings from the same trajectory and the corresponding prompt for all scenes. As Text2Room's results degrade significantly away from the initial pose, we compare with a render from the initial pose for CLIP. As shown in Tab. 2, our method shows significantly better performance across all metrics.

5.5. Ablations

We verify the proposed contributions of our method by ablating the key components in Fig. 7 with the specified prompt (Tab. 3). In the first row, we show our method. In the second row, we show the importance of the low variance samples from the inpainting diffusion model (Sec. 4.2). Distillation with a vanilla text-to-image model as in the final stage, results in high-variance samples causing the 3DGS representation to diverge. In the third row, we remove L_{depth}; this results in incorrect geometry and incoherent renderings. Note in particular the discrepancy in the background when viewing from left versus right. In the fourth row, we initialize our method using only the reference image I_{ref} without outpainting at the neighbouring poses P_{aux}. This results in poor results in the corresponding regions, as they lack a good initialization. Finally, in the last row, we show our result without using the μ initialization from Eq. (4), which results in divergence.

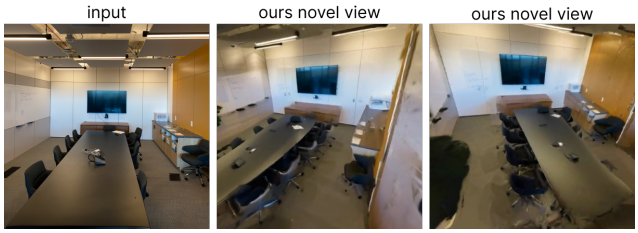


Figure 8. **Result for single-image to 3D.** Using a provided image and a prompt obtained via an image captioning model, our technique can generate a 3D scene and fill in occluded regions.

Table 3. **Ablation Study Results** showing the impact of different components on Depth Pearson correlation and CLIP score. CLIP scores are scaled by 100. Higher is better for both metrics.

Ablation	Depth	CLIP
No Depth Loss	0.86	31.55
No Initialization	0.42	20.31
No Inpainting	0.50	21.14
No Outpainting	0.79	31.00
Ours	**0.90**	**33.10**

5.6. Application: Single image to 3D

Our technique extends to creating 3D scenes from a single image, as shown in Fig. 8, by using a user's image as I_{ref} and a text-prompt T_{ref} obtained using an image-captioning model. Our pipeline can effectively fill in occluded areas and generate realistic geometry for unseen regions.

6. Conclusion

We have proposed **RealmDreamer**, a method for generation of forward-facing 3DGS scenes leveraging inpainting and depth diffusion. Our key insight was to leverage the lower variance of image conditioned (inpainting) diffusion models for synthesis of 3D scenes, providing much higher quality results than existing baselines as measured by a comprehensive user study. Still, limitations remain; our method takes several hours, and produces blurry results for complex scenes with significant disocclusion. Future work may explore efficient diffusion models for faster training, and conditioning for 360-degree generations.

7. 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 (ICML)*, pages 214–223, 2017. 2

[2] Mohammadreza Armandpour, Huangjie Zheng, Ali Sadeghian, Amir Sadeghian, and Mingyuan Zhou. Re-imagine the negative prompt algorithm: Transform 2d diffusion into 3d, alleviate janus problem and beyond. *arXiv preprint arXiv:2304.04968*, 2023. 16

[3] 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. 13

[4] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, and Yufei et al. Guo. Improving image generation with better captions. page 8, 2023. 14

[5] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023. 13

[6] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18392–18402, 2023. 4

[7] 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

[8] 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

[9] Angel Chang, Manolis Savva, and Christopher D Manning. Learning spatial knowledge for text to 3d scene generation. pages 2028–2038, 2014. 2

[10] 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

[11] 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 *Asian Conference on Computer Vision (ACCV)*, pages 100–116. Springer, 2019. 2

[12] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *International Conference on Computer Vision (ICCV)*, 2023. 8

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

[14] 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, 3, 7, 8, 13, 15, 16

[15] Bob Coyne and Richard Sproat. Wordseye: An automatic text-to-scene conversion system. pages 487–496, 2001. 2

[16] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13142–13153, 2022. 2

[17] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20637–20647, 2023. 2, 6

[18] Rafail Fridman, Amit Abecasis, Yoni Kasten, and Tali Dekel. Scenescape: Text-driven consistent scene generation. *arXiv preprint arXiv:2302.01133*, 2023. 2

[19] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5501–5510, 2022. 6

[20] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul P. Srinivasan, Jonathan T. Barron, and Ben Poole. Cat3d: Create anything in 3d with multi-view diffusion models. *arXiv*, 2024. 3

[21] 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 (ICML)*, pages 11808–11826, 2023. 2

[22] 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. 7, 13

[23] 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. 13

[24] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 4

[25] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NIPS*, 33:6840–6851, 2020. 3

[26] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. In *International Conference on Computer Vision (ICCV)*, pages 7909–7920, 2023. 2, 3, 7, 8, 13, 14, 15, 16

[27] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 867–876, 2022. 2

[28] 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*, pages 1–12, 2023. 3

[29] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *NIPS*, 35:26565–26577, 2022. 3

[30] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. *arXiv preprint arXiv:2312.02145*, 2023. 2, 3, 4, 7, 13, 14

[31] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4), 2023. 3, 7

[32] Jiabao Lei, Jiapeng Tang, and Kui Jia. Rgbd2: Generative scene synthesis via incremental view inpainting using rgbd diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8422–8434, 2023. 3

[33] 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. *arXiv preprint arXiv:2311.11284*, 2023. 2, 13, 14

[34] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 4

[35] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 300–309, 2023. 8

[36] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *NIPS*, 36, 2024. 2

[37] 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 *International Conference on Computer Vision (ICCV)*, pages 9298–9309, 2023. 2

[38] 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

[39] Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Realfusion: 360° reconstruction of any object from a single image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8446–8455, 2023. 6

[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 *European Conference on Computer Vision (ECCV)*, 2020. 3

[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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20669–20679, 2023. 2

[43] 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. 14

[44] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 1, 2, 4, 7, 8, 12, 13, 15, 16

[45] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. In *International Conference on Learning Representations (ICLR)*, 2024. 2

[46] 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 (ICML)*, pages 8748–8763, 2021. 8

[47] 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

[48] 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. 7, 14

[49] Chris Rockwell, David F. Fouhey, and Justin Johnson. Pixelsynth: Generating a 3d-consistent experience from a single image. In *International Conference on Computer Vision (ICCV)*, 2021. 3

[50] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 4, 5, 7, 12, 15

[51] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22500–22510, 2023. 6, 14

[52] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2234–2242, 2016. 8

[53] Aditya Sanghi, Hang Chu, Joseph G Lambourne, Ye Wang, Chin-Yi Cheng, Marco Fumero, and Kamal Rahimi Malek-

shan. Clip-forge: Towards zero-shot text-to-shape generation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18603–18613, 2022. 2

[54] Kyle Sargent, Zizhang Li, Tanmay Shah, Charles Herrmann, Hong-Xing Yu, Yunzhi Zhang, Eric Ryan Chan, Dmitry Lagun, Li Fei-Fei, Deqing Sun, et al. Zeronvs: Zero-shot 360-degree view synthesis from a single real image. *arXiv preprint arXiv:2310.17994*, 2023. 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.

[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 *Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 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 (ICML)*, pages 2256–2265, 2015. 3

[59] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3, 5, 6, 13

[60] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *NIPS*, 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 *SIGGRAPH*, 2023. 7

[63] 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, 4

[64] 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, 7, 8, 12, 13, 15

[65] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3, 7, 13, 14

[66] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024. 8

[67] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 2

[68] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Text2nerf: Text-driven 3d scene generation with neural radiance fields. *IEEE TVCG*, 2024. 3

[69] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. 5, 8

[70] Zhizhuo Zhou and Shubham Tulsiani. Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[71] Junzhe Zhu, Peiye Zhuang, and Sanmi Koyejo. Hifa: High-fidelity text-to-3d generation with advanced diffusion guidance. In *International Conference on Learning Representations (ICLR)*, 2023. 4

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

Supplemental Material:

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

Contents

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 [50]. As these models are often trained on vast collections of internet data, they can reflect negative 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.

A. Why is the occlusion volume important?

Our key contribution is the inpainting distillation loss that provides lower variance and high-quality supervision for text-to-3D, compared to regular text-to-image model based distillation, as shown in the ablations. Given that we use 2D inpainting models for 3D inpainting via this distillation, we must ask: *how to compute the 3D region that needs to be inpainted?*

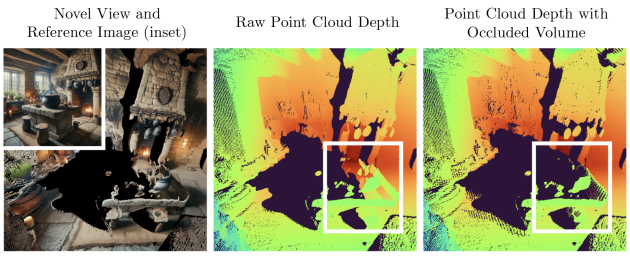


Figure 9. **Creating the Inpainting Mask.** 3D inpainting requires filling in a 3D volume, which is not always equivalent to missing regions in 2D point cloud renders. By computing an *occlusion volume*, we avoid situations where the floor is visible through the table (middle) but instead could be occluded. The right depth map accounts for the ambiguity of the volume in the table.

We proposed a simple technique to do so by computing the **occluded volume** (described in Appendix D.2), which is the 3D region occluded by objects in the reference image I_{ref}. Note that regardless of what objects may be present in this occluded region, rendering from P_{ref} would yield I_{ref}, as elements in the image would occlude the objects. The 2D inpainting masks we obtain then must hence, reflect this unknown 3D region, as that is the part of the scene left to complete.

Instead of computing the occlusion volume, another alternative is using the holes from point cloud renderings as the inpainting mask. Indeed, these holes also represent unknown 3D regions. However, the 3D region indicated by such 2D masks is a **subset** of the occluded 3D region and hence, incomplete. This is shown in Fig. 9, which shows the masked point cloud depth, where the masked region represents the inpainting mask. When using the holes in the point cloud as an inpainting mask, one can observe that the back side of the kitchen table is visible. In reality, a solid kitchen table would never expose this face. In contrast, using the occlusion volume, we can correctly determine the entire 3D region missing in I_{ref}, which can be visually verified by comparing images. Specifically, the latter takes into account self-occlusion, providing a more accurate estimate and allowing details to fill into the occluded region.

In practice, such self-occlusions do not appear for all prompts yet is important to maintain the correctness of the 3D inpainting formulation. If there are many renderings such as in Fig. 9, the quality of inpainted samples would be incorrect and lead to a noisier distillation process.

B. Discussion on Baselines

We are among the first to tackle text-to-3D scenes and showcase a high-level of parallax. As a result, there are limited open-source 3D scene generation techniques to compare with. We choose to compare with techniques that either use distillation - a key part of our pipeline and iterative approaches - which shares similarity with the initialization technique we use.

B.1. Comparison with Dreamfusion and ProlificDreamer

By comparing with prior SOTA distillation techniques [44, 64], we demonstrate that these approaches are suboptimal

for scene generation, which has not been demonstrated before. Arguably, distillation techniques should be able to build any 3D scene since the 2D priors used are general, assuming object-centric regularizers and prompts are absent. However, we empirically find that simple distillation from text-to-image models is insufficient for wide baselines. This comparison highlights the importance of our inpainting distillation, which conditions on a partial 3D scene, enabling high level of parallax not seen in prior work.

We also note that ProlificDreamer [64] first showcased scene-level results using distillation, indicating that distillation is not purely for object-generation. Still, it presented a limited set of scenes and did not show high parallax as it focused on rotating a camera through a scene. Our baseline comparison attempts to test the performance of this approach on wide camera trajectories and finds that it often produces hazy results.

B.2. Relation between SDS and our distillation

Our distillation is similar to the score distillation loss (SDS) used in Dreamfusion [44]. However, unlike prior work, we do not use just text conditioning, but also renders from the point cloud. As a result, the classifier-free guidance weight we is much lower - at 7.5, avoiding over-saturated results typically found with SDS. Further, unlike SDS which denoises noisy renders in one step, we take multiple steps with DDIM sampling. Further, we also use a loss on the denoised latent and decoded image, to produce high quality supervision.

B.3. Comparison with LucidDreamer and Text2Room

Our initialization step is a key part of the pipeline and reminiscent of prior and concurrent iterative techniques which incrementally grow a scene. Yet, such techniques have yet to showcase high quality over high-parallax camera trajectories, which we demonstrate. We find that incremental generation of 3D scenes can lead to noise accumulation, due to errors in monocular depth and alignment of geometry. Hence, unlike concurrent work LucidDreamer [14], we do not limit our scene generation to our initialization step. We rely on our inpainting distillation loss to produce highly cohesive 3D scenes, by distilling across multiple views, rather than building a scene incrementally. We also note that while LucidDreamer does use 3DGS optimization, it is a more conventional reconstruction-based optimization, with no inpainting or geometry priors incorporated. As a result, its optimization stage is very different from our inpainting distillation, which provides rich priors for appearance and geometry at every iteration.

Further, we also avoid many limitations of prior work such as Text2Room, which often produces scenes with low-prompt alignment, especially those in outdoor scenes. We attribute this to the technique deleting regions of the mesh before inpainting, as the original technique prescribes. When such deletions accumulate over time, they are prone to erasing parts of the original scene defined in I_{ref} entirely. In contrast to this, our technique maintains a simple initialization strategy and relies on a high-quality inpainting distillation process to fill-in mission regions, without sacrificing the quality of the initialized regions.

B.4. Implementation

Text2Room [26] and LucidDreamer [14] We use the official implementation of Text2Room and LucidDreamer on Github. To ensure a fair comparison, we estimate depth using Marigold [30] and DepthAnything [65] as in our technique, replacing the original IronDepth [3] and ZoeDepth [5] respectively. The rest of the pipeline is kept the same.

ProlificDreamer [64] and Dreamfusion [44] We use the implementation of these baselines provided in threestudio [22] 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.

C. Additional ablations and discussion

C.1. Use of DDIM Inversion.

During the inpainting and refinement 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 [23, 33]. 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 refinement, we use 100 steps.

C.2. Use of sharpening filter.

In Fig. 10, we also see that applying a sharpening filter to the sampled images results in slightly more detail. We attribute this to the blurry nature of some samples of the diffusion model.

D. Additional Implementation Details

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

D.1. Point Cloud Generation

Image Generation. We generate our reference image I_{ref} using a variety of state-of-the-art text-image generation

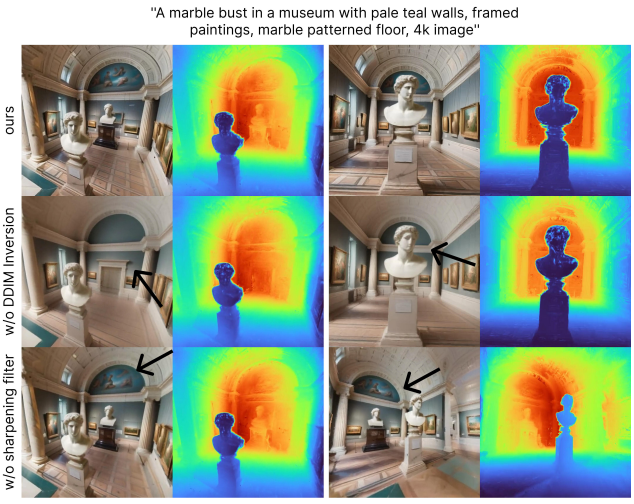


Figure 10. We ablate the importance of DDIM Inversion and applying a sharpening filter. As in [33], we find that DDIM inversion allows more details to be synthesized by our method. Additionally, we find that detail slighly increases when applying a sharpening filter to the sampled images.

models, choosing between Stable Diffusion XL [43], Adobe Firefly, and DALLE-3 [4].

Depth Estimation. As mentioned earlier, we use Marigold [30] as our depth estimation model, with absolute depth obtained using DepthAnything [65]. 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 [26] and align the predicted depth with the ground truth depth rendered via Pytorch3D [48] 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"*.

D.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 $\mathcal{O}$.

D.3. Optimization

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 refinement 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 fine-tuning the output from stage 2, we use Dreambooth [51] 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.

Opacity Loss. We compute the opacity loss as the binary cross entropy of each splat's opacity σ_i with itself. This encourages the opacity to reach either 0 or 1.

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 refinement 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. During the inpainting distillation, we also dilate M_{occl} to improve cohesion at mask boundaries. Further, we find it essential to mask the latent-space L2 loss, to prevent unwanted gradients outside the masked region.

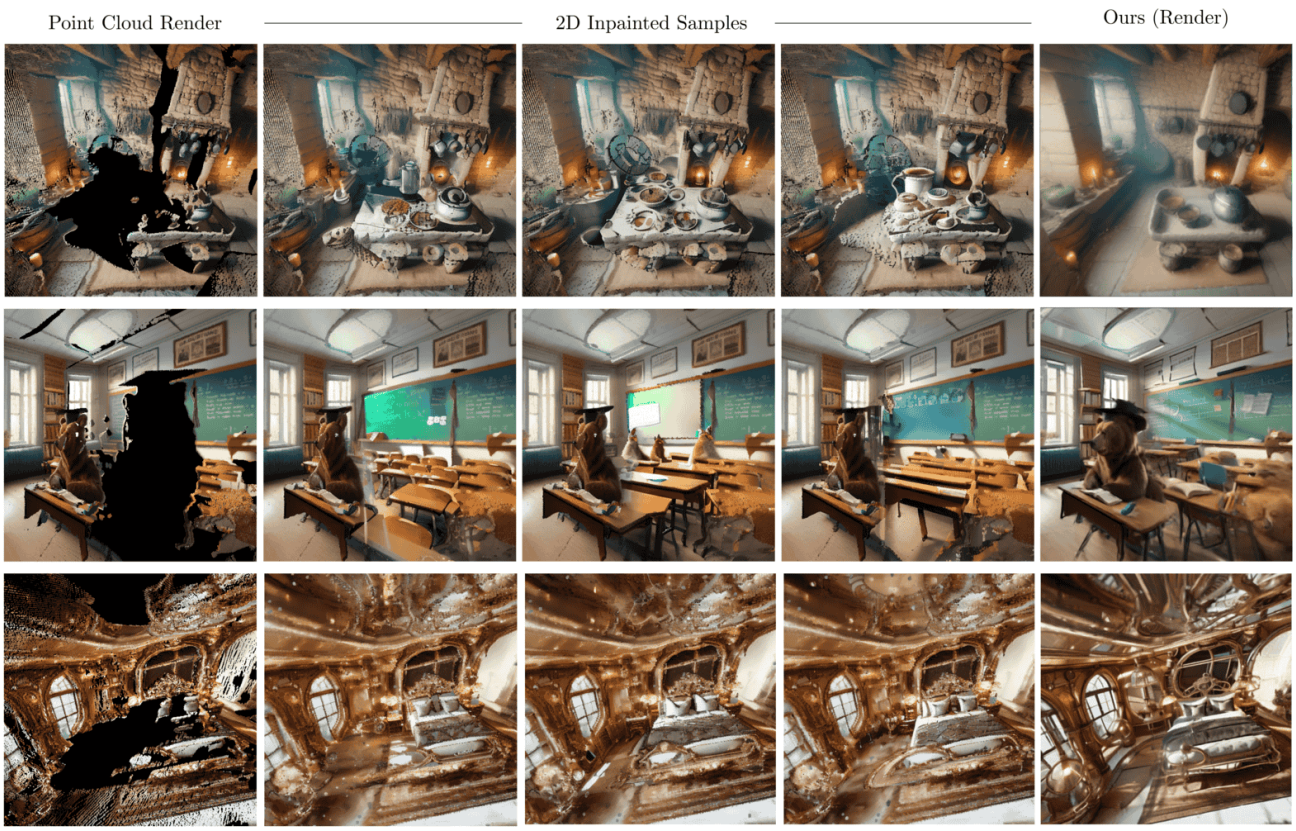


Figure 11. **Comparison of sampling from 2D inpainting models and our optimized model.** Left: Renders from the point cloud generated in stage 1. Middle (cols 2-4): Inpainted Samples of the previous render using an occlusion-based inpainting mask and Stable Diffusion [50]. Right: A render from our final 3DGS model for the corresponding scene. We find that our distillation techniques produce results with high cohesion while avoiding many artifacts from ancestral sampling of 2D inpainting models.

E. User Study

For comparison with ProlificDreamer [64], DreamFusion [44], and LucidDreamer [14], 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 [26], 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 user's preferred triplet was logged along with their brief explanation. The images we showed looked slightly left and right of the reference pose P_{ref}.

E.1. Common themes of the user study.

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 [64] 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 H. In contrast, participants described 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 [44] 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 partici-

pant 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 [26] 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.

LucidDreamer [14]'s scenes lack cohesion and can be distorted. Multiple participants pointed out that LucidDreamer's scenes degrade in quality when moving away from the initial pose. One participant wrote "The image on the left loses cohesion when rotated." referring to LucidDreamer and in contrast another wrote "There is less visual distortion when the camera is moved around the room." about RealmDreamer. Some participants also noted that objects produced by our technique were more *solid*, with one participant noting "The shapes are solid on the right and hold their form.". These comments underscore the limitations of purely iterative approaches.

F. Additional Discussion

F.1. Impact of Distillation

In Fig. 11, we show the importance of our distillation process for filling in occluded regions and the challenge in doing so. Column 1 shows renders following stage 1, which contains large holes, giving objects a *thin* look (such as the bear in row 2 or the table in row 1). By computing an occlusion volume and obtaining inpainting masks, we can inpaint these renders to obtain several inpainted samples (columns 2-4). However, these samples can contain several artifacts. For instance, in row 1 of Fig. 11, the surface of the table is quite cluttered in individual samples. This is likely due to the challenge of inpainting images with complex masks that are out of distribution. These images also show the challenge in building cohesive scenes with single view inpainting. For instance the blackboard in row 2 has multiple shades of green in the 2D samples. Despite these challenges, our final render for the scene, in column 5 of Fig. 11 is clean and free of stray artifacts such as bright colours or ambiguous objects. We attribute this difference to our distillation process.

As mentioned earlier, since we optimize over multiple views, we are less susceptible to artifacts present in individual samples and can produce 3D inpaintings that satisfy multiple views. Prior work, such as Text2Room [26] instead relies primarily on dilating masks and deleting regions of generated scenes to simplify the inpainting process. Our inpainting distillation process does not require any aggressive modification to the scene but can produce high-quality results. We highly encourage the viewer to view the video renderings to appreciate the extent of occluded regions that our distillation technique generates.

Figure 12. **Janus Problem due to multi-view optimization.** Since we optimize over multiple-views, sometimes the final model can show the same object multiple times to satisfy all views, such as the pair of glasses above the octopus. Prompt: "A blue octopus wearing glasses on a couch in the living room, watercolor style"

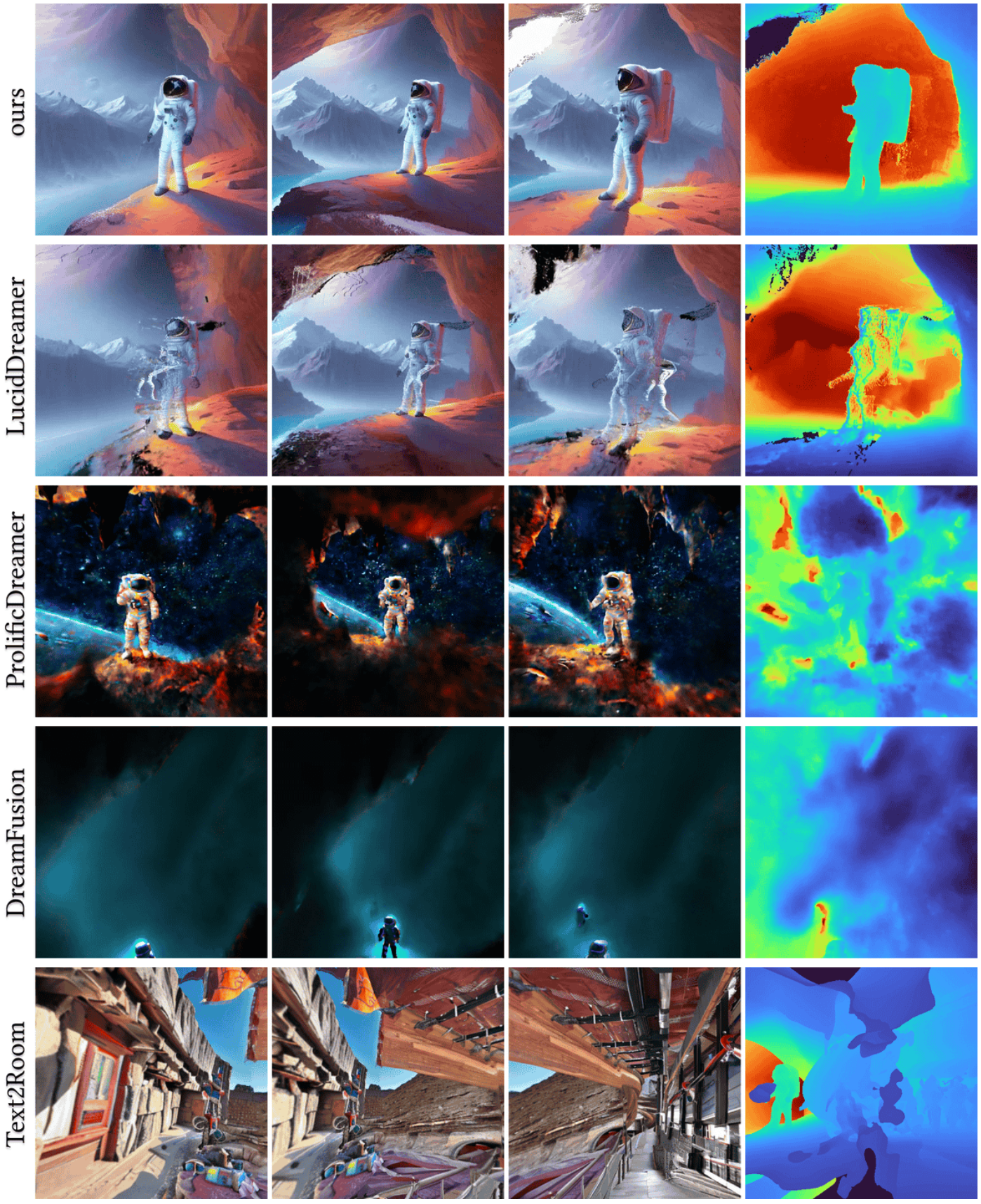
G. Limitations

Janus Problem. By adopting a distillation based approach, we occasionally encounter the Janus problem, where the face of an object appears multiple times across renders. An example is shown in Fig. 12 As we focus on scene generation and additionally condition on the 3D scene, this is less pronounced than in object generation [44] and can likely be alleviated with view-dependent prompting [2].

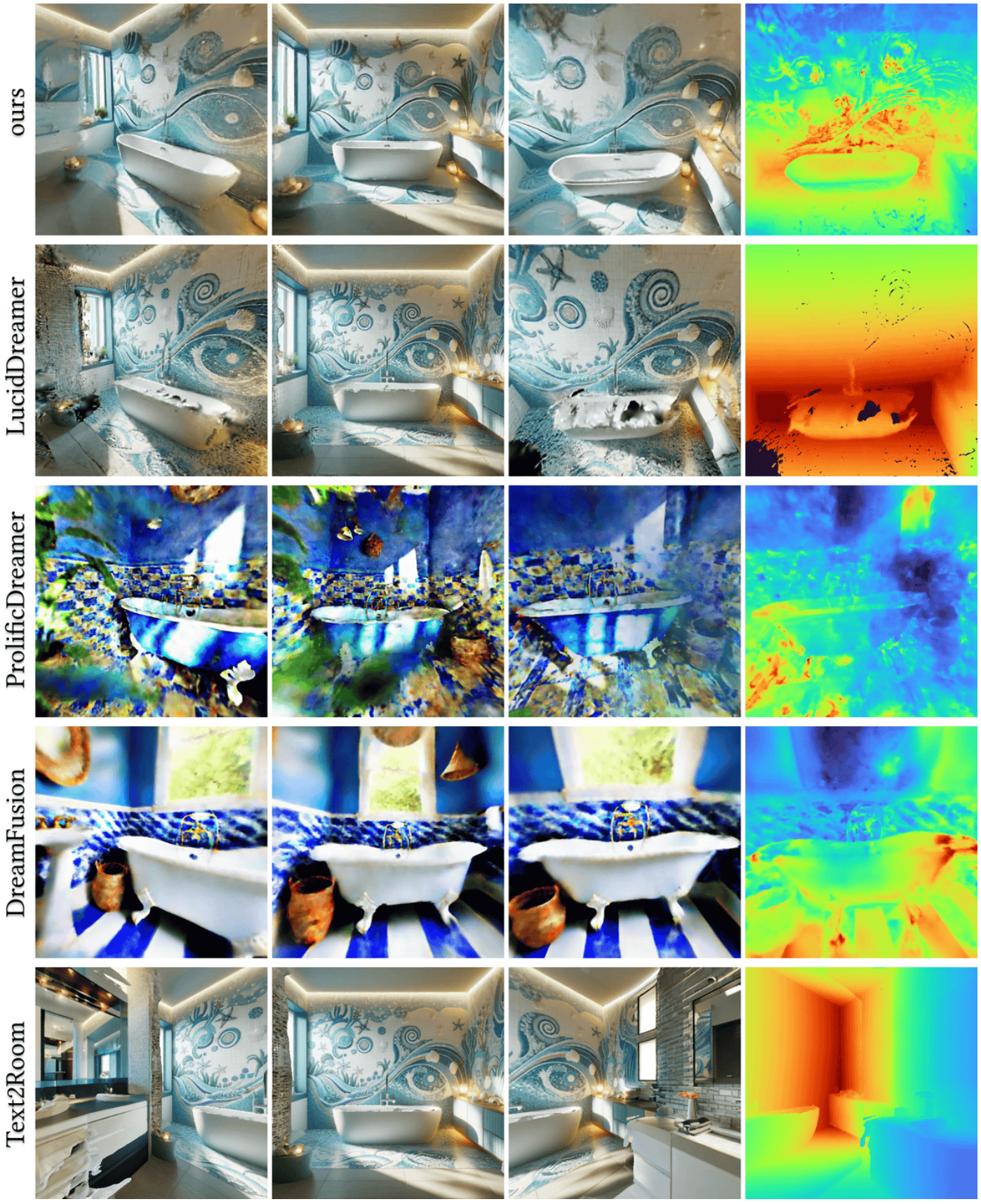
Artifacts in rendering. Some scenes also display artifacts at the surface of objects over a wide baseline. We believe improvements to our 3DGS implementation, such as by incorporating anti-aliasing, and surface regularizers might help with this. We note that our results are still significantly better than prior work and uses only 2D priors.

H. Additional Qualitative Results

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

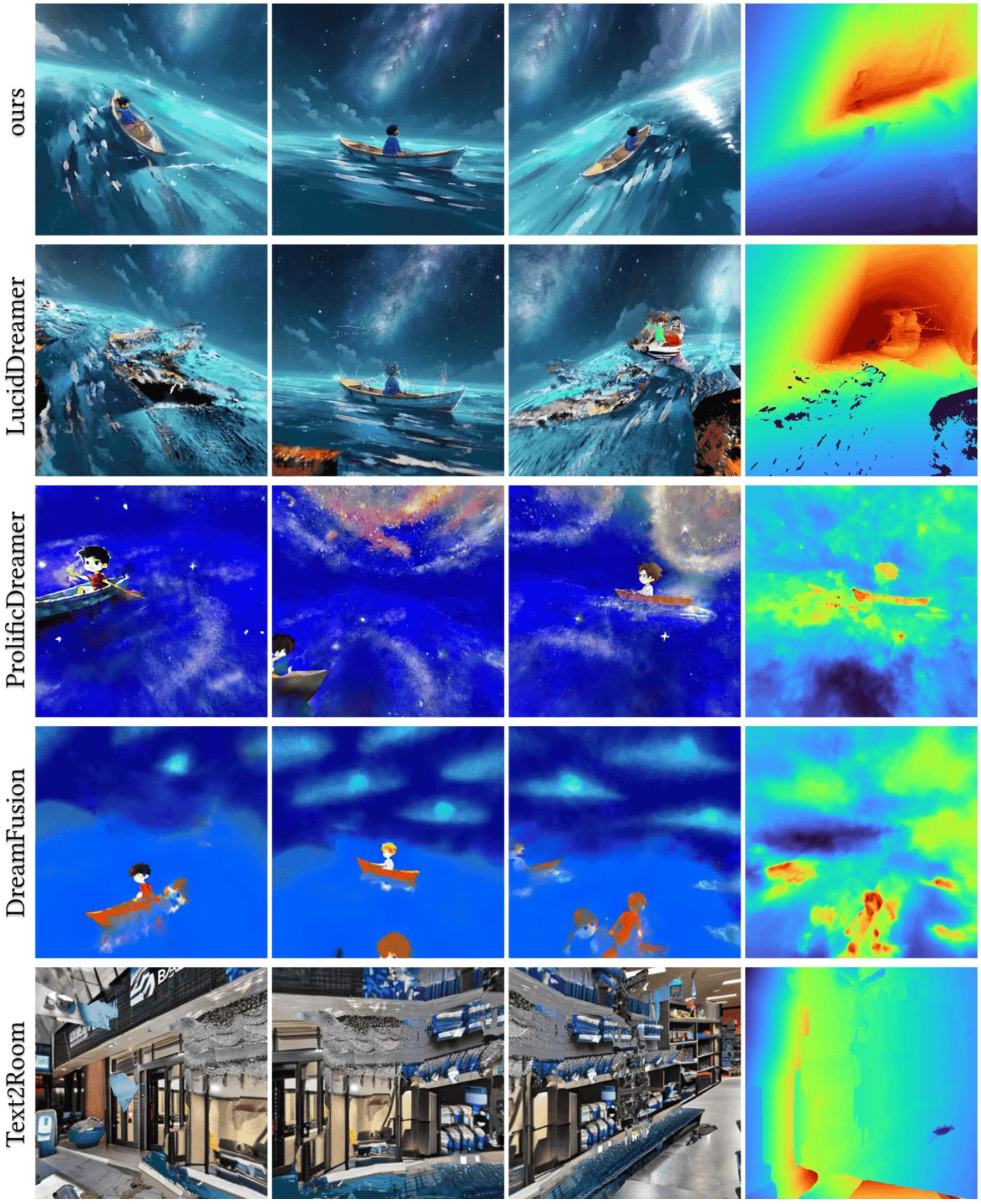


Prompt: ”An astronaut in a cave, trending on artstation, 8k image”

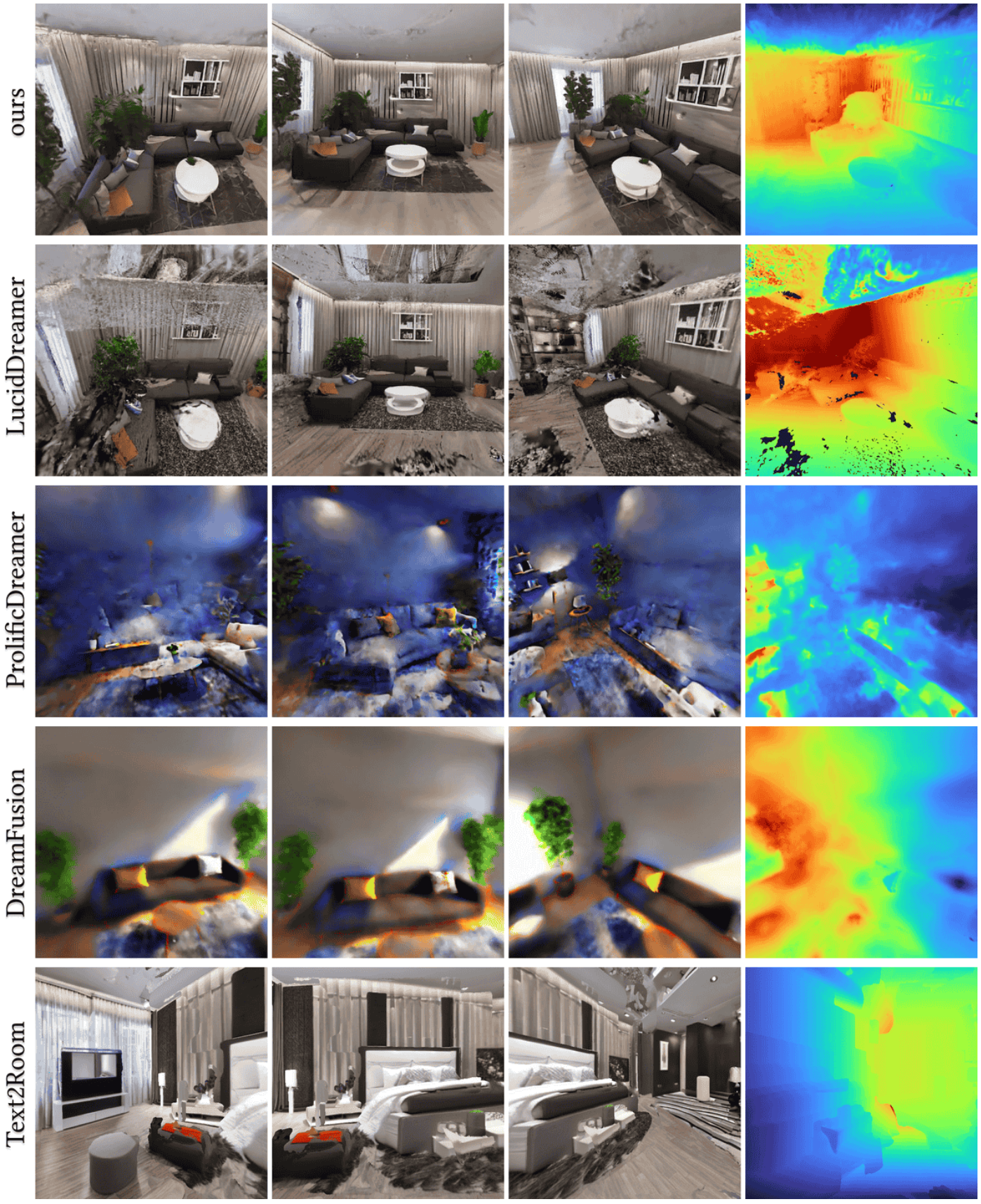


Prompt: ”Editorial Style Photo, Coastal Bathroom, Claw-foot Tub, Seashell, Wicker, Mosaic Tile, Blue and White”

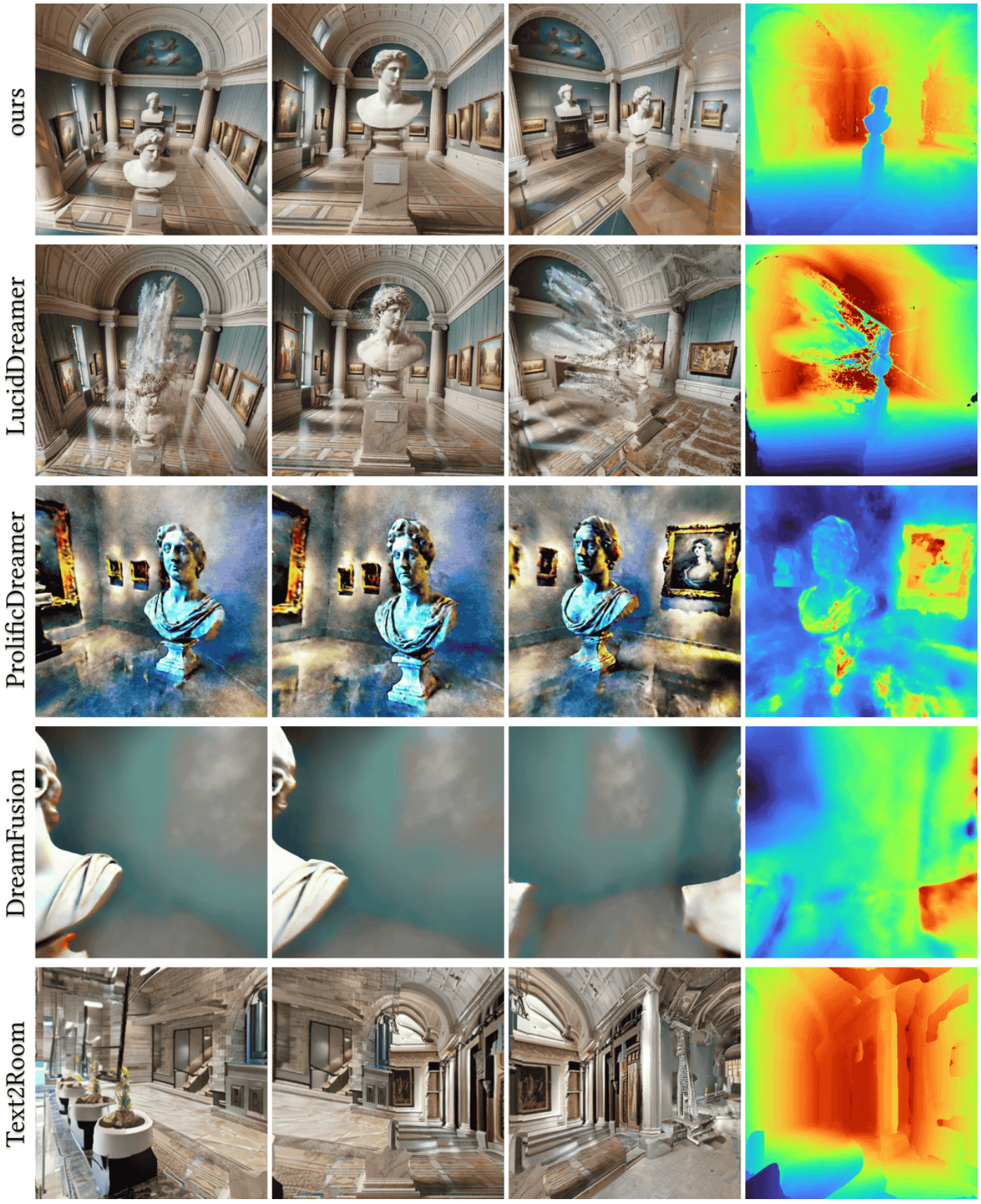
Prompt: ”A minimalist bedroom, 4K image, high resolution”



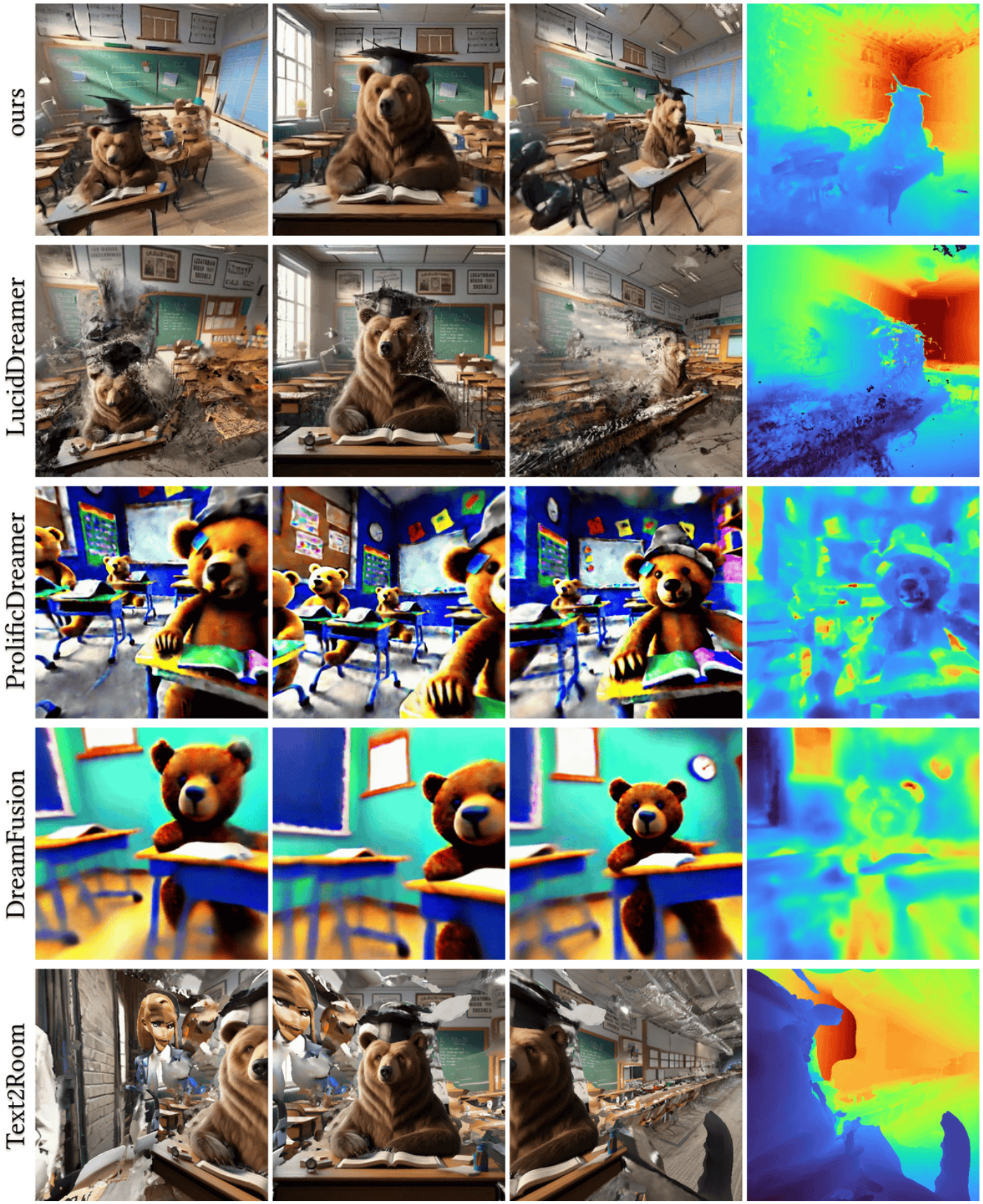
Prompt: ”A boy sitting in a boat in the middle of the ocean, under the milkyway, anime style”



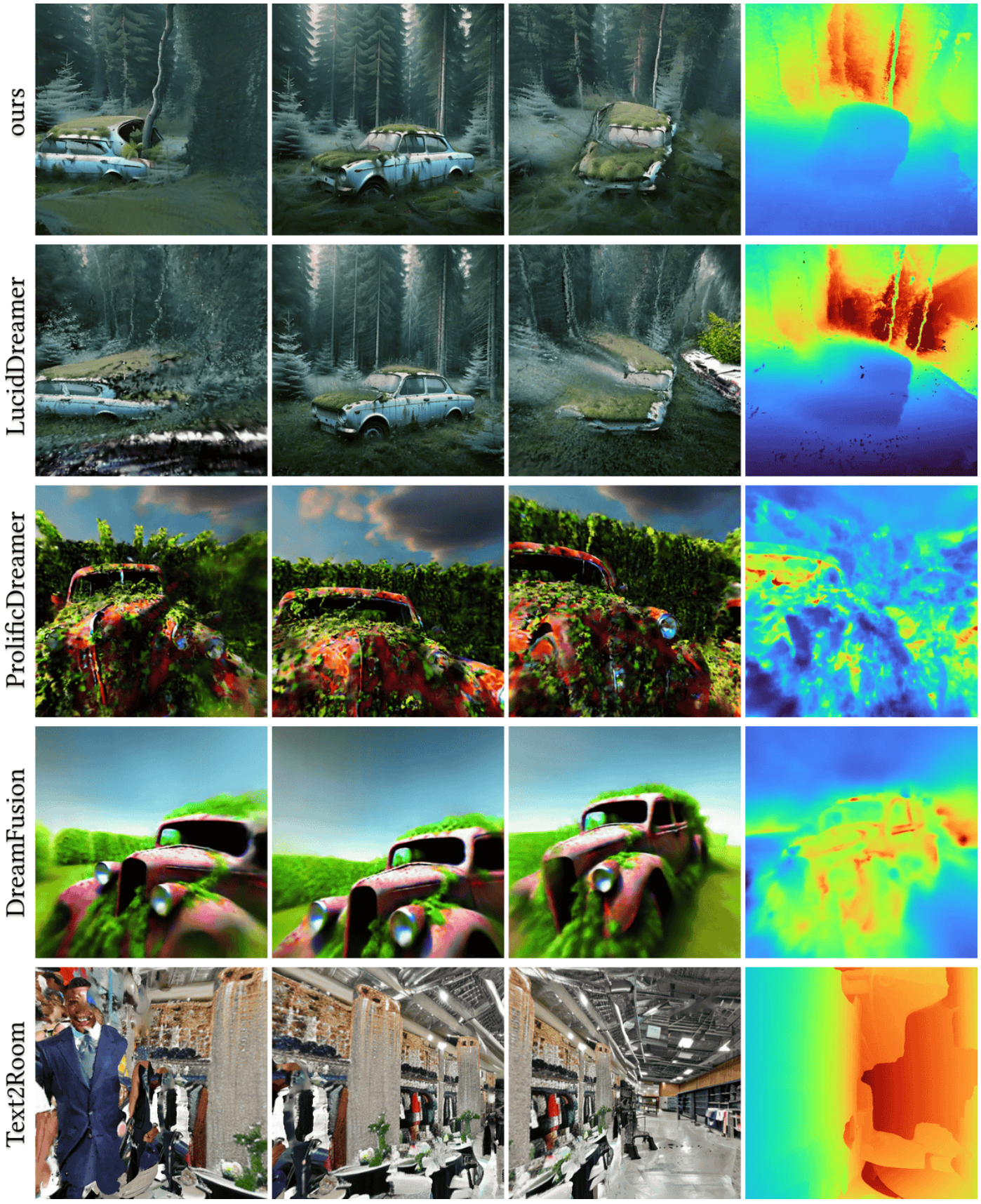
Prompt: ”a living room, high quality, 8K image, photorealistic”



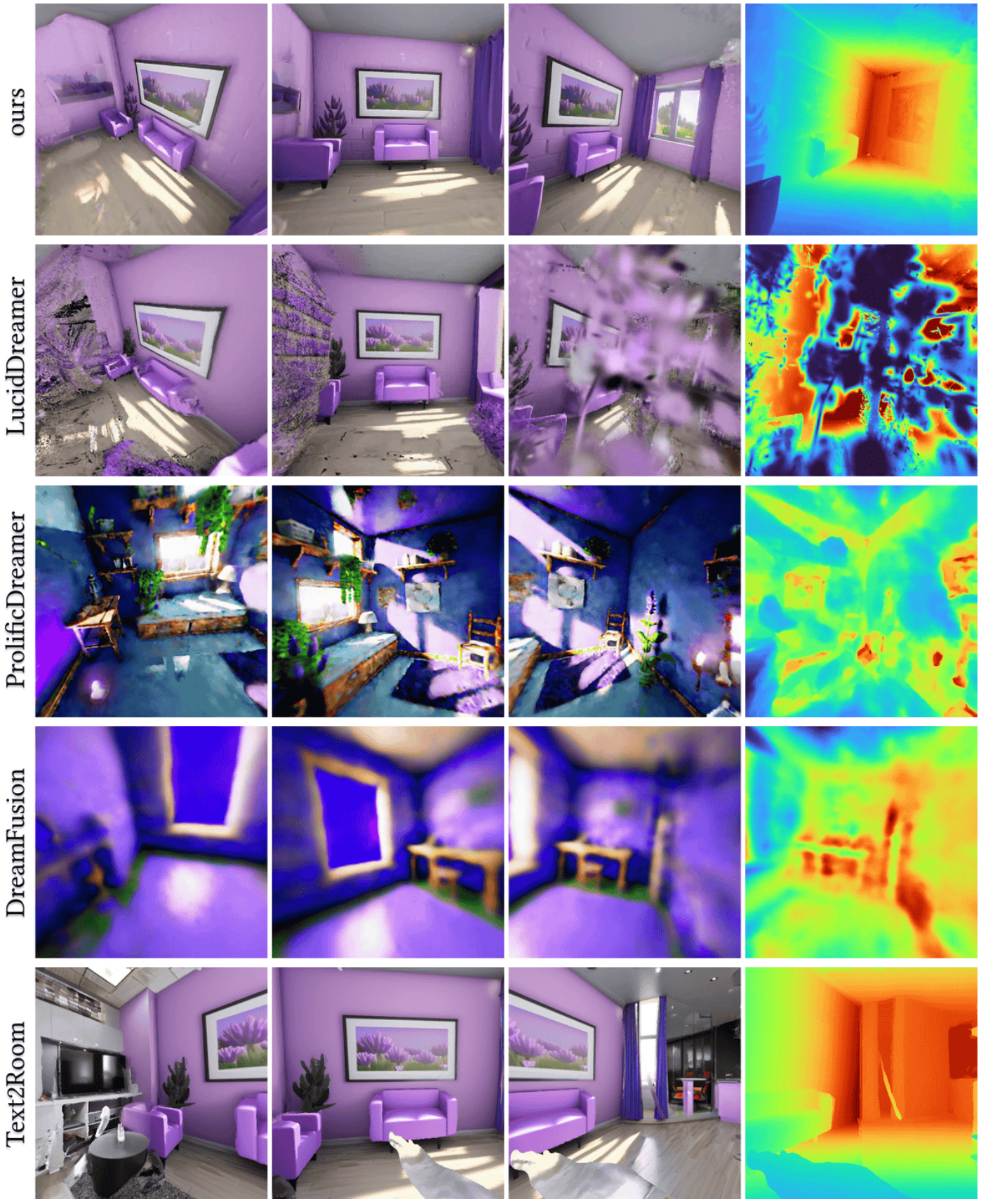
Prompt: ”A marble bust in a museum with pale teal walls, framed paintings, marble patterned floor, 4k image”



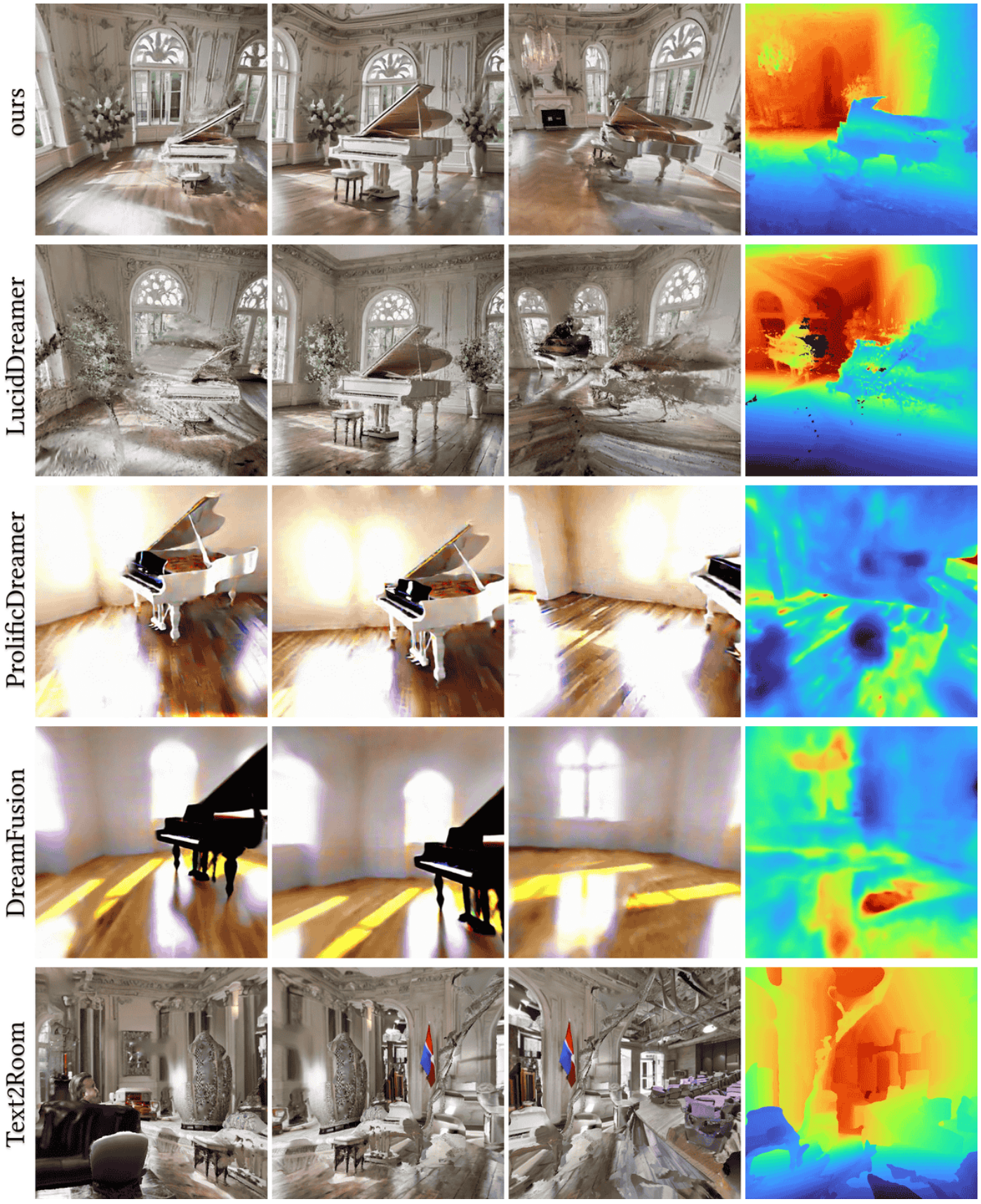
Prompt: ”A bear sitting in a classroom with a hat on, realistic, 4k image, high detail”



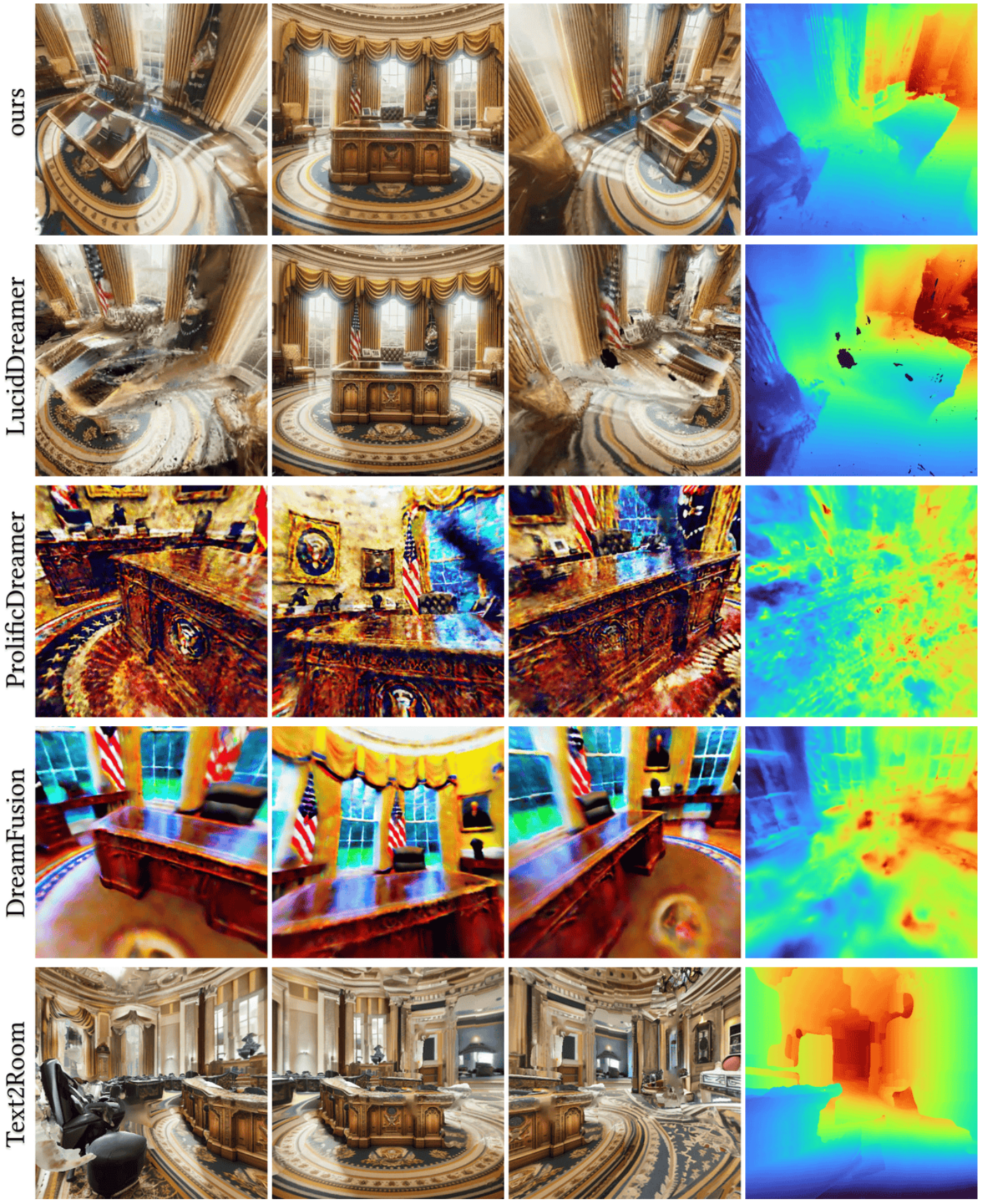
Prompt: ”An old car overgrown by vines and weeds, high quality image, photorealistic, 4k image”



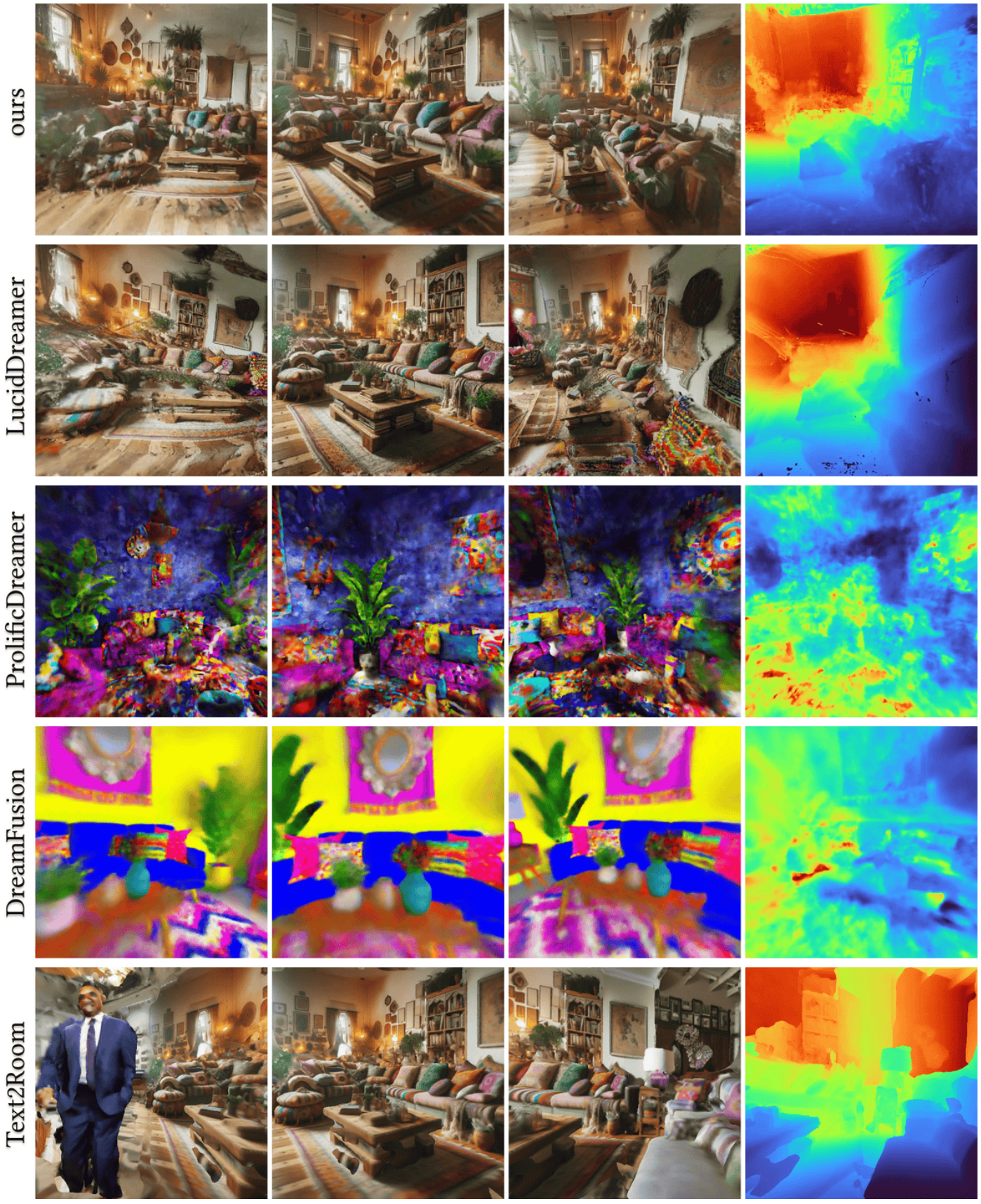
Prompt: ”Small lavender room, soft lighting, unreal engine render, voxels.”



Prompt: ”White grand piano on wooden floors in an empty hall, 4k image”

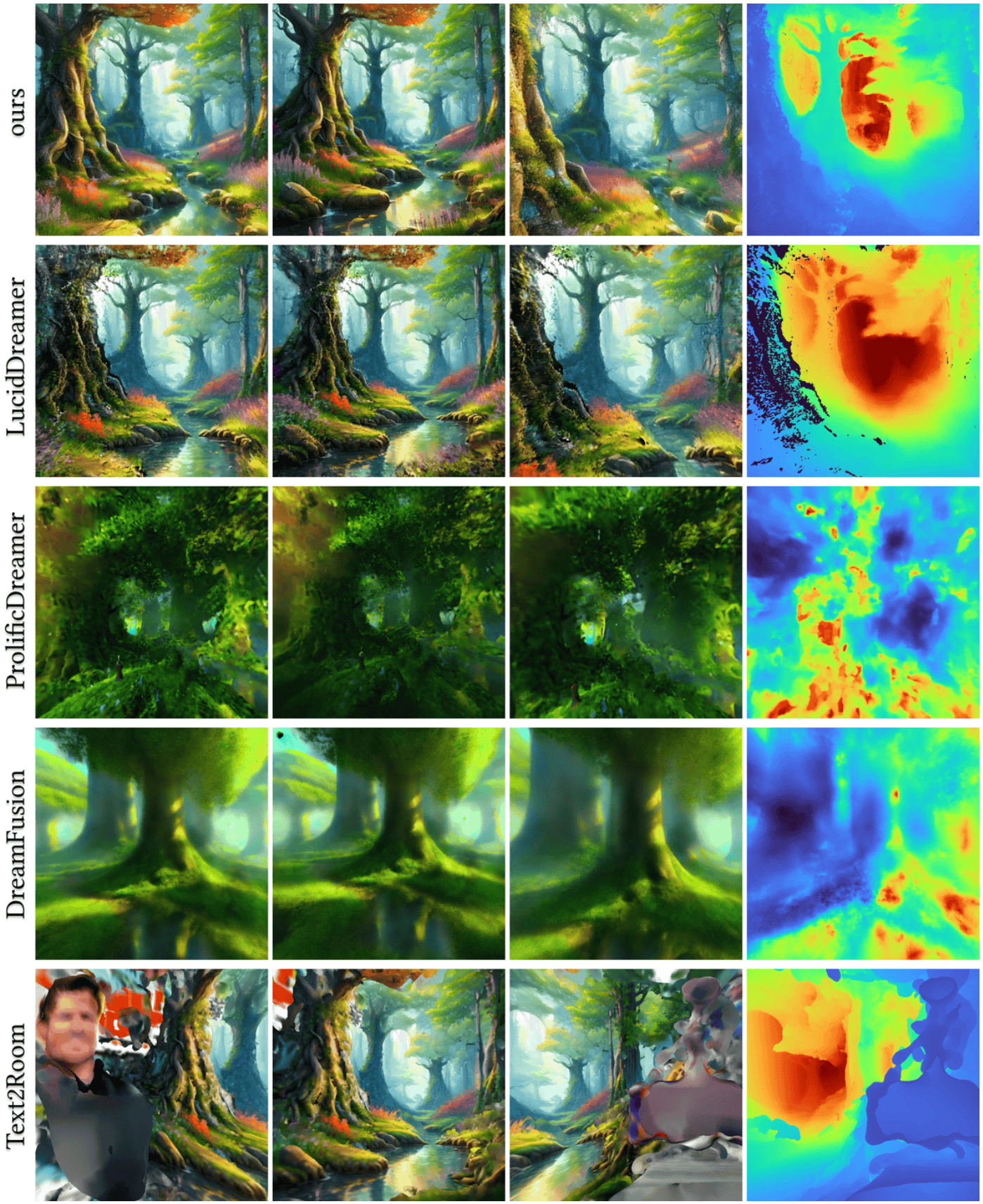


Prompt: ”A highly detailed image of the resolute desk in the oval office, 4k image”

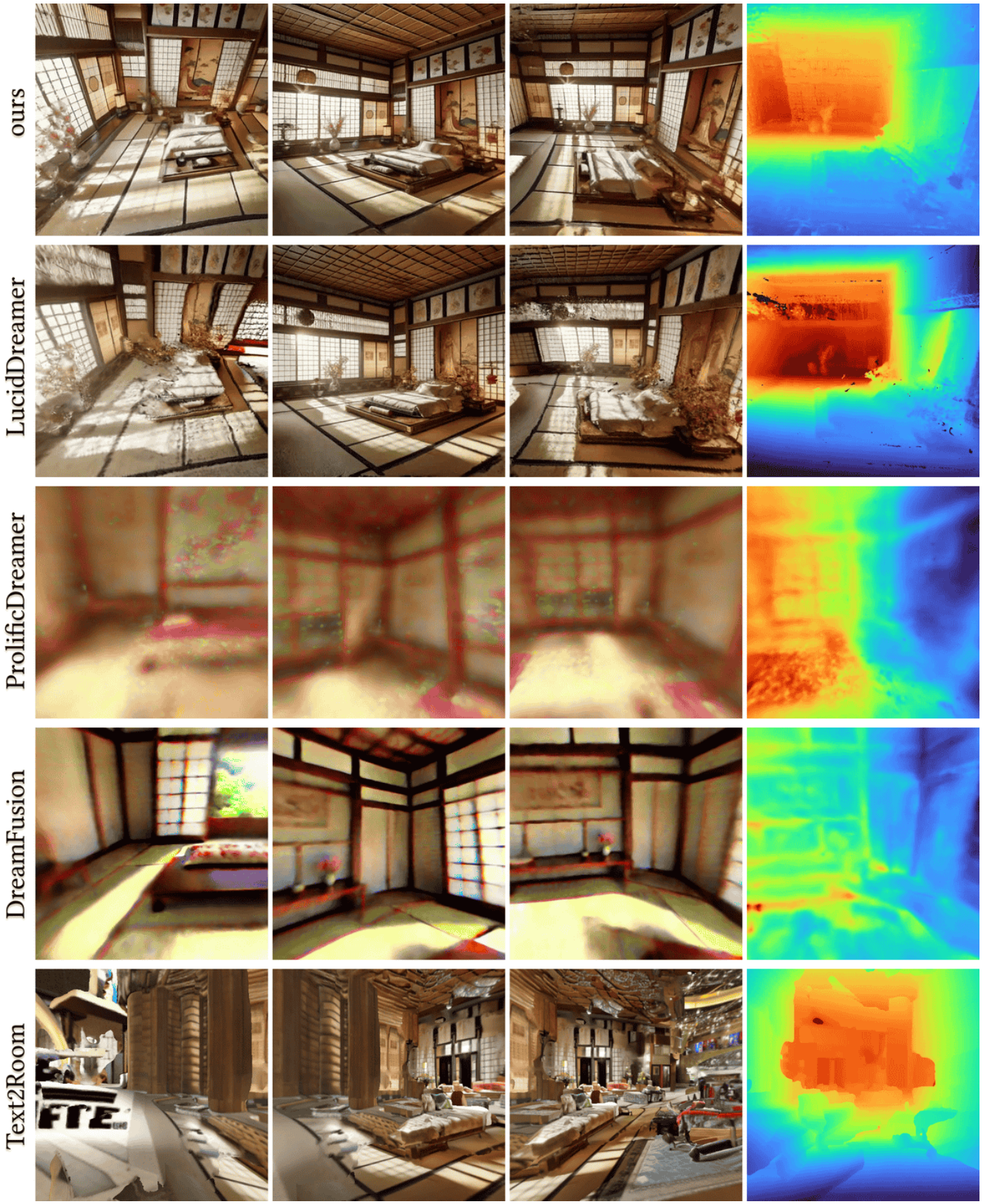


Prompt: ”A bohemian living room, colorful textiles, vibrant, eclectic, 4k image, photorealistic”

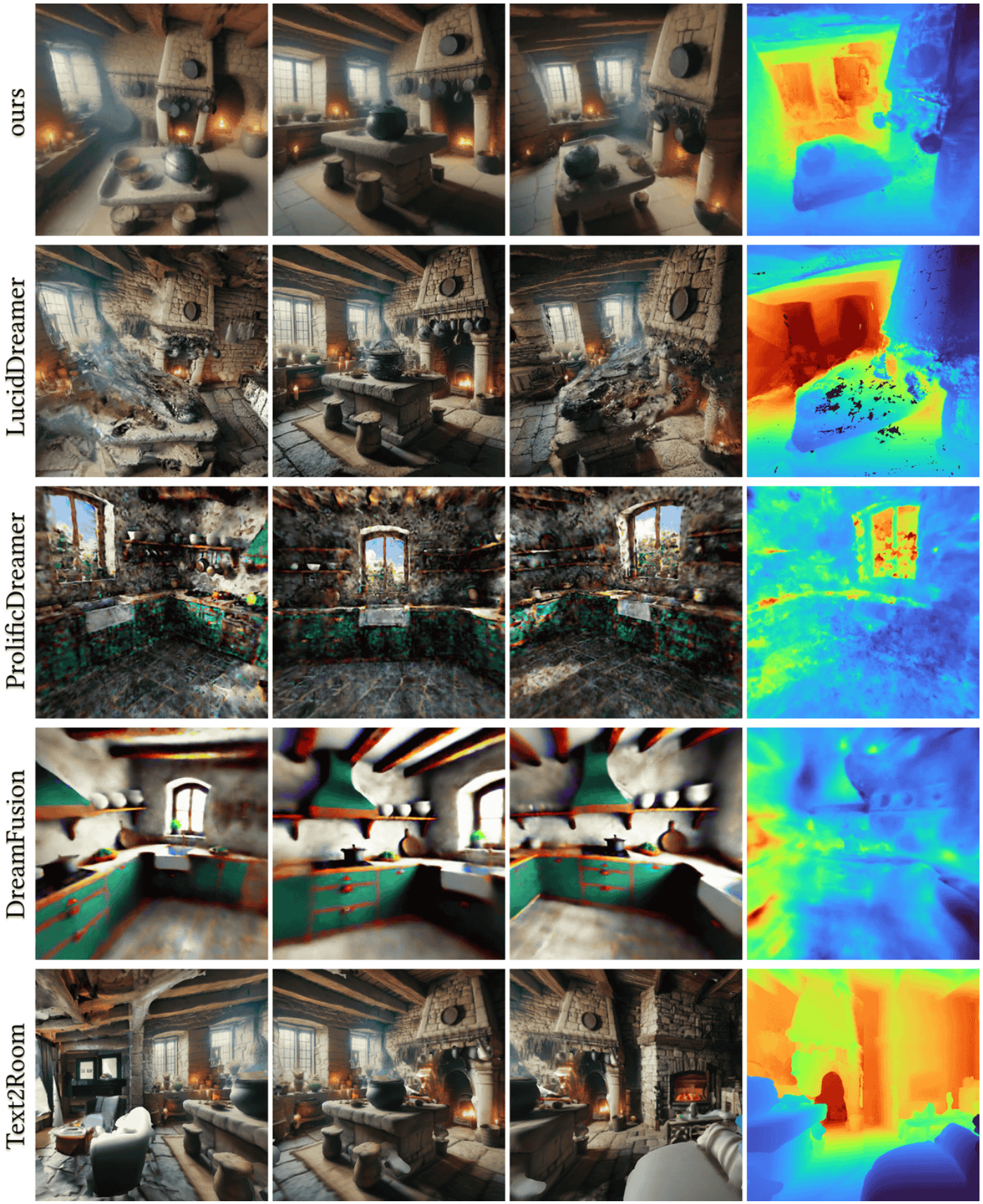
Prompt: ”Retro arcade room with posters on the walls, retro art style, illustration”



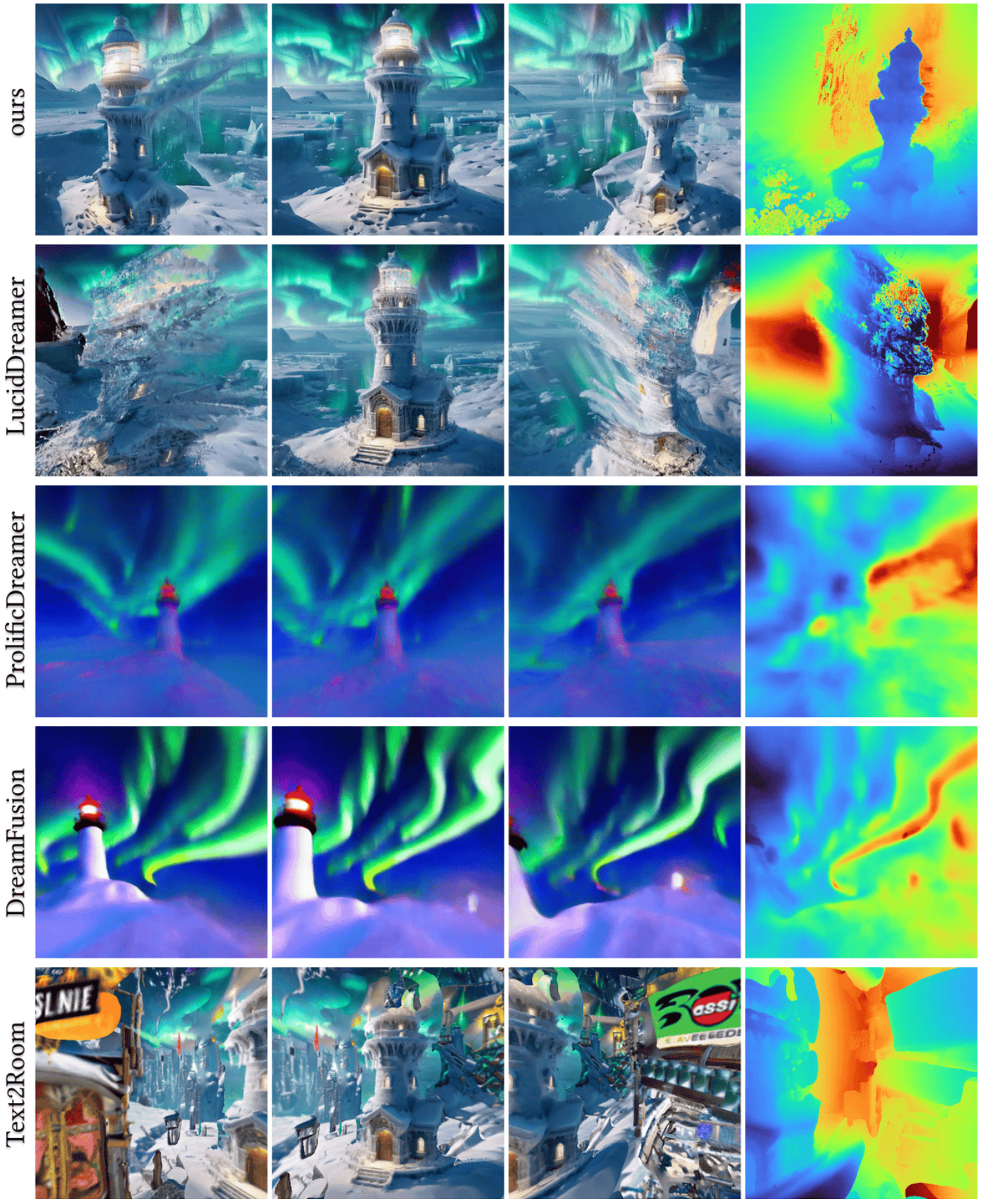
Prompt: ”A thick elven forest, fantasy art, landscape, picturesque, 4k image”



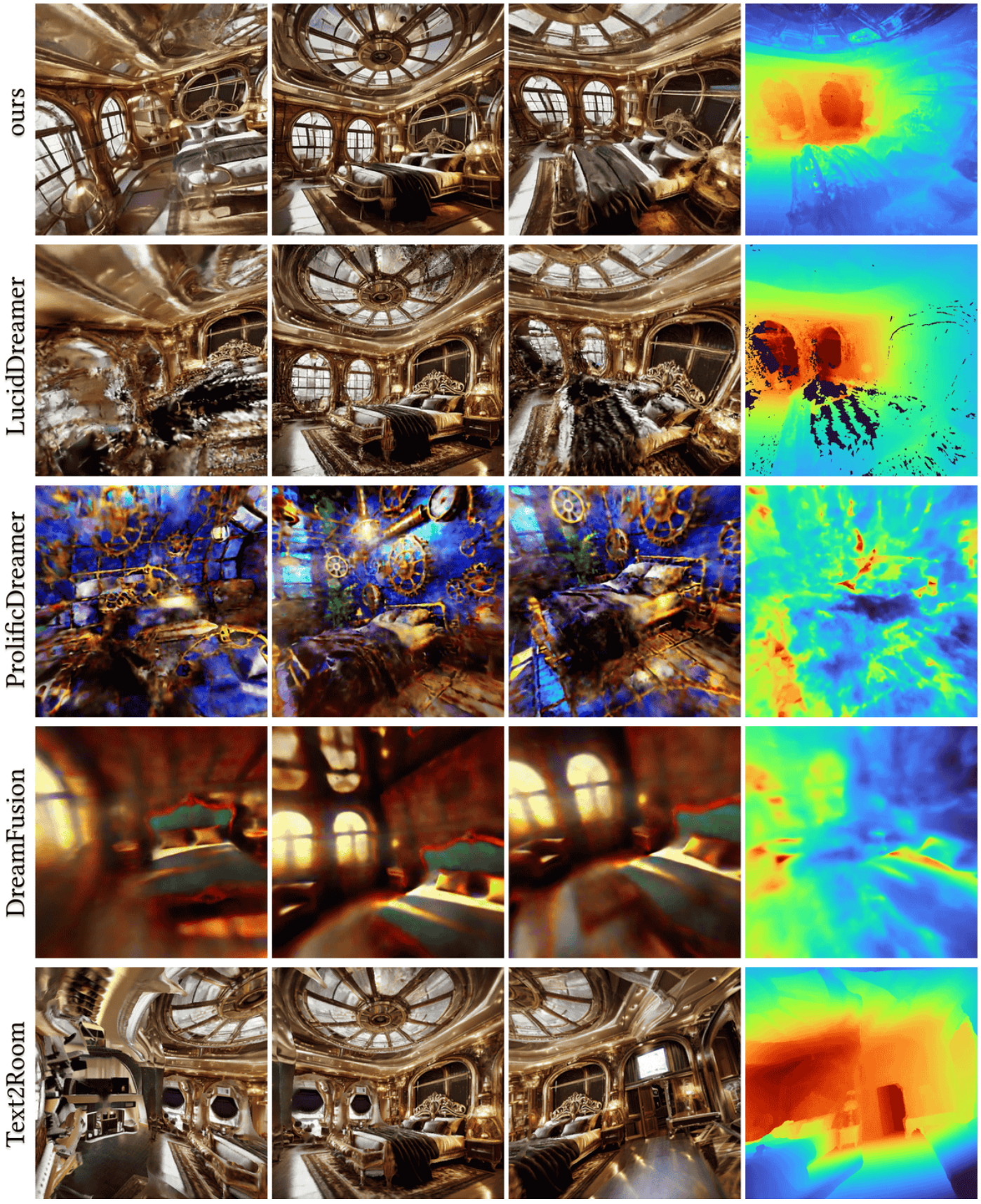
Prompt: ”A sunny royal traditional Japanese bedroom, 4k image, ornate, high detail”



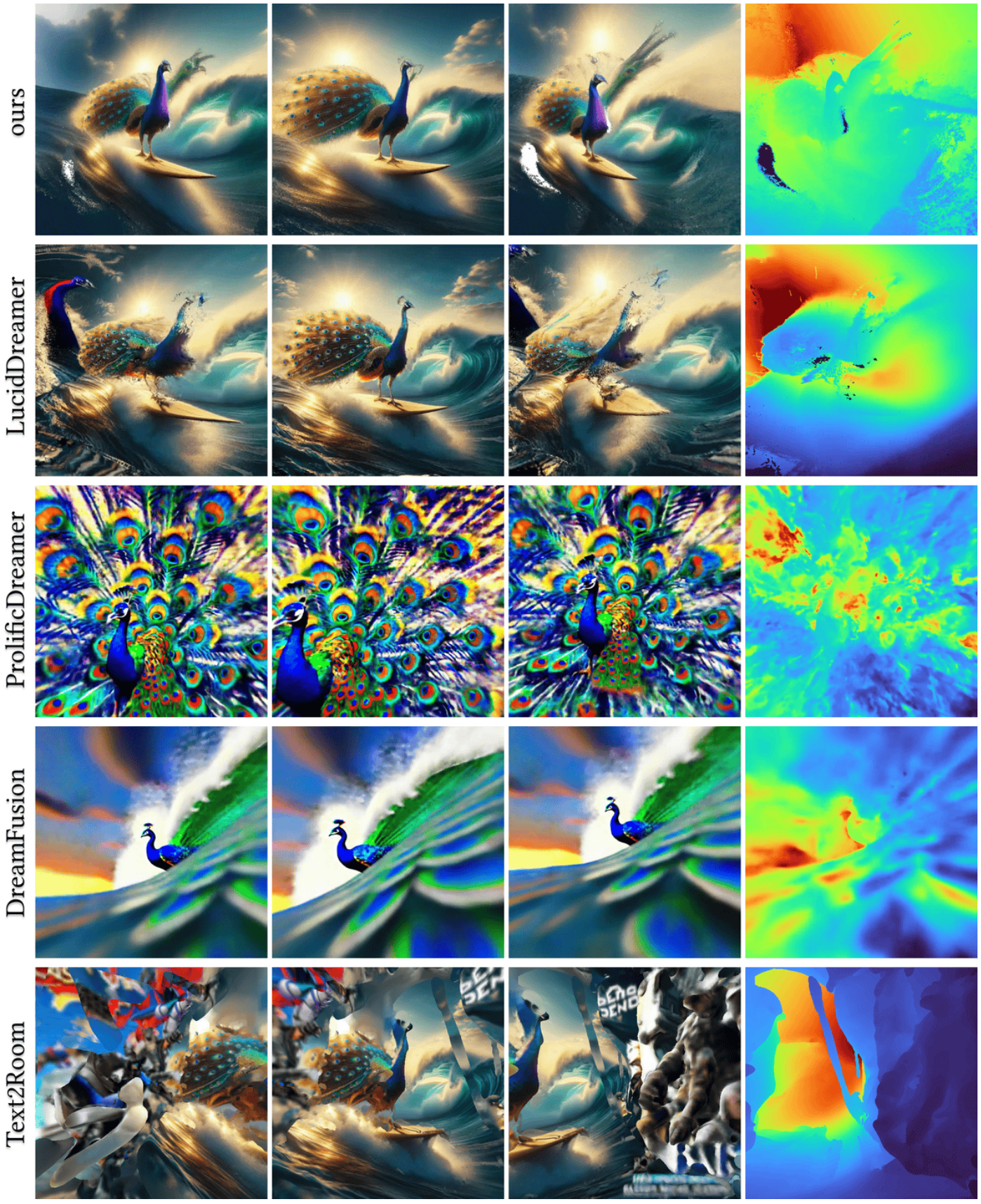
Prompt: ”An old charming stone kitchen, 4k image, photo-realistic, high detail”



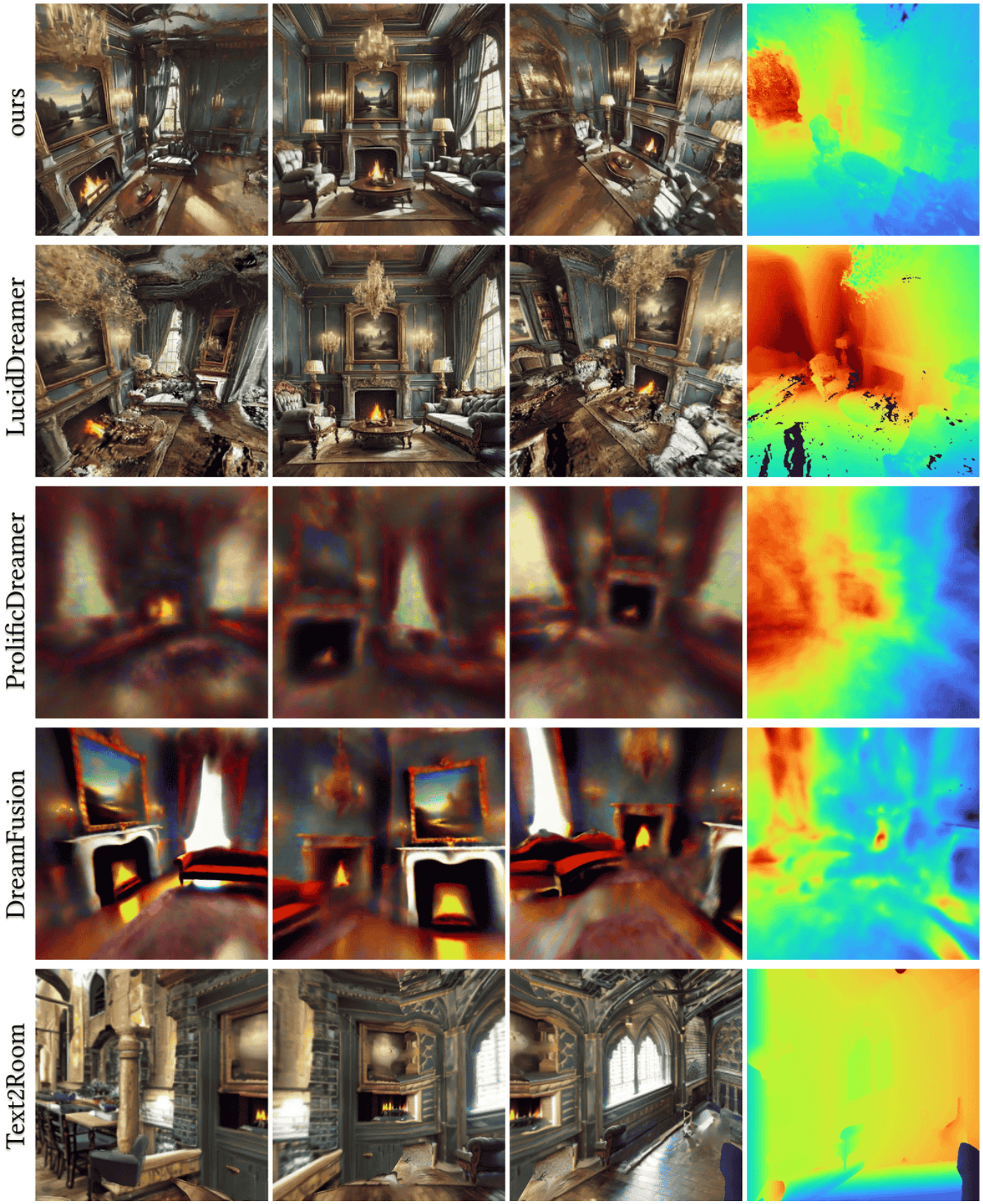
Prompt: ”Fantasy lighthouse in the Arctic, surrounded by a world of ice and snow, shining with a mystical light under the aurora borealis, 4k, sharp”



Prompt: ”A steampunk bedroom with glass ceilings, photo-realistic, 4k image, bright lighting”



Prompt: ”A majestic peacock, surfing a tall wave, photorealistic, detailed image, 4k image”



Prompt: ”A victorian living room with a grand fireplace and a long sofa, painting over the fireplace, mysterious vibe, giant windows, 4k image, photorealistic”