

# Supplemental Material: *RealmDreamer: Text-Driven 3D Scene Generation with Inpainting and Depth Diffusion*

Anonymous 3DV submission

Paper ID 312

001	<b>Contents</b>			
002	<b>1. Why is the occlusion volume important?</b>	1		
003	<b>2. Discussion on Baselines</b>	2		
004	2.1. Comparison with Dreamfusion and Prolific-			
005	Dreamer . . . . .	2		
006	2.2. Relation between SDS and our distillation . .	2		
007	2.3. Comparison with LucidDreamer and			
008	Text2Room . . . . .	2		
009	2.4. Implementation . . . . .	2		
010	<b>3. Additional ablations and discussion</b>	3		
011	3.1. Use of DDIM Inversion. . . . .	3		
012	3.2. Use of sharpening filter. . . . .	3		
013	<b>4. Additional Implementation Details</b>	3		
014	4.1. Point Cloud Generation . . . . .	3		
015	4.2. Occlusion Volume Computation . . . . .	3		
016	4.3. Optimization . . . . .	4		
017	<b>5. User Study</b>	4		
018	5.1. Common themes of the user study. . . . .	4		
	<b>6. Additional Discussion</b>	5	019	
	6.1. Impact of Distillation . . . . .	5	020	
	<b>7. Limitations</b>	5	021	
	<b>8. Additional Qualitative Results</b>	6	022	
	<b>Ethical Considerations</b>		023	
	While we use pretrained models for all components of our		024	
	pipeline, it is important to acknowledge biases and ethical		025	
	issues that stem from the training of these large-scale image		026	
	generative models [14]. As these models are often trained		027	
	on vast collections of internet data, they can reflect negative		028	
	biases and stereotypes against certain populations, as well		029	
	as infringe on the copyright of artists and other creatives. It		030	
	is essential to consider these factors when using these mod-		031	
	els and our technique broadly.		032	
	<b>1. Why is the occlusion volume important?</b>		033	
	Our key contribution is the inpainting distillation loss that		034	
	provides lower variance and high-quality supervision for		035	
	text-to-3D, compared to regular text-to-image model based		036	
	distillation, as shown in the ablations. Given that we use 2D		037	
	inpainting models for 3D inpainting via this distillation, we		038	
	must ask: <i>how to compute the 3D region that needs to be</i>		039	
	<i>inpainted?</i>		040	
	We proposed a simple technique to do so by comput-		041	
	ing the <b>occluded volume</b> (described in Sec. 4.2), which is		042	
	the 3D region occluded by objects in the reference image		043	
	$I_{\text{ref}}$ . Note that regardless of what objects may be present in		044	
	this occluded region, rendering from $P_{\text{ref}}$ would yield $I_{\text{ref}}$ ,		045	
	as elements in the image would occlude the objects. The		046	
	2D inpainting masks we obtain then must hence, reflect this		047	
	unknown 3D region, as that is the part of the scene left to		048	
	complete.		049	
	Instead of computing the occlusion volume, another al-		050	
	ternative is using the holes from point cloud renderings as		051	

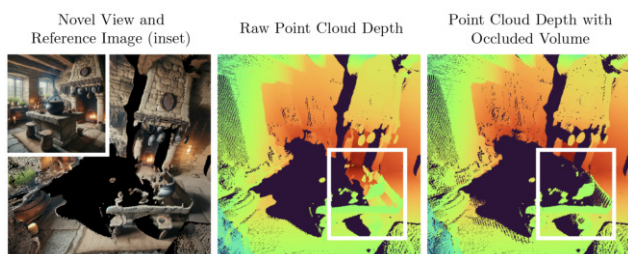


Figure 1. **Creating the Inpainting Mask.** 3D inpainting requires filling in a 3D volume, which is not always equivalent to missing regions in 2D point cloud renders. By computing an *occlusion volume*, we avoid situations where the floor is visible through the table (middle) but instead could be occluded. The right depth map accounts for the ambiguity of the volume in the table.

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. 1, 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  $I_{\text{ref}}$ , 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. 1, the quality of inpainted samples would be incorrect and lead to a noisier distillation process.

## 2. 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.

### 2.1. Comparison with Dreamfusion and ProlificDreamer

By comparing with prior SOTA distillation techniques [12, 17], 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 [17] 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.

### 2.2. Relation between SDS and our distillation

Our distillation is similar to the score distillation loss (SDS) used in Dreamfusion [12]. 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.

### 2.3. 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 [5], 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  $I_{\text{ref}}$  entirely. In contrast to this, our technique maintains a simple initialization strategy and relies on a high-quality inpainting distillation process to fill-in missing regions, without sacrificing the quality of the initialized regions.

### 2.4. Implementation

**Text2Room [8] and LucidDreamer [5]** We use the official implementation of Text2Room and LucidDreamer on Github. To ensure a fair comparison, we estimate depth using Marigold [9] and DepthAnything [18] as in our technique, replacing the original IronDepth [2] and ZoeDepth [4] respectively. The rest of the pipeline is kept the same.



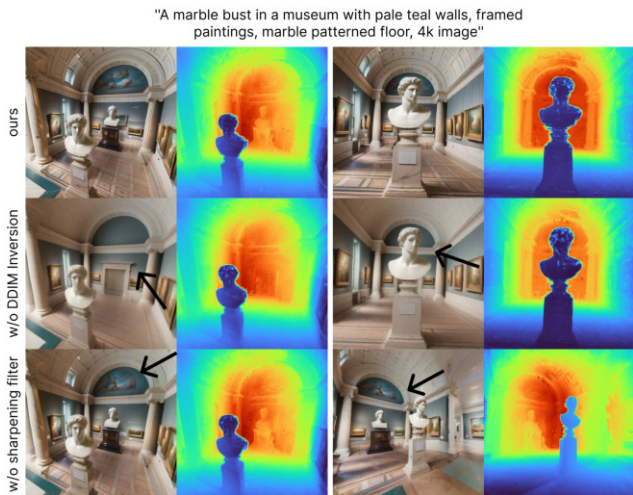


Figure 2. We ablate the importance of DDIM Inversion and applying a sharpening filter. As in [10], we find that DDIM inversion allows more details to be synthesized by our method. Additionally, we find that detail slightly increases when applying a sharpening filter to the sampled images.

**ProlificDreamer [17] and Dreamfusion [12]** We use the implementation of these baselines provided in threestudio [6] 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.

### 3. Additional ablations and discussion

#### 3.1. 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 [16], 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 [7, 10]. We demonstrate the importance of doing so in Fig. 2, 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.

#### 3.2. Use of sharpening filter.

In Fig. 2, 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.

## 4. Additional Implementation Details

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

### 4.1. 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 [11], Adobe Firefly, and DALLE-3 [3].

**Depth Estimation.** As mentioned earlier, we use Marigold [9] as our depth estimation model, with absolute depth obtained using DepthAnything [18]. 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 [8] and align the predicted depth with the ground truth depth rendered via Pytorch3D [13] 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".

### 4.2. 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  $\mathcal{O}$ .

### 4.3. 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 [15] to personalize the text-to-image diffusion model with the reference image  $I_{\text{ref}}$  and associated prompt  $T_{\text{ref}}$ . We find that this helps the final 3D model adhere closer to  $I_{\text{ref}}$  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  $\sigma_i$  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  $M_{\text{occl}}$  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.

## 5. User Study

For comparison with ProlificDreamer [17], DreamFusion [12], and LucidDreamer [5], 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 [8], 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  $P_{\text{ref}}$ .

### 5.1. 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 [17] 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 Sec. 8. 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 [12] 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 [8] 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 [5]’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.



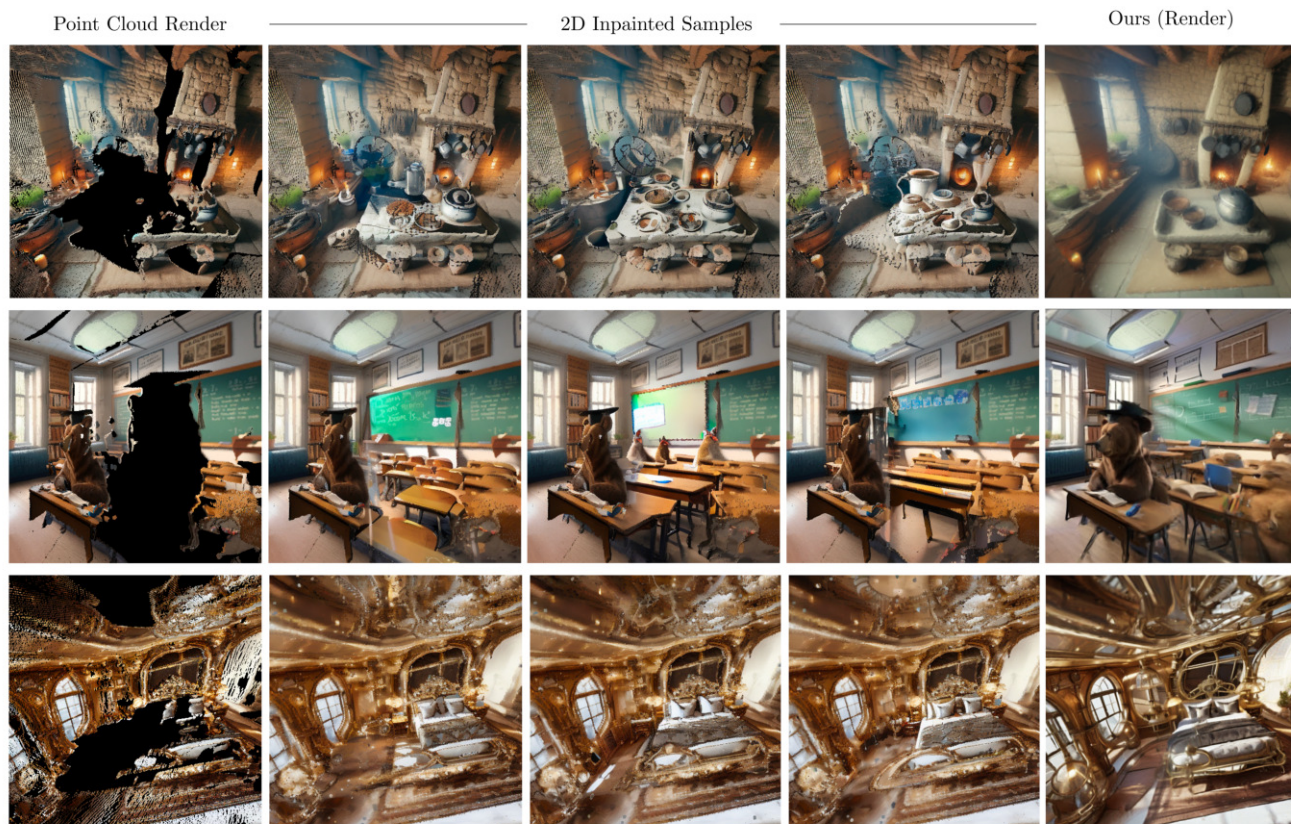


Figure 3. **Comparison of sampling from 2D inpainting models and our optimized model.** Left: Renders from the point cloud generated in stage 1. Middle (cols 2-4): Inpainted Samples of the previous render using an occlusion-based inpainting mask and Stable Diffusion [14]. Right: A render from our final 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.

## 6. Additional Discussion

### 6.1. Impact of Distillation

In Fig. 3, 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. 3, 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. 3 is clean and free of stray artifacts such as bright colours or ambiguous objects. We attribute this difference to our dis-

tillation 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 [8] 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.

## 7. 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. 4 As we focus on scene generation and additionally condition on the 3D scene, this is less pronounced than in object generation [12] and can likely be



Figure 4. **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”

alleviated with view-dependent prompting [1].

**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.

## 8. Additional Qualitative Results

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



ours



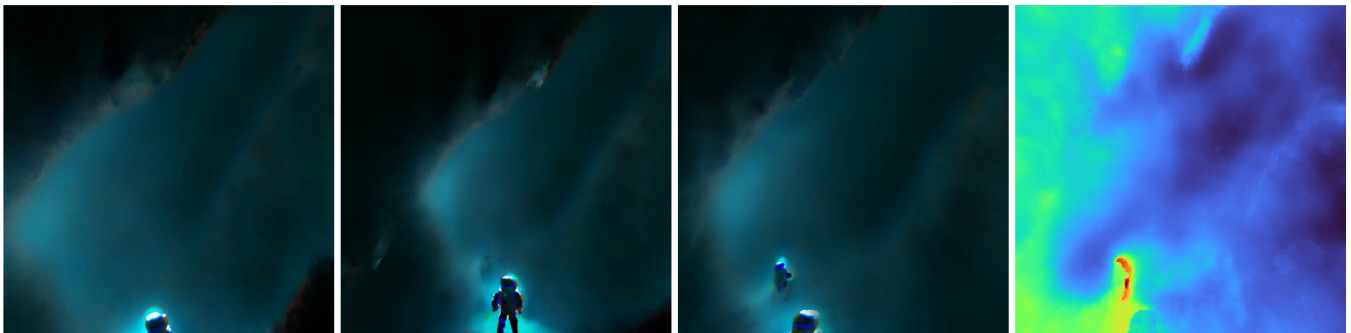
LucidDreamer



ProlificDreamer



DreamFusion



Text2Room



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



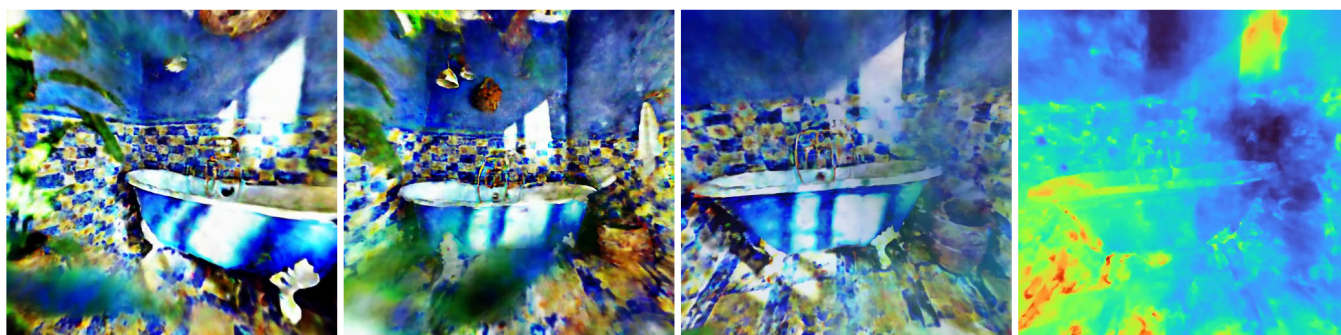
ours



LucidDreamer



ProlificDreamer



DreamFusion

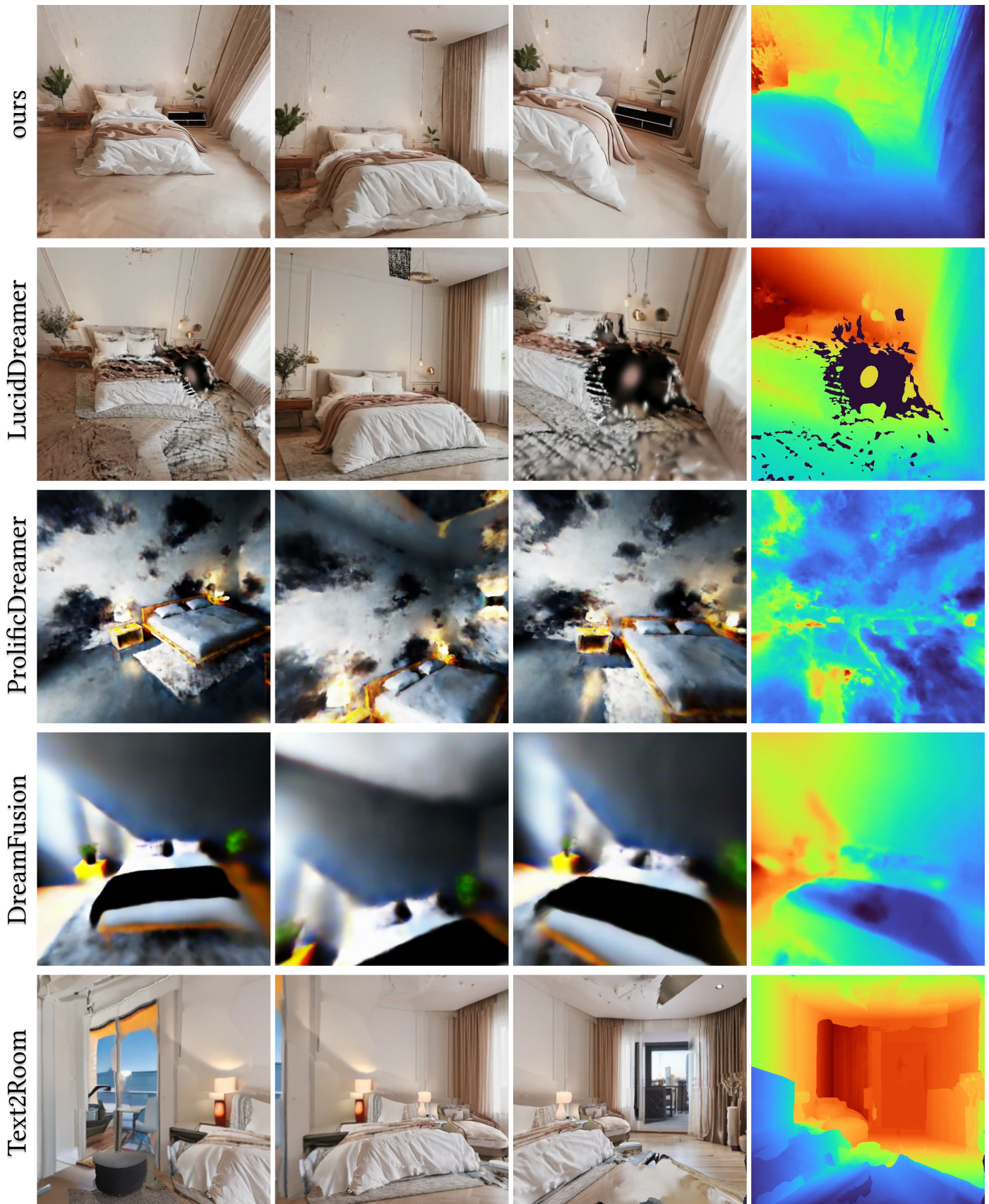


Text2Room



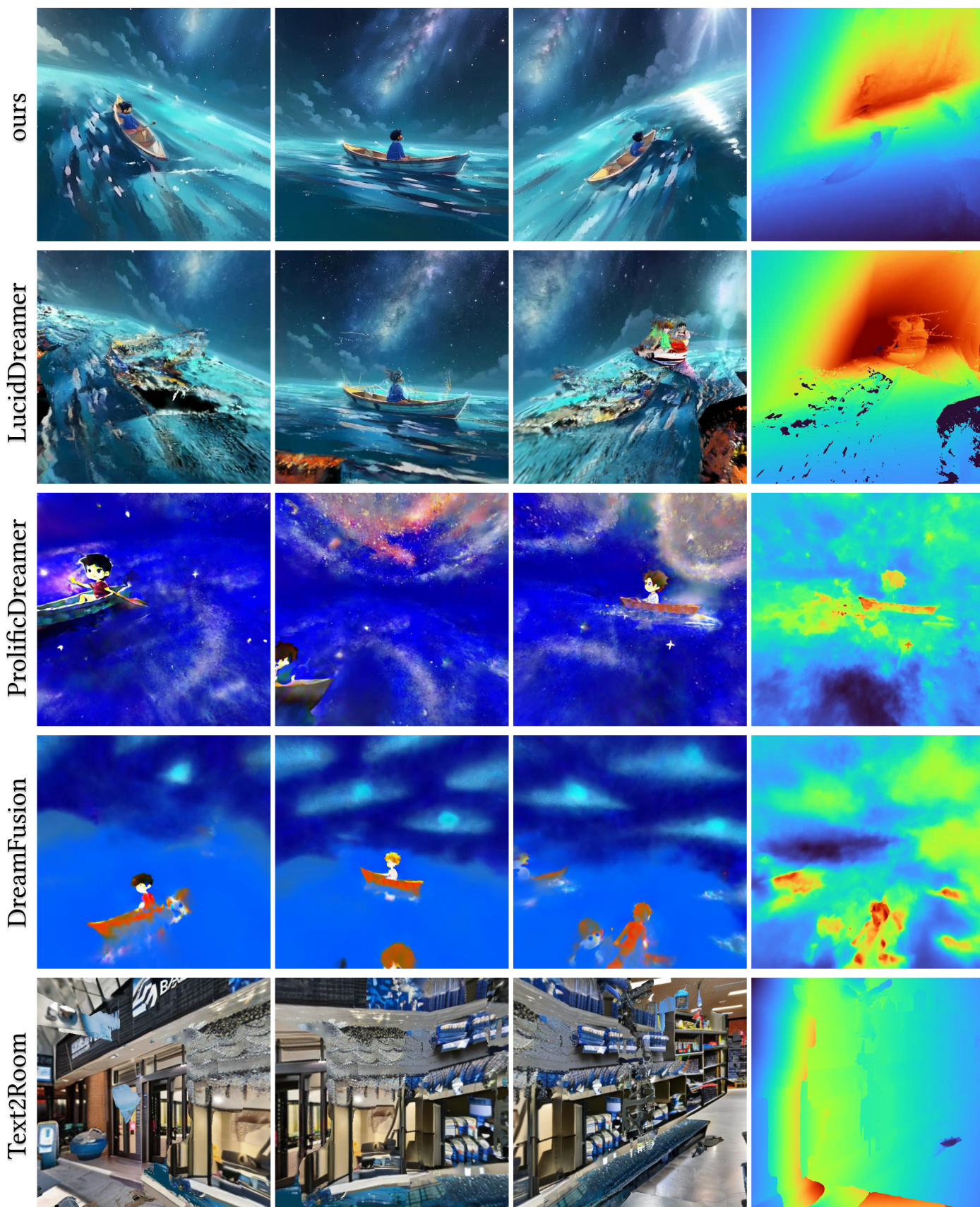
Prompt: "Editorial Style Photo, Coastal Bathroom, Claw-foot Tub, Seashell, Wicker, Mosaic Tile, Blue and White"





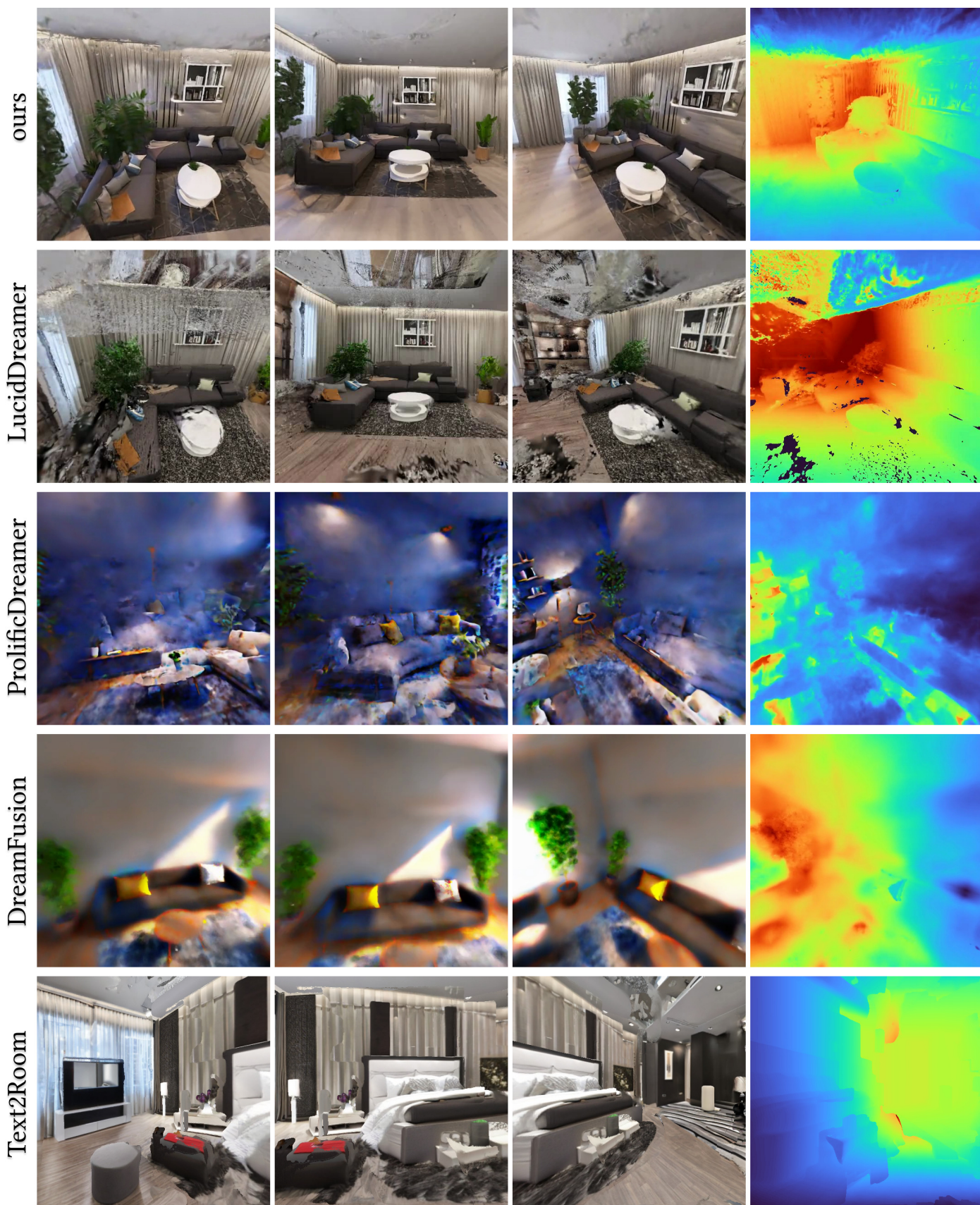
Prompt: "A minimalist bedroom, 4K image, high resolution"





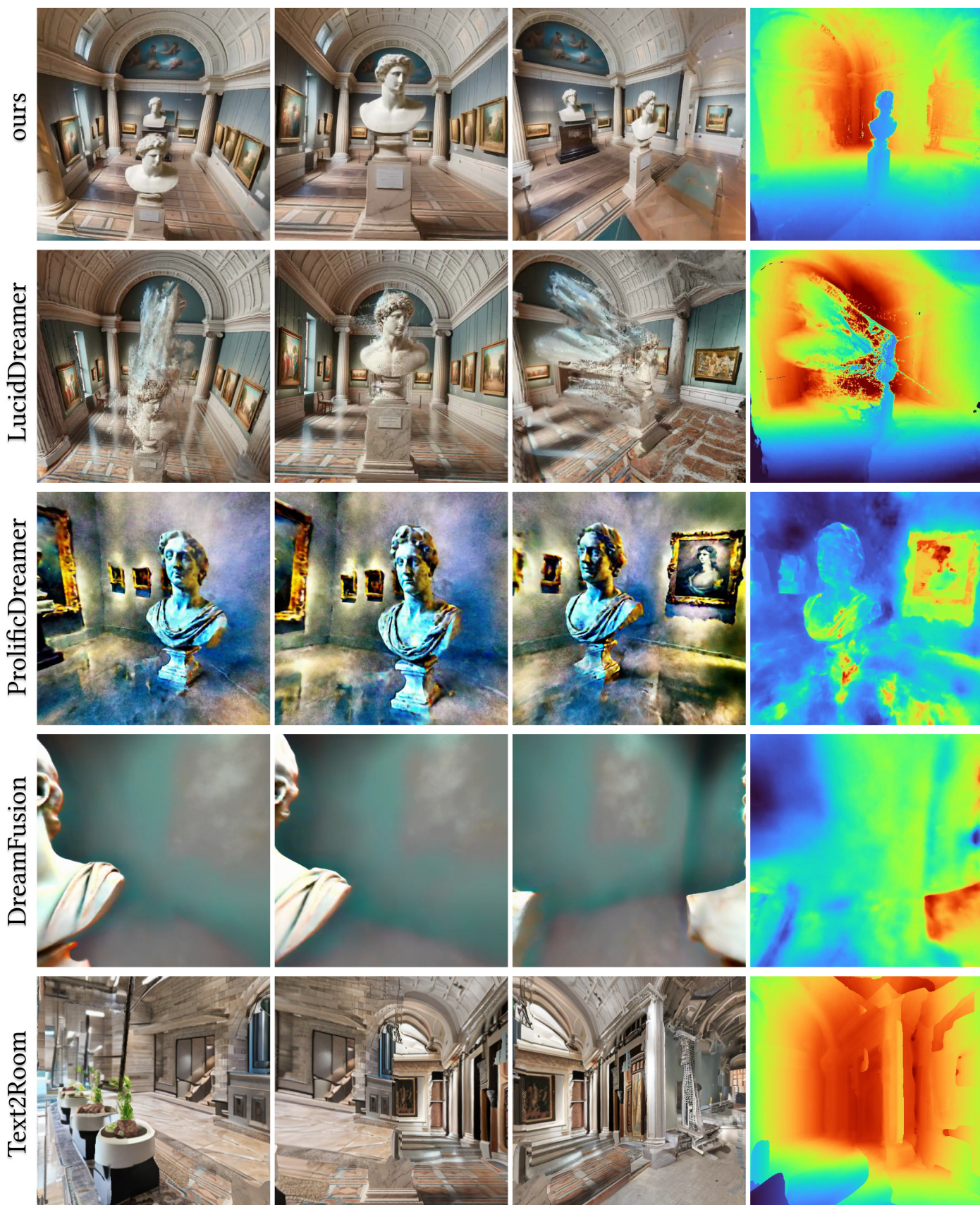
Prompt: "A boy sitting in a boat in the middle of the ocean, under the milkyway, anime style"





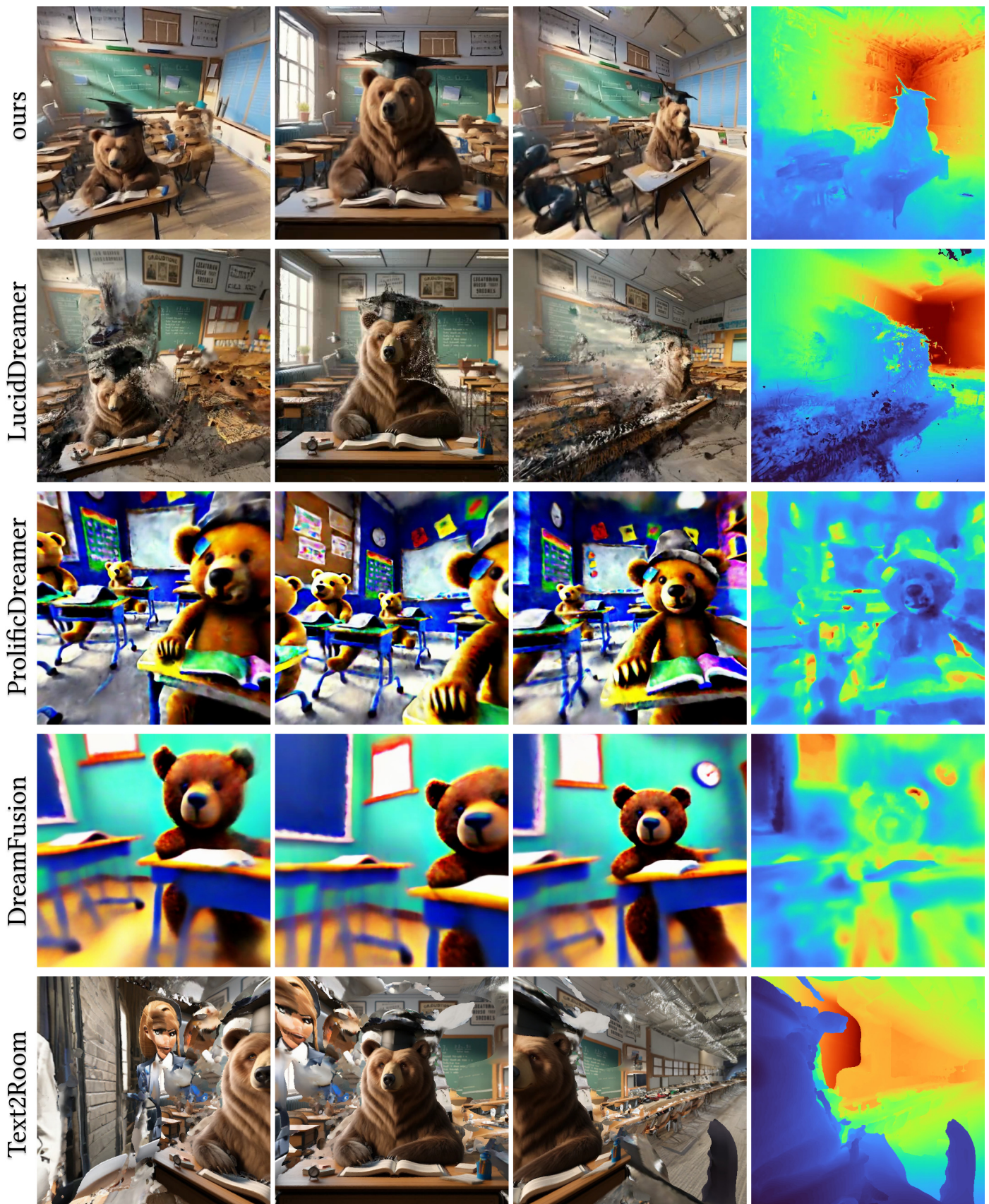
Prompt: "a living room, high quality, 8K image, photorealistic"





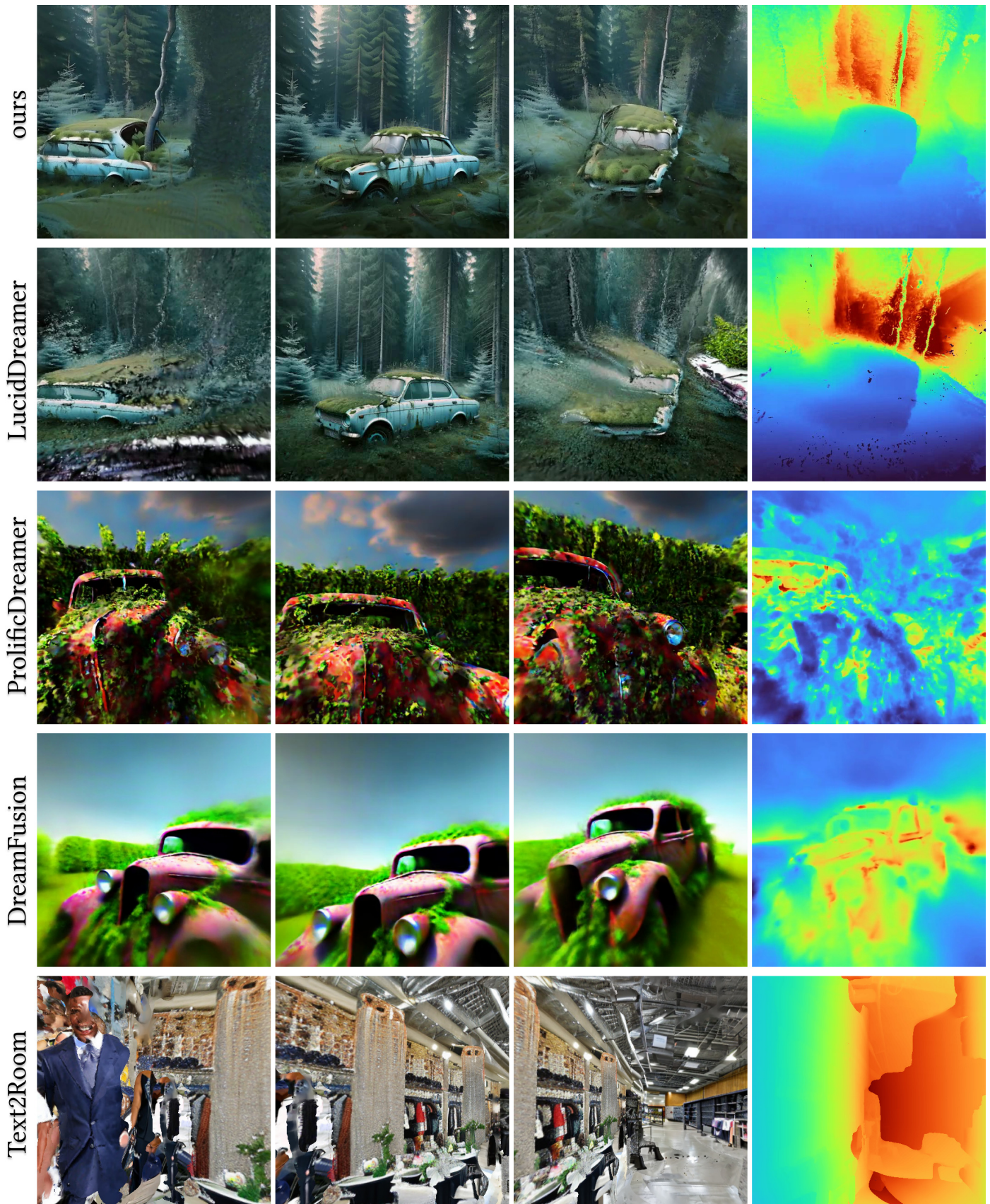
Prompt: "A marble bust in a museum with pale teal walls, framed paintings, marble patterned floor, 4k image"





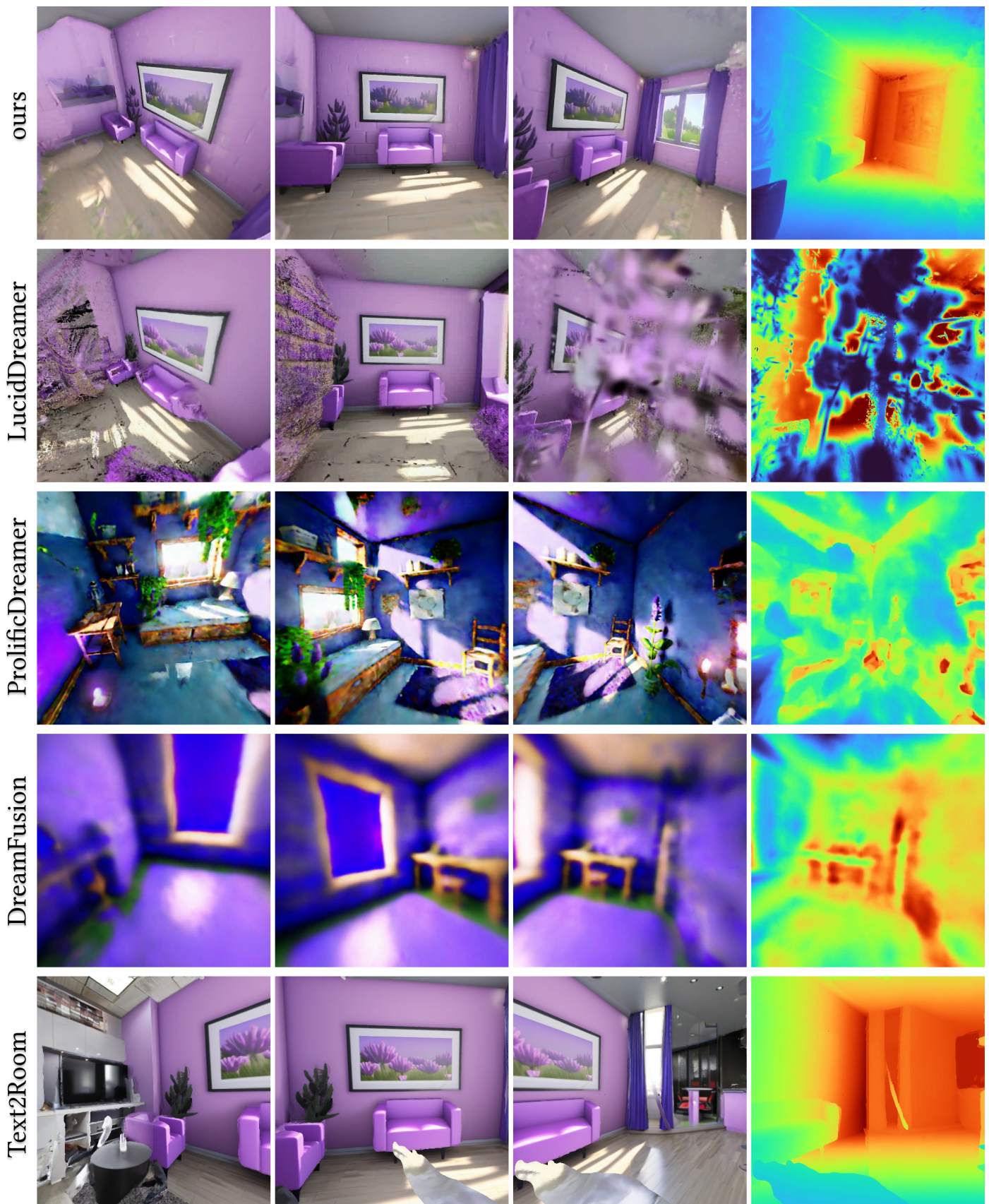
Prompt: "A bear sitting in a classroom with a hat on, realistic, 4k image, high detail"





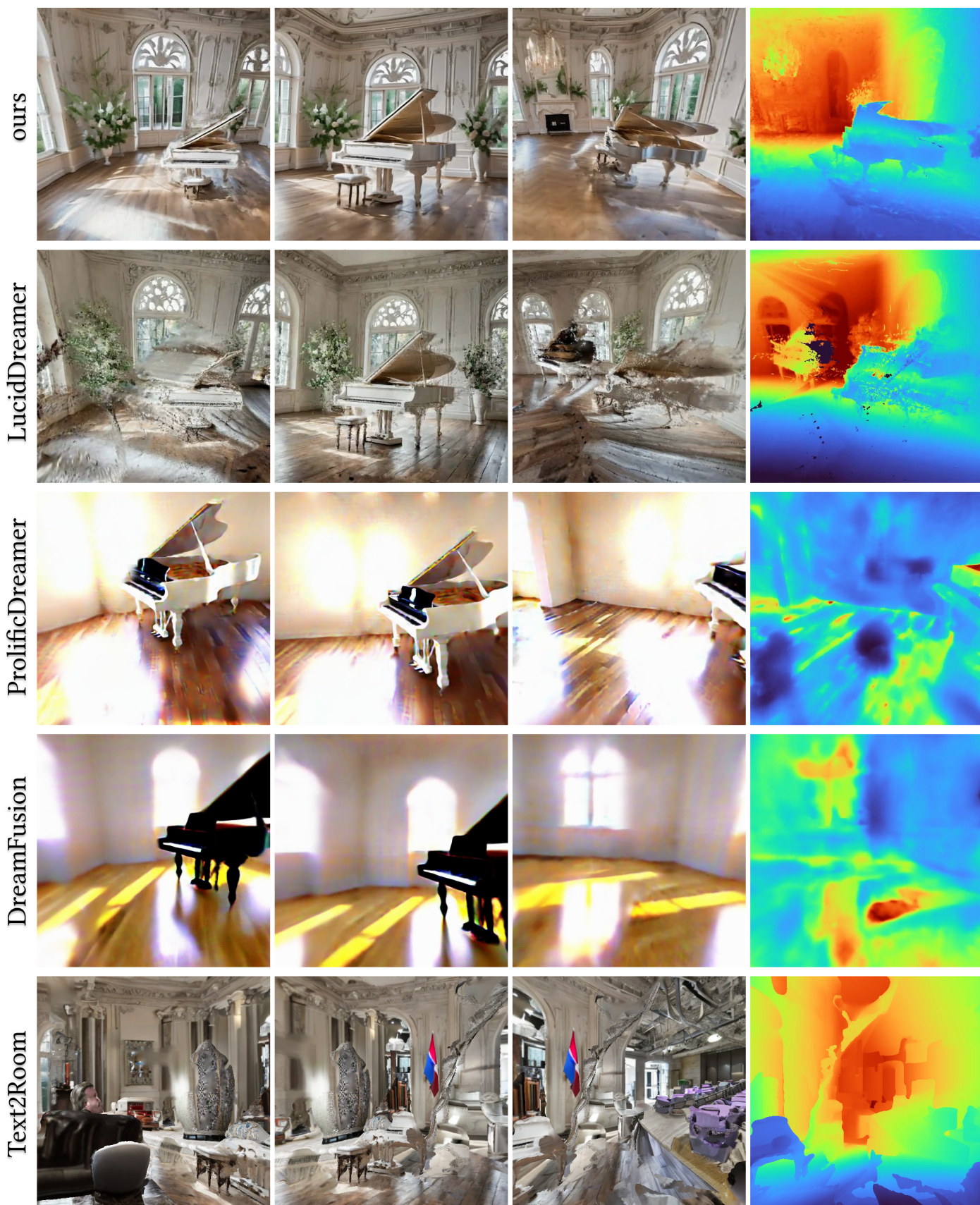
Prompt: "An old car overgrown by vines and weeds, high quality image, photorealistic, 4k image"





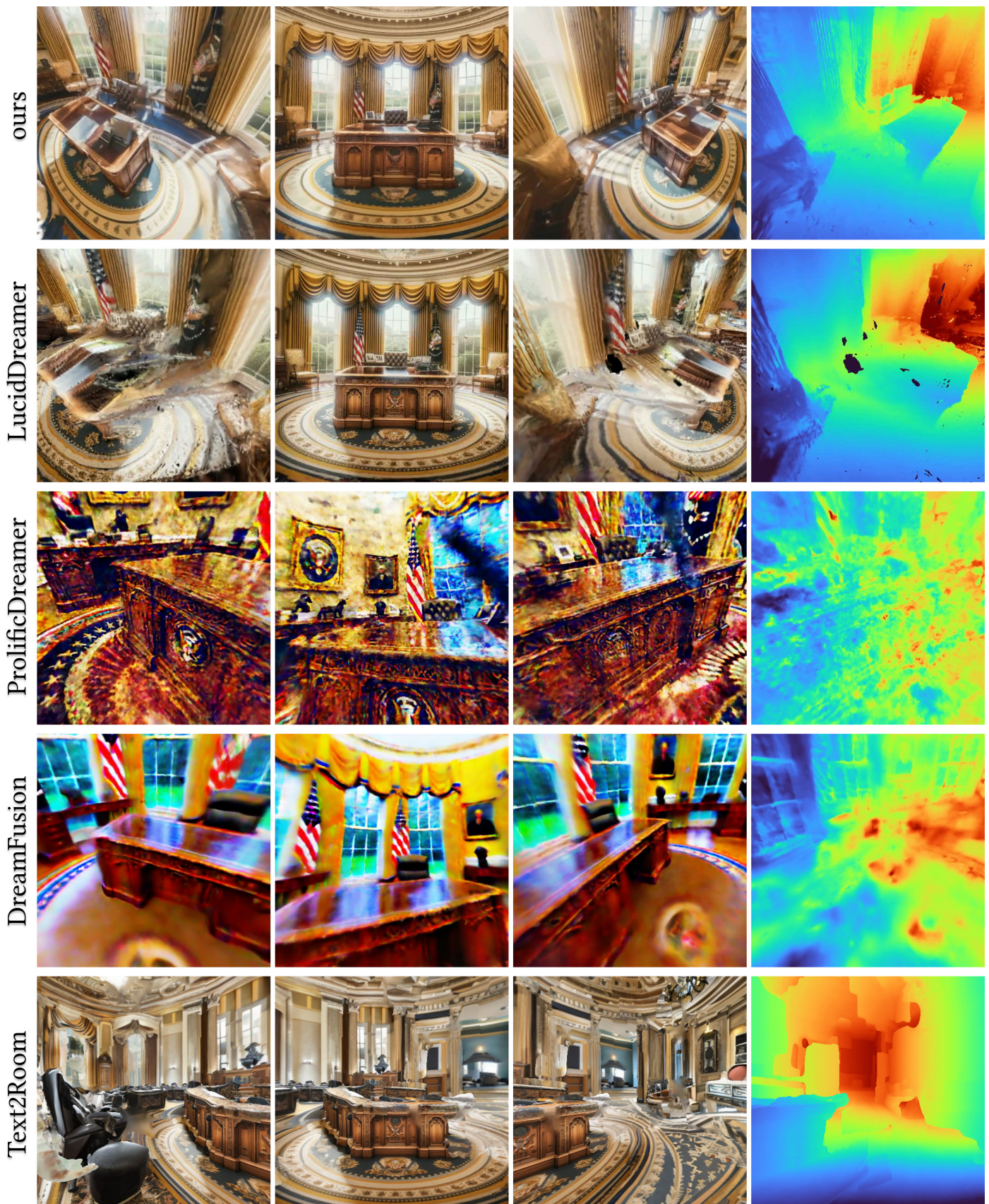
Prompt: "Small lavender room, soft lighting, unreal engine render, voxels."





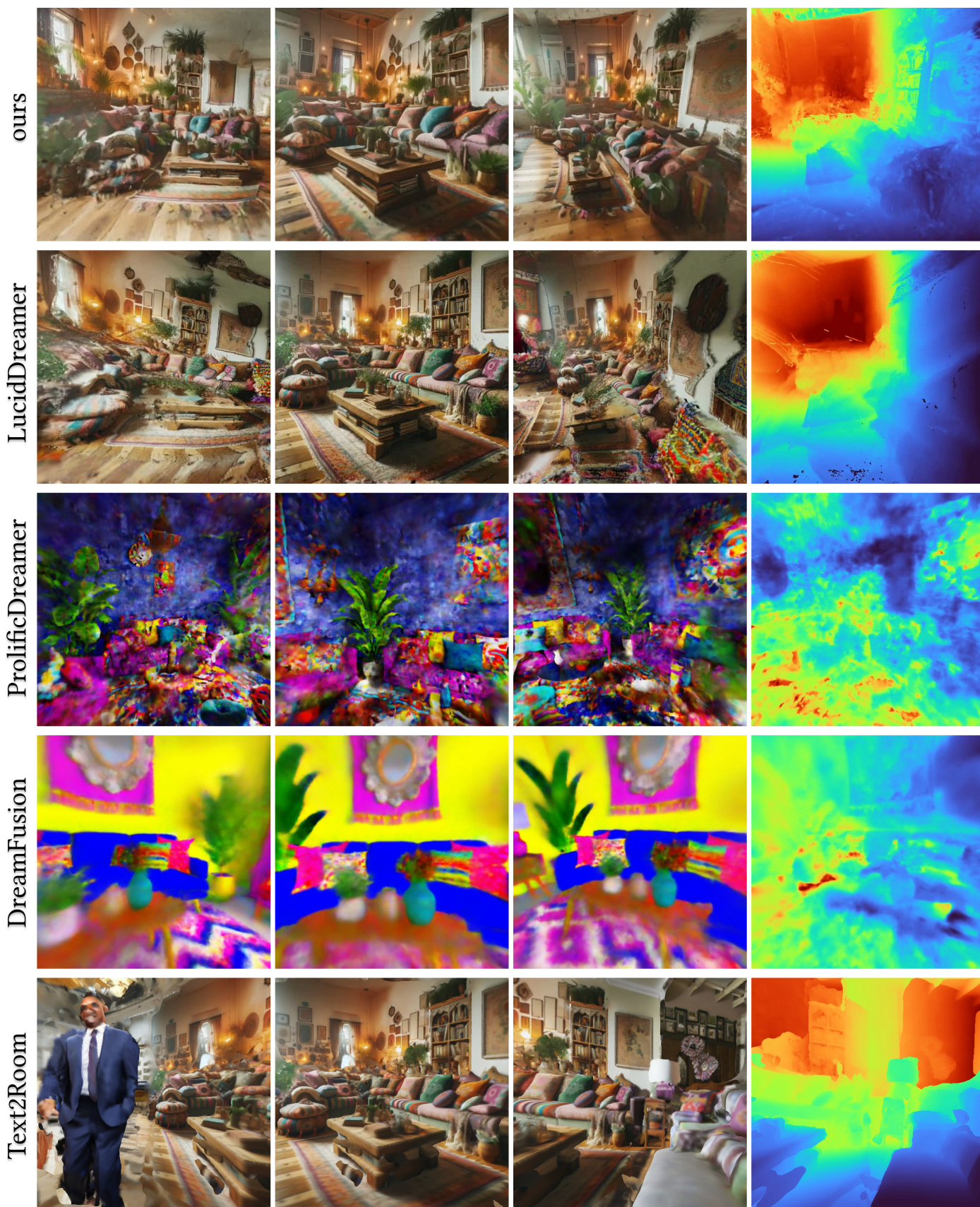
Prompt: "White grand piano on wooden floors in an empty hall, 4k image"





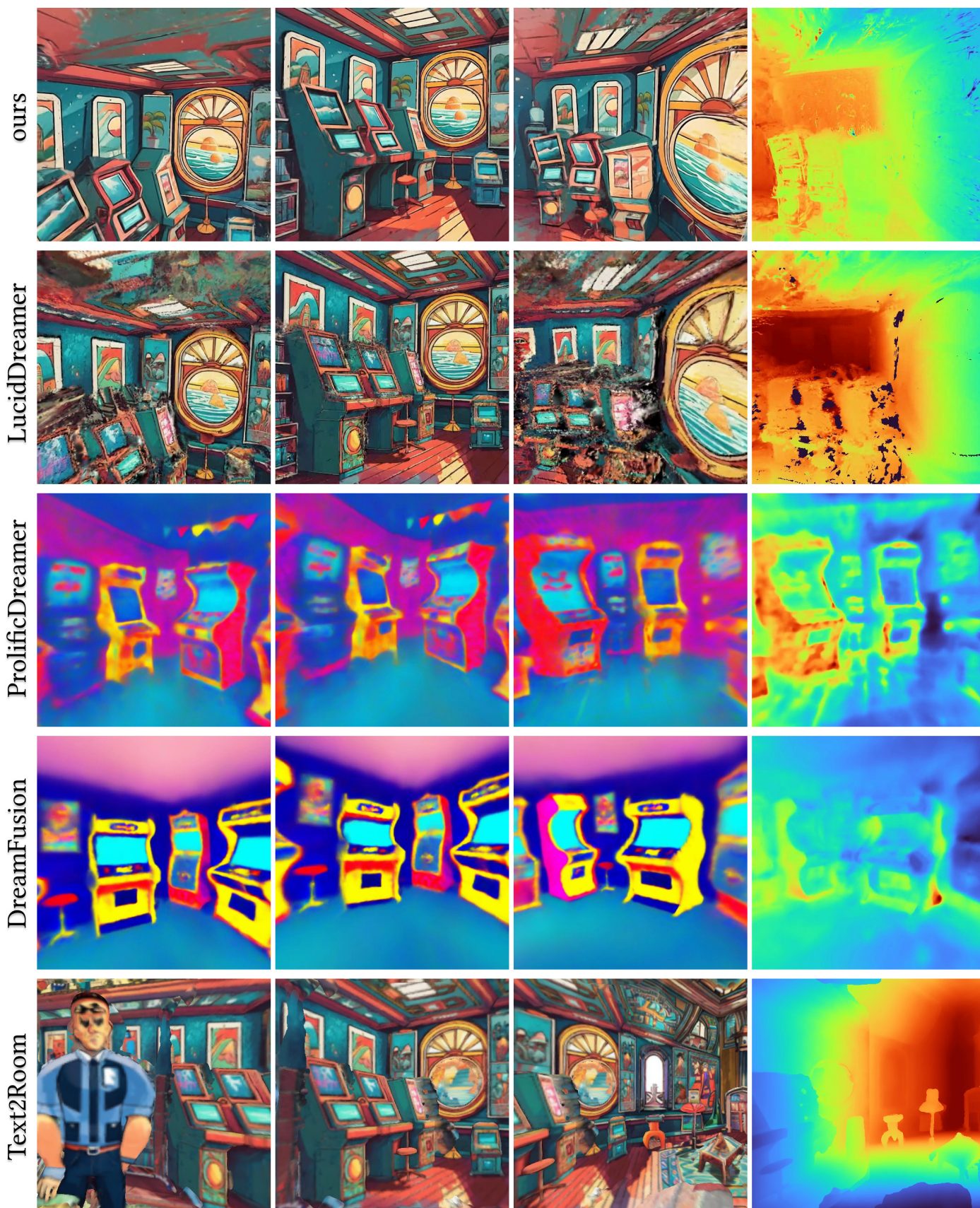
Prompt: "A highly detailed image of the resolute desk in the oval office, 4k image"





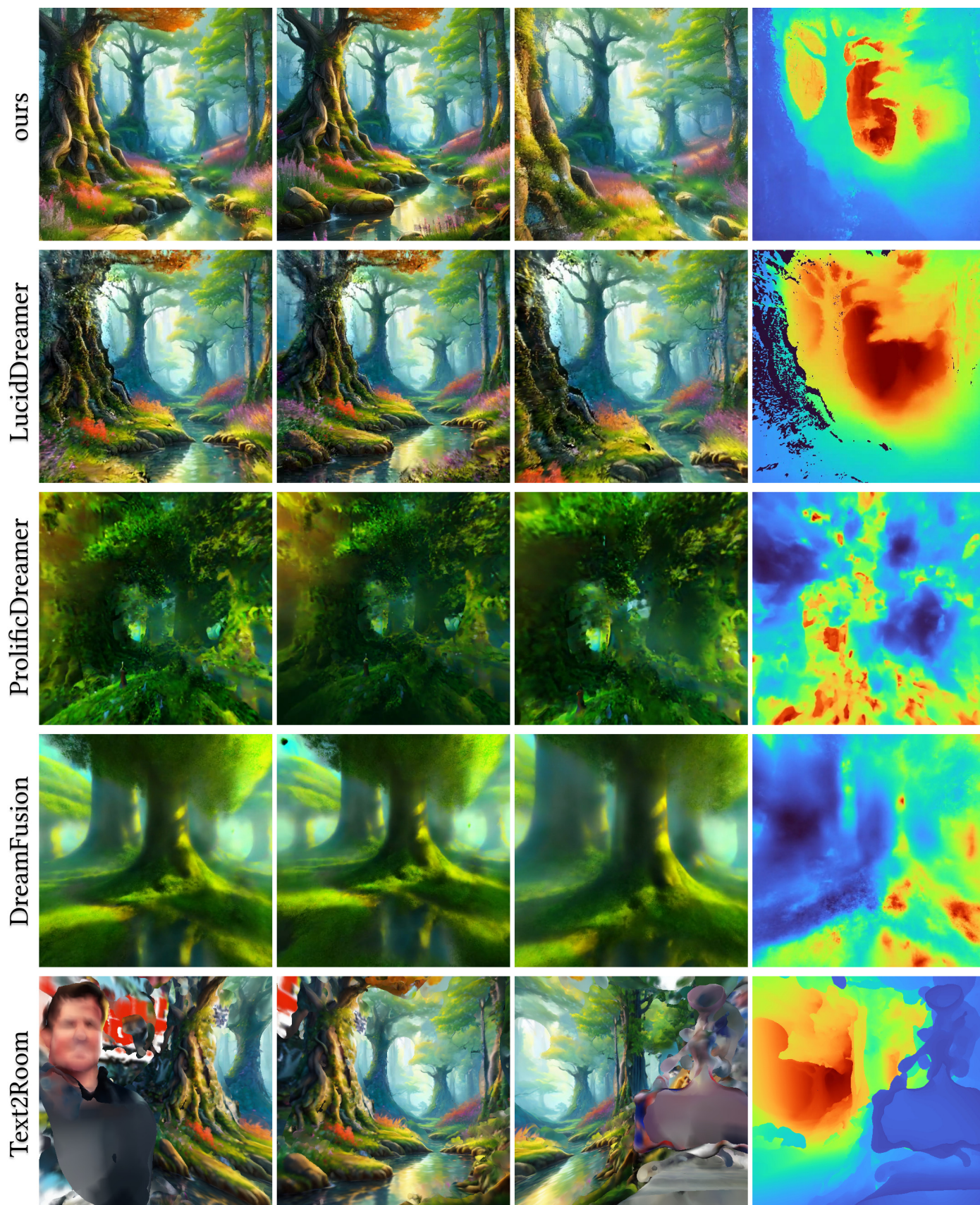
Prompt: "A bohemian living room, colorful textiles, vibrant, eclectic, 4k image, photorealistic"





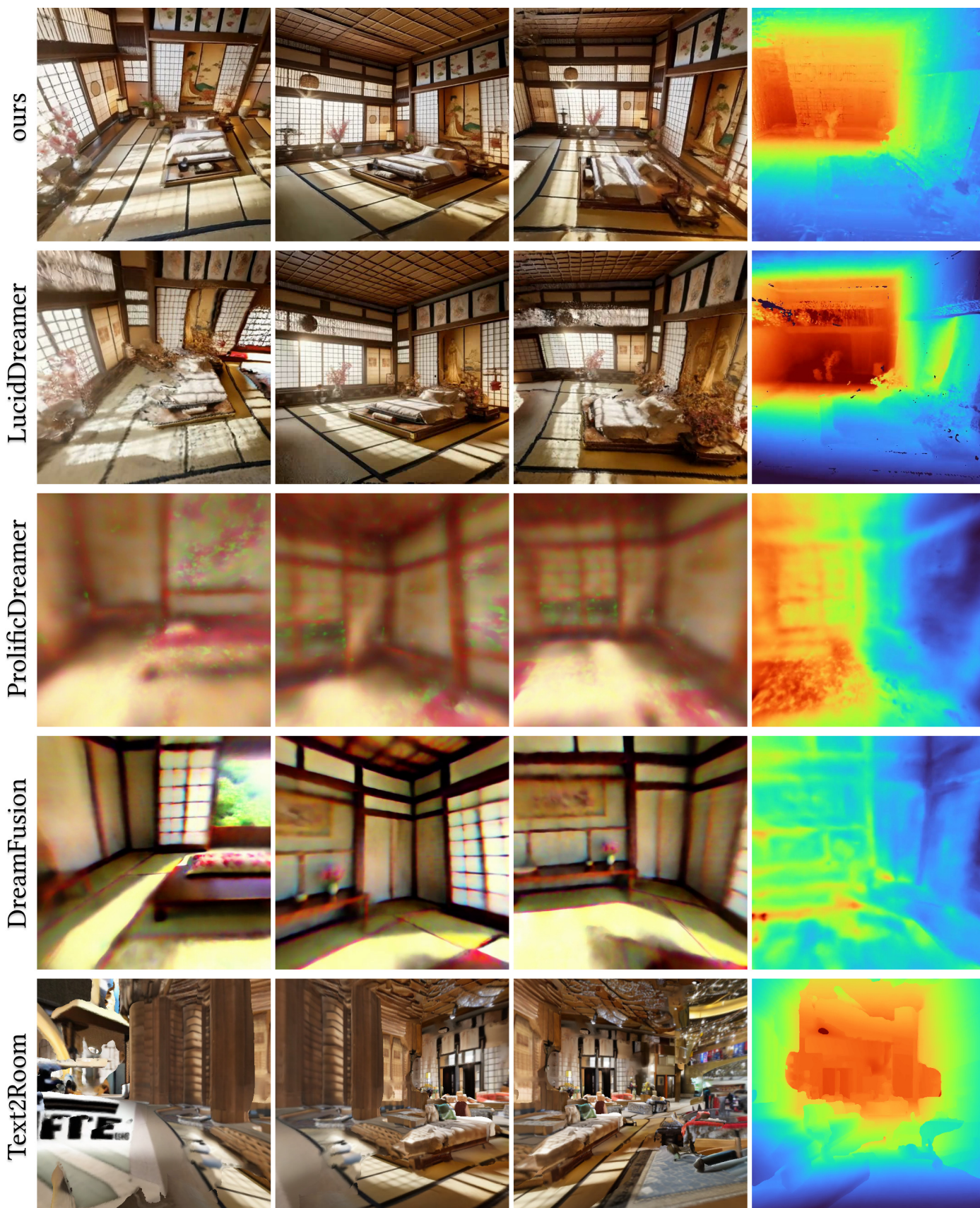
Prompt: "Retro arcade room with posters on the walls, retro art style, illustration"





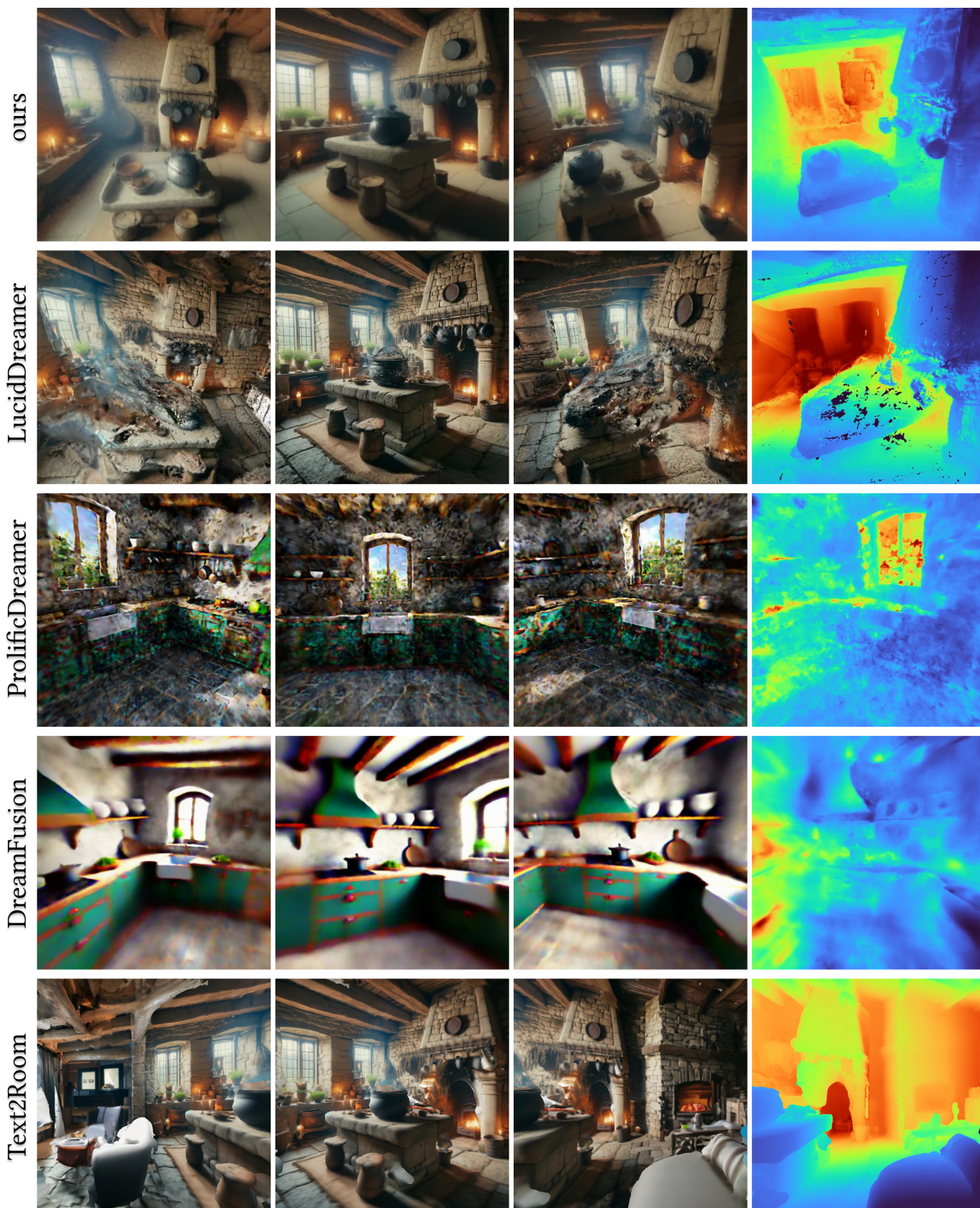
Prompt: "A thick elven forest, fantasy art, landscape, picturesque, 4k image"





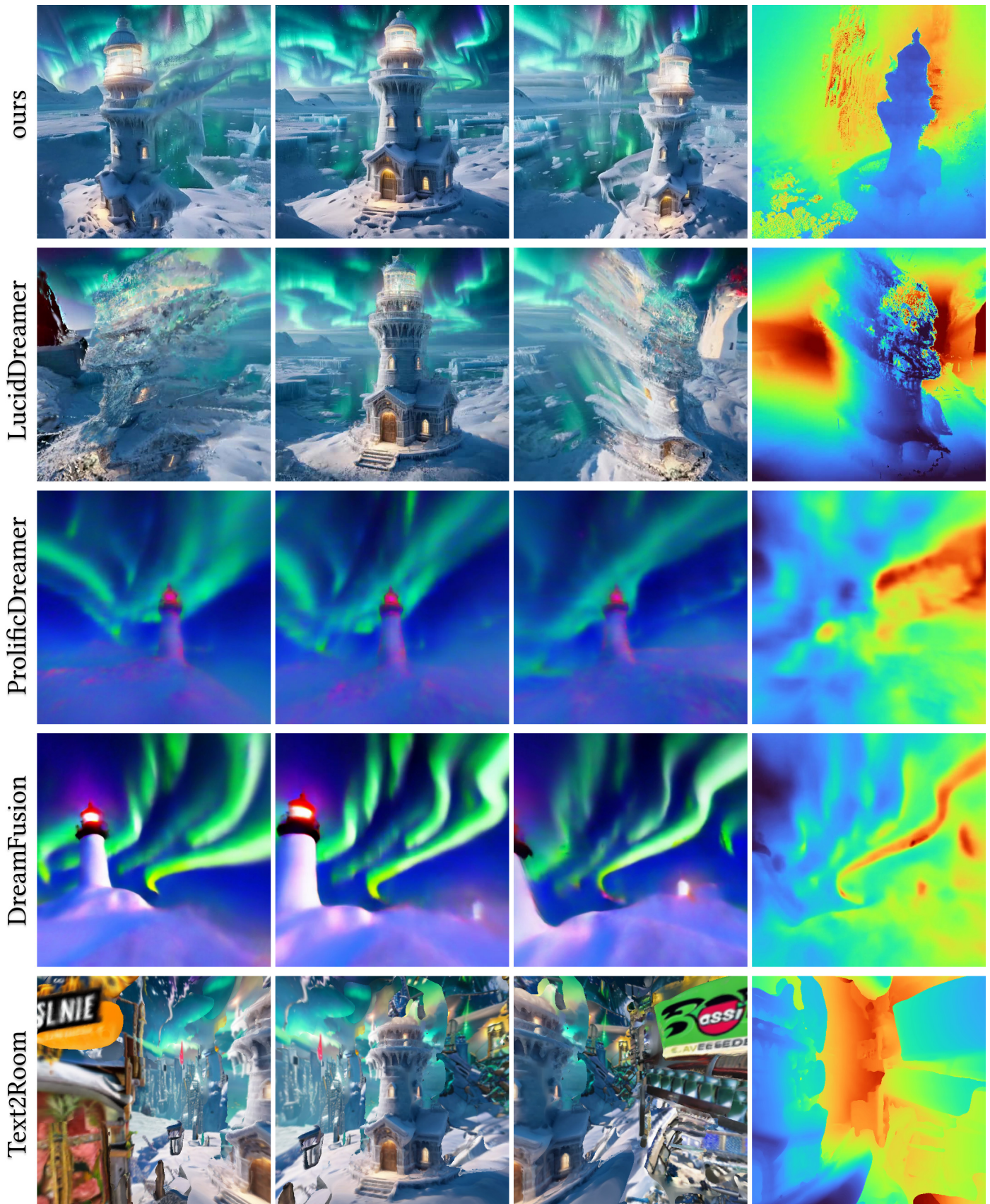
Prompt: "A sunny royal traditional Japanese bedroom, 4k image, ornate, high detail"





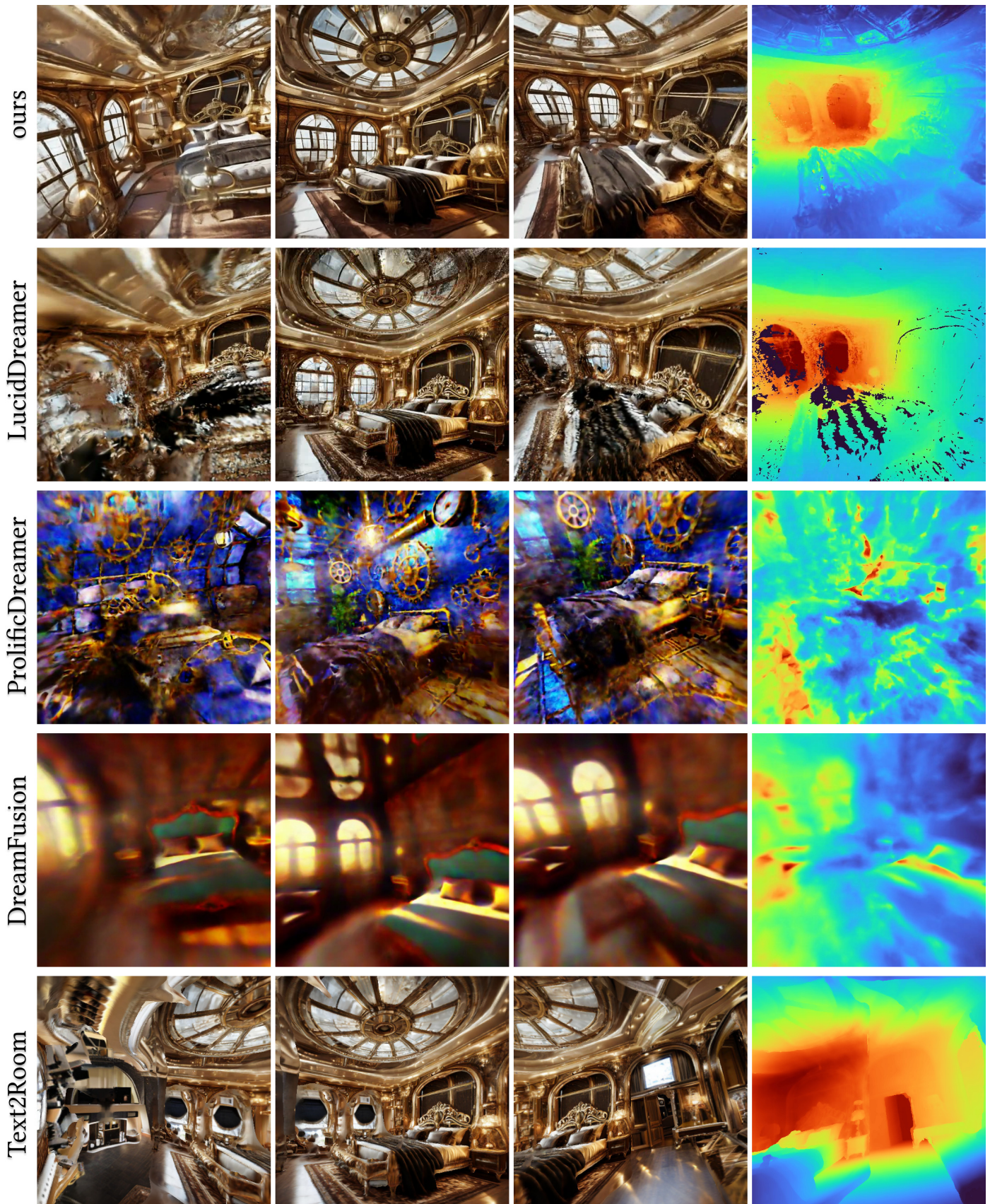
Prompt: "An old charming stone kitchen, 4k image, photo-realistic, high detail"





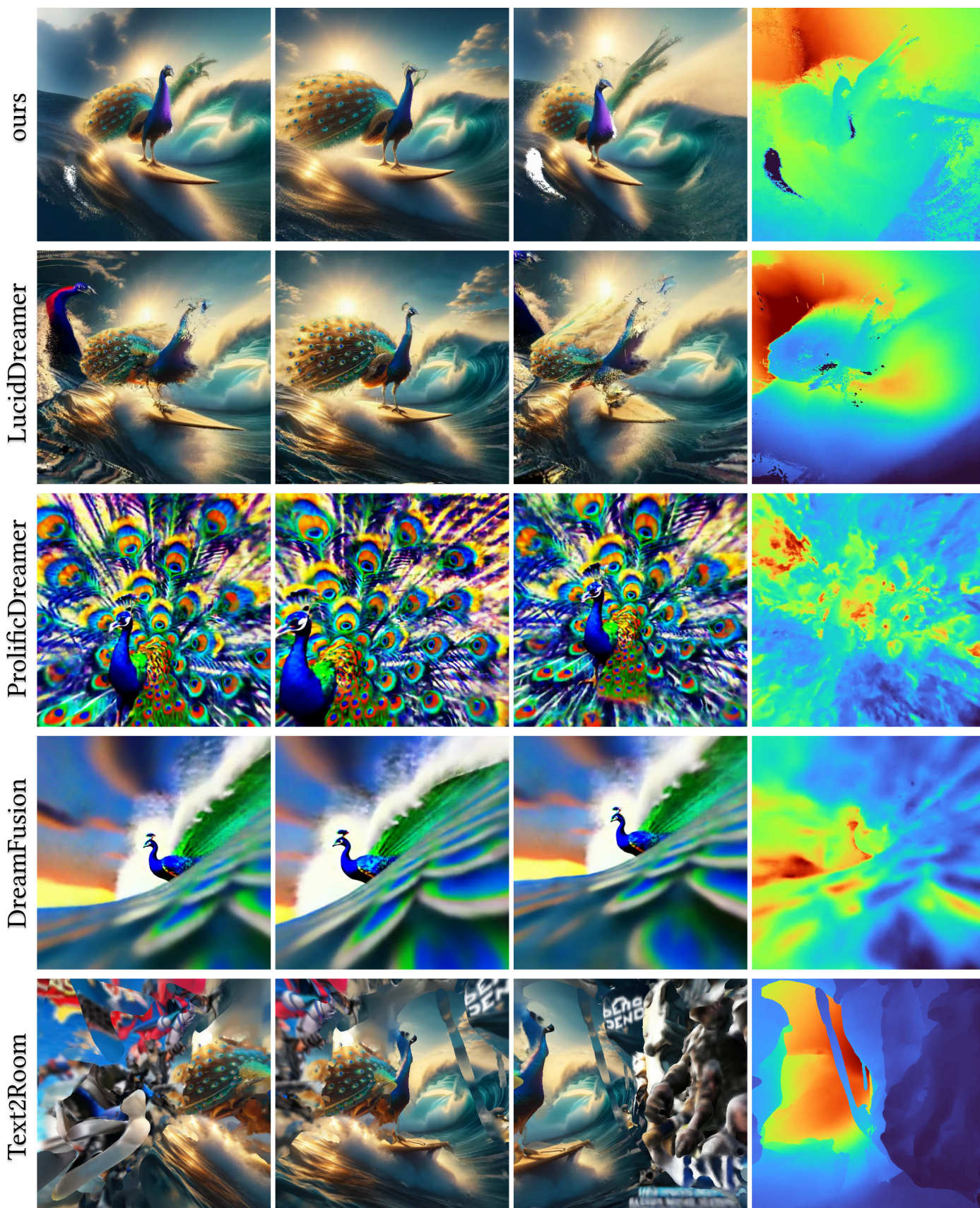
Prompt: "Fantasy lighthouse in the Arctic, surrounded by a world of ice and snow, shining with a mystical light under the aurora borealis, 4k, sharp"





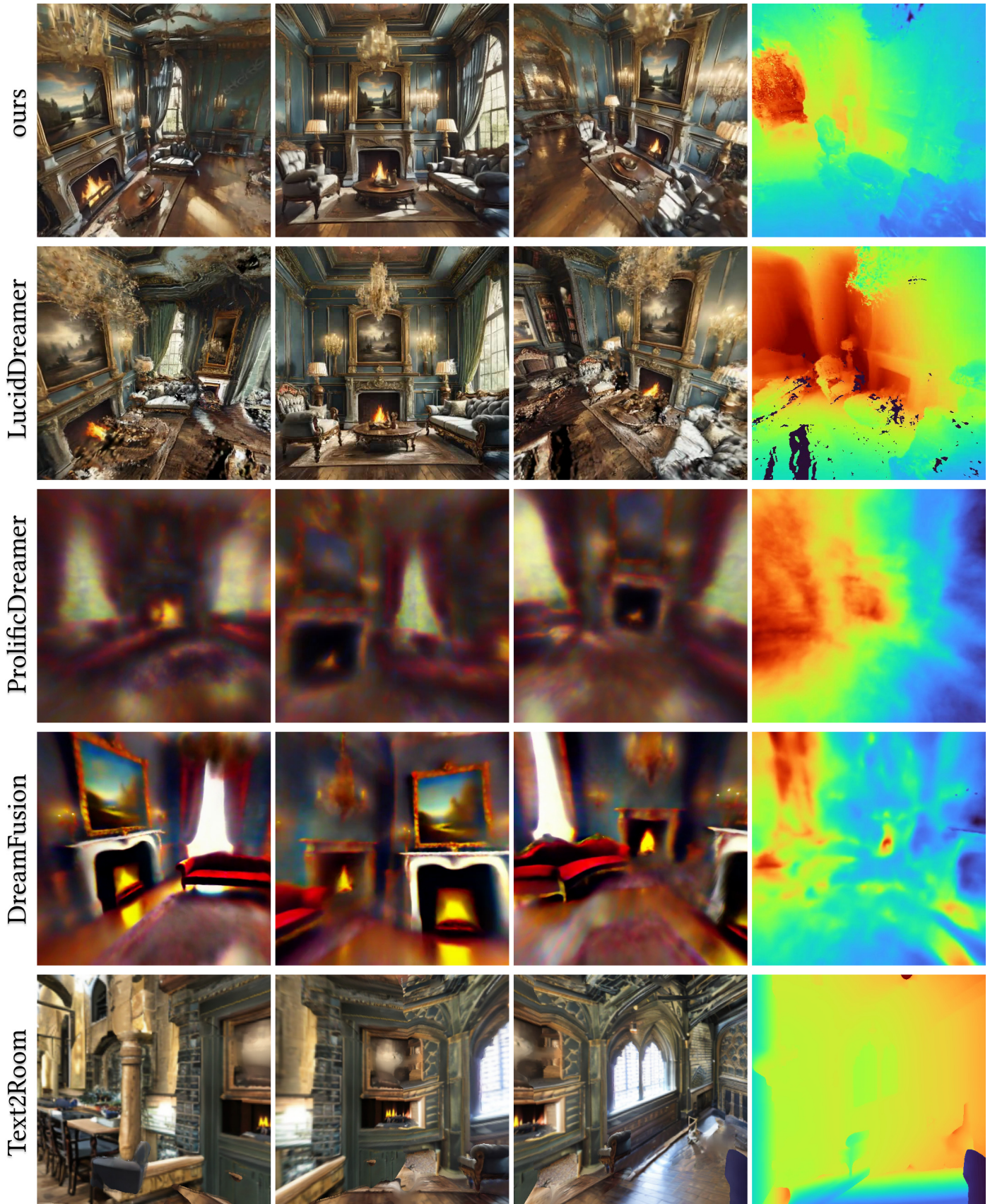
Prompt: "A steampunk bedroom with glass ceilings, photo-realistic, 4k image, bright lighting"





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"



## References

- [1] Mohammadreza Armandpour, Huangjie Zheng, Ali Sadeghian, Amir Sadeghian, and Mingyuan Zhou. 2023. Re-imagine the Negative Prompt Algorithm: Transform 2D Diffusion into 3D, alleviate Janus problem and Beyond. *arXiv preprint arXiv:2304.04968* (2023).
- [2] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. 2022. IronDepth: Iterative Refinement of Single-View Depth using Surface Normal and its Uncertainty. In *British Machine Vision Conference (BMVC)*.
- [3] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. 2023. Improving image generation with better captions. *Computer Science*. <https://cdn.openai.com/papers/dall-e-3.pdf> 2, 3 (2023), 8.
- [4] Shariq Farooq Bhat, Reiner Birkel, Diana Wofk, Peter Wonka, and Matthias Müller. 2023. ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth. (2023). <https://doi.org/10.48550/ARXIV.2302.12288>
- [5] Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. 2023. Lucid-dreamer: Domain-free generation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2311.13384* (2023).
- [6] 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. 2023. threestudio: A unified framework for 3D content generation. <https://github.com/threestudio-project/threestudio>.
- [7] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. 2022. Prompt-to-Prompt Image Editing with Cross Attention Control. *arXiv preprint arXiv:2208.01626* (2022).
- [8] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. 2023. Text2Room: Extracting Textured 3D Meshes from 2D Text-to-Image Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 7909–7920.
- [9] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. 2023. Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation. *arXiv:2312.02145 [cs.CV]*
- [10] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. 2023. LucidDreamer: Towards High-Fidelity Text-to-3D Generation via Interval Score Matching. *arXiv:2311.11284 [cs.CV]*
- [11] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952* (2023).
- [12] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2022. DreamFusion: Text-to-3D using 2D Diffusion. *arXiv* (2022).
- [13] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501* (2020).
- [14] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.
- [15] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. 2023. 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*. 22500–22510.
- [16] Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* (2020).
- [17] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. 2023. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [18] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. 2024. Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data. In *CVPR*.