Supplementary Material for "RichDreamer: A Generalizable Normal-Depth Diffusion Model for Detail Richness in Text-to-3D"

Lingteng Qiu[1,3*†] Guanying Chen[2,1*] Xiaodong Gu[3*]
Qi Zuo[3] Mutian Xu[1] Yushuang Wu[2,1] Weihao Yuan[3] Zilong Dong[3]
Liefeng Bo[3], Xiaoguang Han[1,2‡]

[1]SSE, CUHKSZ [2]FNii, CUHKSZ [3]Alibaba Group

Contents

1. More Details for Normal-Depth Diffusion

1.1. Pre-training on the LAION Dataset

Training of VAE We initialize the parameters of our model with the pre-trained weights of SD 2.1. Specifically, we modify the input channels of the VAE from 3 to 4, and the weights of the newly added channel are initialized as the average of the original weights. We do not modify the number of channels in the latent space of our VAE, which remains at 4 channels.

For VAE fine-tuning, we randomly sample the training data from LAION-Aesthetics V1 [7], selecting data with aesthetics scores larger than 8.0, considering the need for high-quality training data. The training dataset consists of 8 million training samples.

We apply center cropping on the training images and resize them to 384×384. We employ random rotation and flipping of the images to augment our training dataset. Subsequently, we use augmented images as inputs to the monocular prior models (NormalBae [1] and Midas-3.1 [5]) to obtain corresponding estimates of the normal and depth. Notably, we perform depth normalization on the predicted depth values, scaling them from -1 to 1. This normalization step is necessary since the original depth predictions are in the form of real values. These estimated images are then resized to a resolution of 256×256 and serve as the input for the VAE.

During the training phase, following the latent diffusion model [6], we employ the Adam optimizer with a learning rate of 5e-5 to optimize our VAE model. To ensure a well-behaved latent space, we incorporate KL regularization loss. Moreover, to further improve the quality of the generated images, we train an auxiliary discriminator on the output of the VAE. The weights of the KL regulariza-

* Equal contribution.
† Work done during internship at Alibaba.
‡ Corresponding author: hanxiaoguang@cuhk.edu.cn.

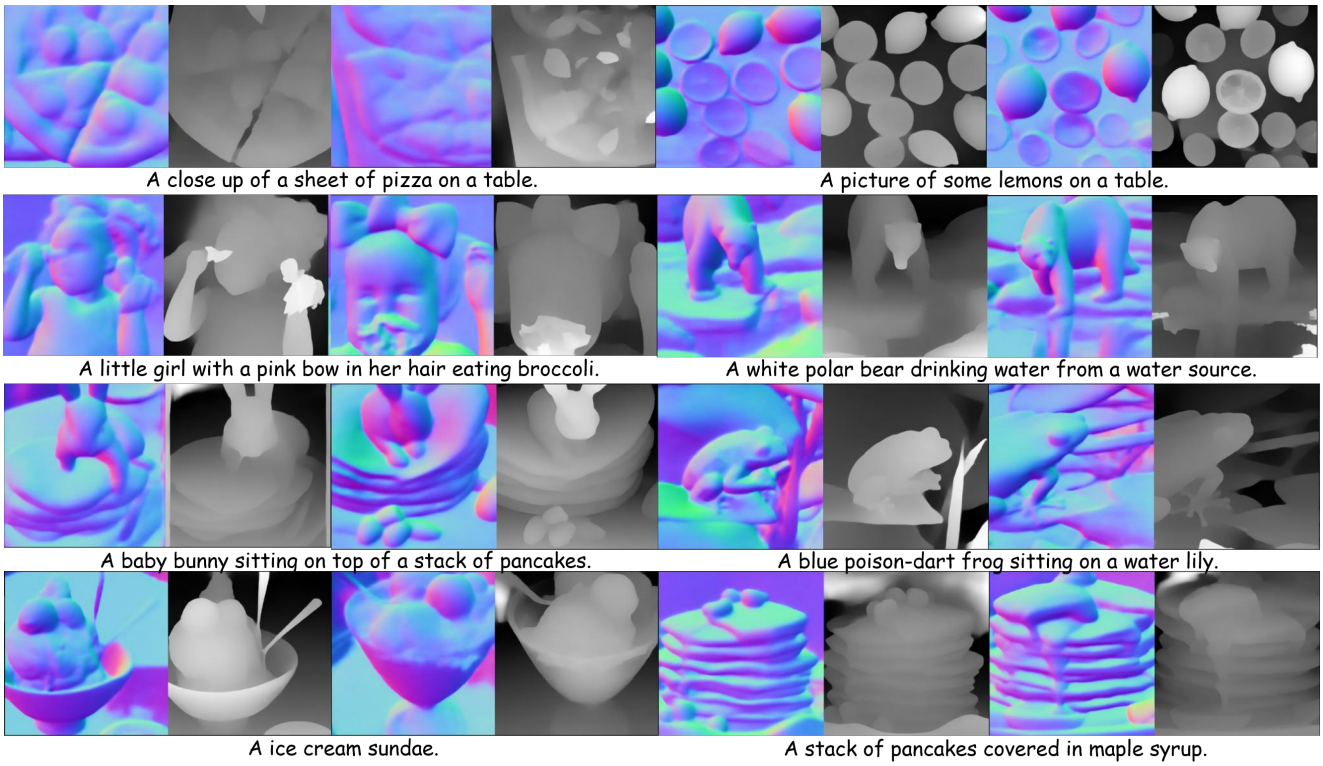


Figure S1. Text to Normal-Depth sampling results.

tion and discriminator are set to 1e-6 and 0.5, respectively. For each iteration, the batch size is set to 1024. The training process is conducted on 8 A100-80G GPUs for two weeks, reaching a total of 100K iterations.

Training of UNet Diffuser Notably, we maintain the original structure of the UNet model as the channels of the latent remain unchanged. During the training phase, we randomly sample data from the Laion-2B-en dataset. We perform a center crop on the sampled images and resize them to a resolution of 512 pixels. Subsequently, these images are passed through the monocular prior models and the encoder of the trained VAE. The output of this process serves as the input of the UNet model.

We follow the strategies utilized in the latent-diffusion model [6] to train our Normal-Depth diffusion model. Specifically, we first train our Normal-Depth diffusion model using the entire Laion-2B-en dataset. After 121,000 iterations of training, we proceed to sample our data from the "Laion-Aesthetics v2 5+" subset. This subset contains images with aesthetics scores greater than 5, and we only select images with an unsafety probability higher than 0.1. The fine-tuning process continues for about 167,000 iterations at a resolution of 512×512. To enable classifier-free guidance sampling, we incorporate a 10% dropping of the text-conditioning. We utilize the Adam optimizer with a learning rate of 1e-4 to optimize our Normal-Depth diffusion model. For each iteration, the batch size is set to 1024. This process costs 11,520 GPU hours.

Text to Normal-Depth Sampling Figure S1 shows the sampling results of our Normal-Depth diffusion model trained on the Laion-2B datasets. As shown in the figure, the sampled normal and depth are not only highly consistent with the textual description but also with high quality. We set the classifier-free guidance scale to 7.5, with 50 DDIM [9] steps.

1.2. Fine-tuning on Synthetic Dataset

Multi-view Normal-Depth Diffusion Fine-tune Thanks to the open-source large-scale Objaverse dataset [2], we fine-tune our Normal-Depth diffusion model on Objaverse to improve its performance for 3D generation tasks. To avoid the Janus problem, following MVDream [8], we fine-tune the Normal-Depth Diffusion model pretrained using multi-view diffusion. Particularly, we use 4 images of orthogonal camera views as the input for the diffusion model and apply a two-layer MLP to embed the extrinsic camera matrix, which is added as a residual to the time embedding.

Figure S2 illustrates the sampling results of our multi-view Normal-Depth Diffusion model, where the classifier-free guidance scale is set to 10 and the negative prompt is

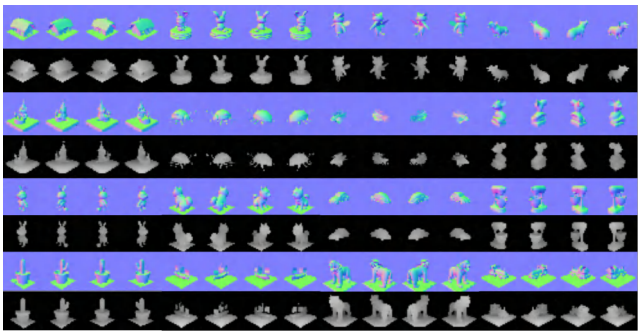


Figure S2. Sampling results of our Normal-Depth diffusion model fine-tuned on the Objaverse dataset.

set to null.

Depth Normalization For a synthetic dataset, we can directly obtain its depth values. Since our Normal-Depth Diffusion process involves normalized depth during training, it is necessary to normalize the range of synthetic depth. To better normalize the depth values, we can introduce a near plane and a far plane to restrict the depth range to [-1, 1]. By defining these planes, we can map the actual depth values to the normalized range. Considering our synthetic data is confined within a 0.5 uint cubic volume, we define the distance from the object's center point to the near plane and far plane as $0.5\sqrt{3}$. After defining the near and far planes, we can normalize the depth values of our synthetic data.

However, the process of normalizing depth is not trivial. Midas [4] estimates depth in the form of disparity, which is a relative measure of the difference in pixel coordinates and is inversely related to depth.

One straightforward approach is to normalize the disparity of synthetic data directly. For simplicity, we assume that the synthetic object is located at the coordinate origin. The equation for normalizing the disparity is as follows:

$$\begin{aligned} \text{Disp}(z) &= \frac{\frac{1}{d_{\text{cam}}-z} - \frac{1}{d_{\text{cam}}+\sqrt{3}l}}{\frac{1}{d_{\text{cam}}-\sqrt{3}l} - \frac{1}{d_{\text{cam}}+\sqrt{3}l}} \\ &= \frac{(\sqrt{3}l+z)\cdot(d_{\text{cam}}-\sqrt{3}l)}{2\sqrt{3}l\cdot(d_{\text{cam}}-z)}. \end{aligned} \tag{1}$$

where z represents the depth value between the given point and the original plane, d_{cam} denotes the depth value from the origin plane to the camera, and l represents the size of the cube that confines the object, as illustrated in Fig. S3.

From the equation, it is evident that the normalized disparity lacks the desirable property of scale invariance. Consider an example: for a fixed camera distance (d_{cam}), if we double the values of l and z, the resulting values will differ from the original l and z values. Similarly, for the same values of l and z, different camera distances will yield different normalized disparities.

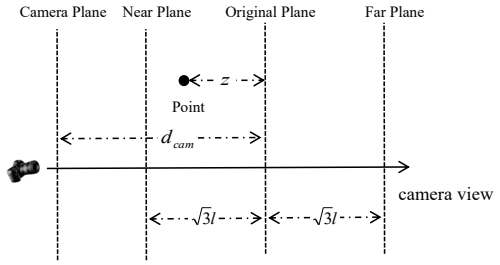


Figure S3. Depth normalization figure.

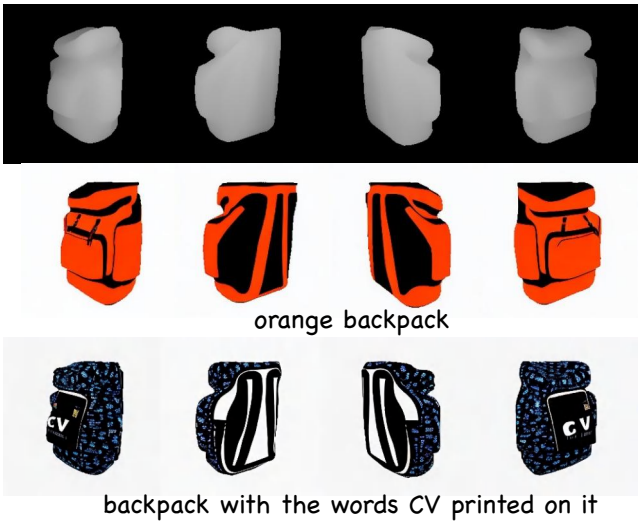


Figure S4. Depth-Condition Albedo diffusion Model.

During the optimization process of text to 3D, we randomly sample different camera distances. This randomness further exacerbates the lack of scale invariance, introducing significant noise into the optimization process. As a consequence, achieving accurate results becomes more challenging due to the varying scales, which can adversely affect the convergence and stability of the optimization algorithm.

To address the aforementioned issue, we propose the use of *reverse depth* as an alternative to disparity normalization. The reverse depth is defined as follows:

$$\begin{aligned} \text{RevDepth}(z) &= \frac{(d_{\text{cam}}+\sqrt{3}l)-(d_{\text{cam}}-z)}{(d_{\text{cam}}+\sqrt{3}l)-(d_{\text{cam}}-\sqrt{3}l)} \\ &= \frac{\sqrt{3}l+z}{2\sqrt{3}l}. \end{aligned} \tag{2}$$

From the equation, it is obvious that the normalization value is independent of d_{cam} and remains unchanged when the variables l and z scale proportionally. For the sake of simplicity, in the following content, the depth refers to normalized reverse depth.

2. More Details for Albedo Diffusion Model

Depth-Conditioned Albedo Diffusion Fine-tuning Regarding the training of the albedo model under the depth condition, we resize the normalized depth image and concatenate it with the latent space of the VAE. It means the number of channels in the UNet's input expands from 4 to 5. We initialize the parameters of the Albedo-Diffusion Model with the pre-trained weights of SD 2.1. For the additional dimension in the input channel of the UNet, we set its weights to zero values. To alleviate the multi-face problem in the generated texture, we also employ the same multi-view strategy used in the multi-view Normal-Depth Diffusion model to train our albedo diffusion model (see Fig. S4 for sampling results).

3. More Details for Geometry Generation

Our Normal-Depth diffusion model can be applied to optimize DMTet and NeRF. To better verify the effectiveness of our Normal-Depth Diffusion model, we design two model variants for text-to-3D. The first model, denoted as *Ours (Sphere)*, initializes the DMTet with a Sphere for geometry generation. The second model, denoted as *Ours (NeRF)*, first optimizes a NeRF with our Normal-Depth diffusion model and then converts the NeRF to DMTet as an initialization for geometry generation.

In the paper, the geometry generation loss function is defined as:

$$\mathcal{L}_{\text{Geo}} = \lambda_{\text{SD}}\mathcal{L}^{\text{SD}}_{\text{SDS}-\text{Normal}} + \lambda_{\text{ND}}\mathcal{L}^{\text{ND}}_{\text{SDS}-\text{ND}}, \tag{3}$$

More Details for *Ours (Sphere)* The geometry optimization in *Ours (Sphere)* consists of two stages: a coarse shape optimization stage and a refinement stage. For the coarse shape optimization stage, we follow the approach of Fantasia3D, where we directly resize the rendered normal and depth maps as the latent space features to quickly optimize to obtain a coarse shape. In the refinement stage, we utilize the latent features obtained from the VAE to enhance the details of geometry.

Specifically, our DMTet is initialized from a sphere. In terms of coarse shape optimization, we set λ_{SD} and λ_{ND} to 0.5 and 1.0, respectively. When it comes to refinement optimization, both λ_{ND} and λ_{SD} are set to 1.0. The negative prompt "low quality" is used in both diffusion models. The classifier-guided scale is set to 100 and 50 in SD 2.1 and the Normal-Depth diffusion model, respectively. For the time sampling schedule, we adopt a uniform sampling strategy of annealing from [0.5, 0.98] to [0.05, 0.5] when we switch from the coarse to refinement stage.

Ours (Sphere) is optimized on a single Nvidia A100-80G GPU for 3000 iterations (1500 iterations for the coarse stage and 1500 iterations for the fine stage), where we use the AdamW optimizer with learning rates of 1e-3. The batch size is set to 8, and the entire geometry optimization process takes about 40 minutes.

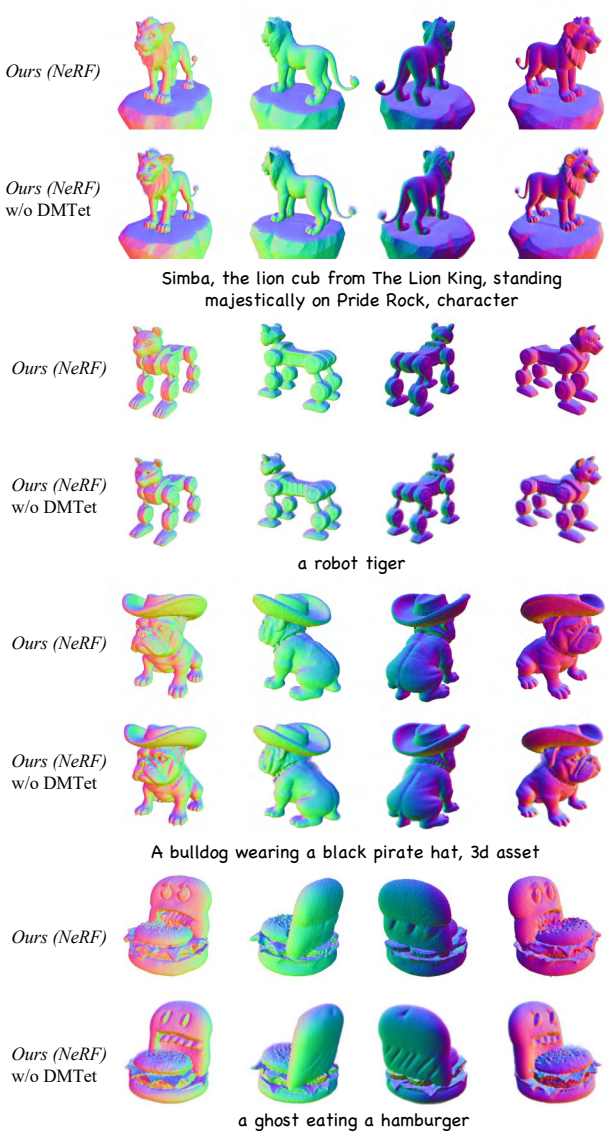


Figure S5. The visual geometric comparison results about with and without DMTet refinement stage in *Ours (NeRF)*.

More Details for *Ours (NeRF)* The geometry optimization in *Ours (NeRF)* also consists of two stages. In the coarse shape optimization stage, we adopt the rendered normal and depth maps as latent space input of SD 2.1 and the Normal-Depth diffusion model without VAE encoding to quickly optimize to obtain the coarse shape. In the fine detail refinement stage, we adopt encoding features obtained from VAE as latent space input of SD 2.1 to get a detailed shape.

Ours (NeRF) is optimized on a single Nvidia A100-80G GPU for 5000 iterations (1500 iterations for the coarse stage and 3500 iterations for the fine stage), where we use the AdamW optimizer with learning rates of 1e-3 except for the hash encoding module using 1e-2. For the time sampling schedule, we adopt a uniform sampling strategy of annealing from [0.5, 0.98] to [0.05, 0.5] at the 3000th iteration. We set λ_{SD} to 1.0 and λ_{ND} annealing 10 to 2 at the 3500th iteration. The classifier-guided scale is set to 50 in both SD 2.1 and the Normal-Depth diffusion model. We utilize a multi-resolution strategy to train NeRF efficiently, the rendering resolution increases from 64×64 to 256×256 at the 3000th iteration, and the batch size decreases from 8 to 4. The entire geometry optimization takes about 40 minutes.

To reduce geometric artifacts when converting NeRF to DMTet representations, we optimize the DMTet with an additional 3000 iterations to refine the geometry. This is done using the same strategy adopted in the fine detail stage, except the rendering resolution is increased to 512. The refinement optimization takes about 20 minutes.

As illustrated in Figure S5, the optimized geometry of NeRF using our Normal-Depth diffusion model is already of very high quality, clearly demonstrating the effectiveness of our Normal-Depth diffusion model in optimizing both the DMTet and NeRF representations. After undergoing DMTet conversion and subsequent refinement, the surface details are further enhanced. However, for geometry types that are more suitably represented by a density volume (e.g., hair and smoke), the geometry shape tends to be better without the DMTet conversion.

Camera Sampling During the training process, we randomly sample the elevation angle between 5 degrees and 30 degrees and uniformly sample the camera distance between 1.5 and 1.9. In addition, for the sampling of azimuth angles, we follow the sampling approach from MVDream [8], where we consecutively sample four orthogonal viewpoints.

4. More Details for Appearance Modeling

The classifier guidance scale is set to 10 for the depth condition Albedo-Diffusion model, while its value is 100 for the original SD 2.1. We harness the same camera sampling strategy in geometry appearance modeling. In terms of the time sampling schedule, we adopt a uniform sampling strategy from [0.02, 0.98].

Our appearance model is optimized on a single Nvidia A100-80G GPU for 3000 iterations. The batch size is set to 8, and the AdamW optimizer with learning rates of 1e-2 is utilized to update the model parameters.

Figure S6. Example of the interactive interface for the user study.

5. More Details for the User Study

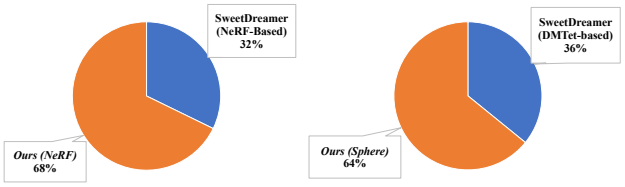
In the user study compared with the baseline, our main focus is to evaluate the quality of the generated geometry and the textured models. Regarding geometry, we primarily compare whether the geometry is complete, if the fine details appear natural, and whether there are significant artifacts. For textured models, our comparison is based on the naturalness of the generated textures and the alignment between the textured model and the textual descriptions. At the beginning of each questionnaire, we will provide an illustrative example to explain what is meant by "visual-textual matching" and how to evaluate the quality of the generated geometry.

The interface of our user study is shown in Fig. S6, where we display the results of different methods in each row and randomly shuffle the order of the methods to avoid introducing bias. For each row, we display four images, which consist of color and normal maps captured from two camera views that are 180 degrees apart.

We distribute our questionnaire to graduate students and professionals working in the field of 3D, such as model designers and engineers from technology companies. All the prompts used in our user study are presented in the attached *txt* file.

6. Ablation Study about Joint Distribution of Normal and Depth

The significance of training the joint distribution of Normal-Depth Diffusion model is substantiated by three additional



Ours (NeRF) vs. NeRF-Based *Ours (Sphere)* vs. DMTet-Based

Figure S7. The user study for the comparison with SweetDreamer.

Figure S8. Comparison between SweetDreamer (NeRF-based) and *Ours (NeRF)*. In each row, from left to right show the rendered image and normal map of the SweetDreamer, followed by the rendered image and normal map of our method.

experiments, namely, Depth-only , Normal-Only and independent Normal-Depth (Normal+Depth). Normal-only and Depth-only mean that we optimize the 3D model separately based on two independent diffusion models for Normal and Depth, respectively. On the other hand, Normal+Depth implies that SDS loss is accumulated from both independent Normal and Depth diffusion models.

Concretely, we train two independent diffusion models for normal and depth as the same setting of our Normal-Depth diffusion model. Figure S10 (a) showcases the text-to-3D results from various diffusion models, where the SDS loss computed from SD is excluded to isolate the impact of the diffusion models. Our observations reveal that while depth diffusion excels at shaping structure, it tends to introduce noise. In contrast, normal diffusion improves surface smoothness but falls short in structuring capability. The last

Figure S9. Comparison between SweetDreamer (DMTet-based) and *Ours (Sphere)*. In each row, from left to right show the rendered image and normal map of the SweetDreamer, followed by the rendered image and normal map of our method.

two columns compare the results of the independent models and the joint diffusion model, underscoring the latter's superior accuracy in geometry generation.

7. Consistency of Normal and Depth

While the initial output of Normal-Bae and Midas exhibits some inconsistency, we observed significant mitigation of this issue after training. To evaluate consistency, we sample 500 captions and images from MSCOCO valuation set (images as the input for prior models, captions for diffusion model). We convert the normal to depth and compute the scale-invariant L1 loss with the generated depth. The results for monocular prior models, Normal-Depth diffusion, and MV-Normal-Depth are $0.0980, 0.0795, 0.0487$, respectively, demonstrating the training largely enhances the consistency of outputs (see Fig. S10 (b) for visual examples).

8. Failure Case and Limitations

Though our method demonstrates the success of generation for many diverse scenarios, it will also sometimes fail especially when the scene is very complex, as shown in Fig. S11. Moreover, for appearance modeling, we currently only consider the constraints for the albedo components. We leave

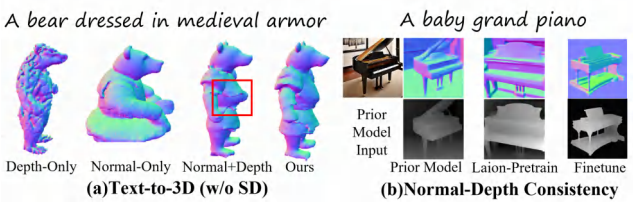


Figure S10. Effects of learning the normal and depth distribution.

Figure S11. Failure cases.

exploring solutions for these two problems as our future work.

9. More Results

9.1. Comparison with SweetDreamer

In comparison to SweetDreamer [3], a concurrent work that does not have publicly available code, we selected 40 text prompts from their project page or paper (20 prompts from their NeRF-based approach and 20 from their DMTet-based approach). Since the DMTet-based method from SweetDreamer visualizes the shape with the *shading normal* instead of the *geometry normal*, it is not feasible to directly compare the geometry quality. Therefore, we focus solely on evaluating the overall quality of the textured models.

Figure S7 illustrates the results of the user study comparing our method with SweetDreamer. In our user study, a significant majority of participants expressed a preference for our method. Specifically, when compared with SweetDreamer's NeRF-based approach, 68% of users selected *Ours (NeRF)* as their preferred choice. When compared with SweetDreamer's DMTet-based approach, 64% of users chose *Ours (Sphere)* as their preferred option. This outcome demonstrates that our method outperforms SweetDreamder in 3D generation.

Figure S8 and Figure S9 present the comparisons with SweetDreamer (NeRF-based) and SweetDreamer (DMTet-based), respectively. We can observe that our method generates better geometry compared to SweetDreamer.

9.2. Comparison with Fantasia3D

we have conducted a comprehensive comparison with Fantasia3D, involving 47 examples in the user study. Figure S13 shows additional comparisons with official Fantasia3D. Besides, Fantasia3D requires case-by-case initialization to achieve good results (*e.g.*, an oriented ellipsoid or a coarse shape), while our method just starts from *a sphere*.

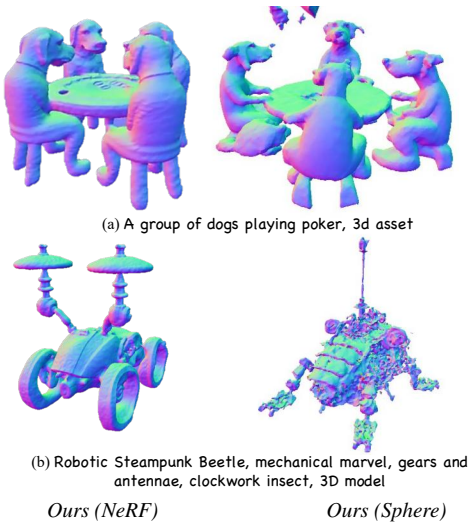


Figure S12. Comparison between *Our (NeRF)* and *Our (Sphere)*.

Figure S13. More comparison results with Fantasia3D.

9.3. Discussion for *Ours (Sphere)* and *Ours (NeRF)*

To better understand the behavior of our method, we discuss differences between the *Ours (Sphere)* and *Ours (NeRF)*. In our experiments, we found that DMTet initialized from NeRF and from a sphere has advantages for different cases.

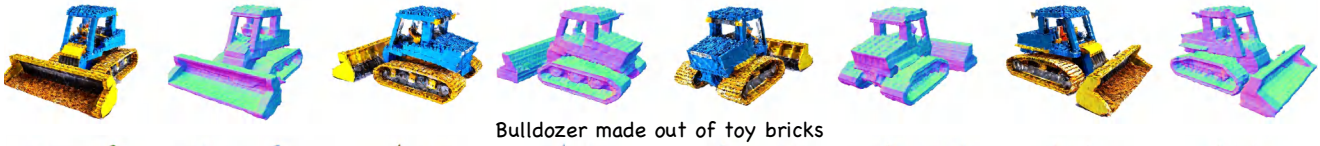
In terms of NeRF initialization, it is easier to generate scenes with multiple objects, e.g., "a group of dogs playing poker". Figure S12 (a) demonstrate the comparison results between *Ours (NeRF)* and *Ours (Sphere)* in this case. However, compared to optimization from a sphere, the geometry from NeRF initialization tends to be smoother, making it difficult to generate surfaces with highly detailed structures, as shown in Figure S12 (b).

In the future, we aim to devise a novel hybrid representation that combines both initialization methods. This approach will allow us to leverage the strengths of each initialization and potentially yield improved results.

9.4. More Visual Results

We present more visual results for *Ours (Sphere)* in Figures S14- S17 and *Ours (NeRF)* in Figures S18-S21.

A crocheted doll wearing a crown, 4K, HD.



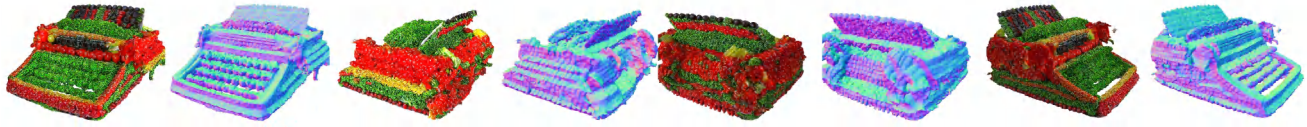
Bulldozer made out of toy bricks

A chihuahua wearing a tutu

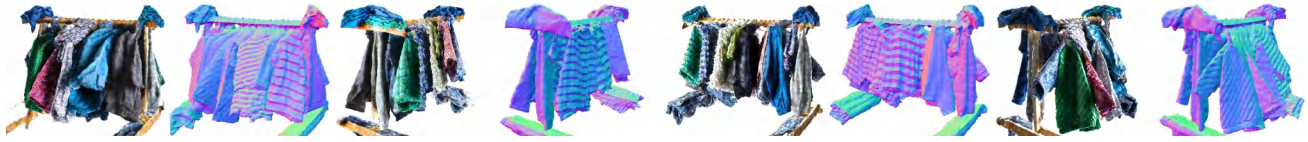
A charming scene of a crocodile chef cooking in a gourmet kitchen. The crocodile is wearing a chef's hat and apron, skillfully preparing exotic dishes with a variety of colorful ingredients spread around, 4K, HD.

A 3d model of an adorable cottage with a thatched roof

A crocodile playing a drum set



A DSLR photo of edible typewriter made out of vegetables



A drying rack covered in clothes

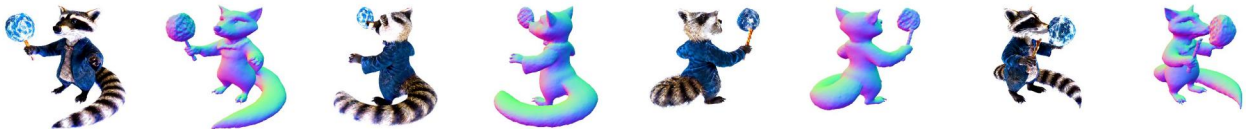
A fox playing the cello

A wide angle DSLR photo of a squirrel in samurai armor wielding a katana

Figure S14. Visual results of *Ours (Sphere)* (Part I).

A DSLR photo of a turtle standing on its hind legs, wearing a top hat

A shiny silver robot cat

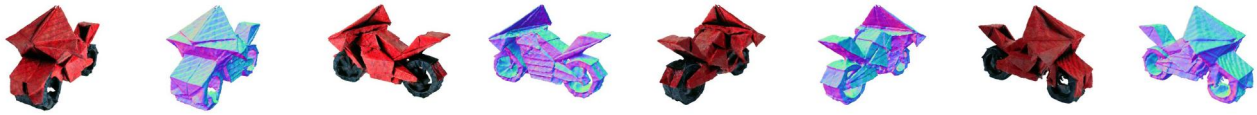


A wizard raccoon casting a spell

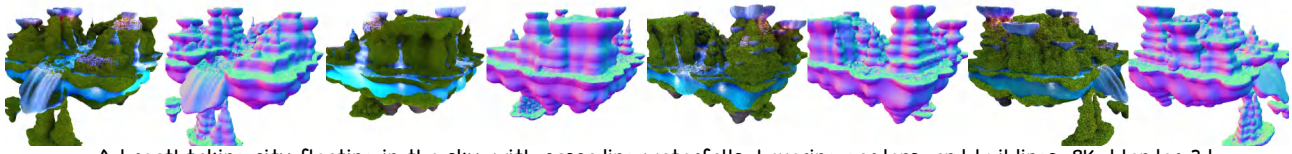
The Statue of Liberty, aerial view

A statue of angel, 3d asset

A sleek, sand-colored dragon that blends into the desert landscape.



A DSLR photo of an origami motorcycle



A breathtaking city floating in the sky, with cascading waterfalls, hovering gardens, and buildings, 8K, blender 3d.

A raccoon stealing a pie

The Joker from Gotham City wearing a dirty hat, with a sinister expression on his face, captured in stunning photorealistic detail, 4K, HD.

Figure S15. Visual results of *Ours (Sphere)* (Part II).

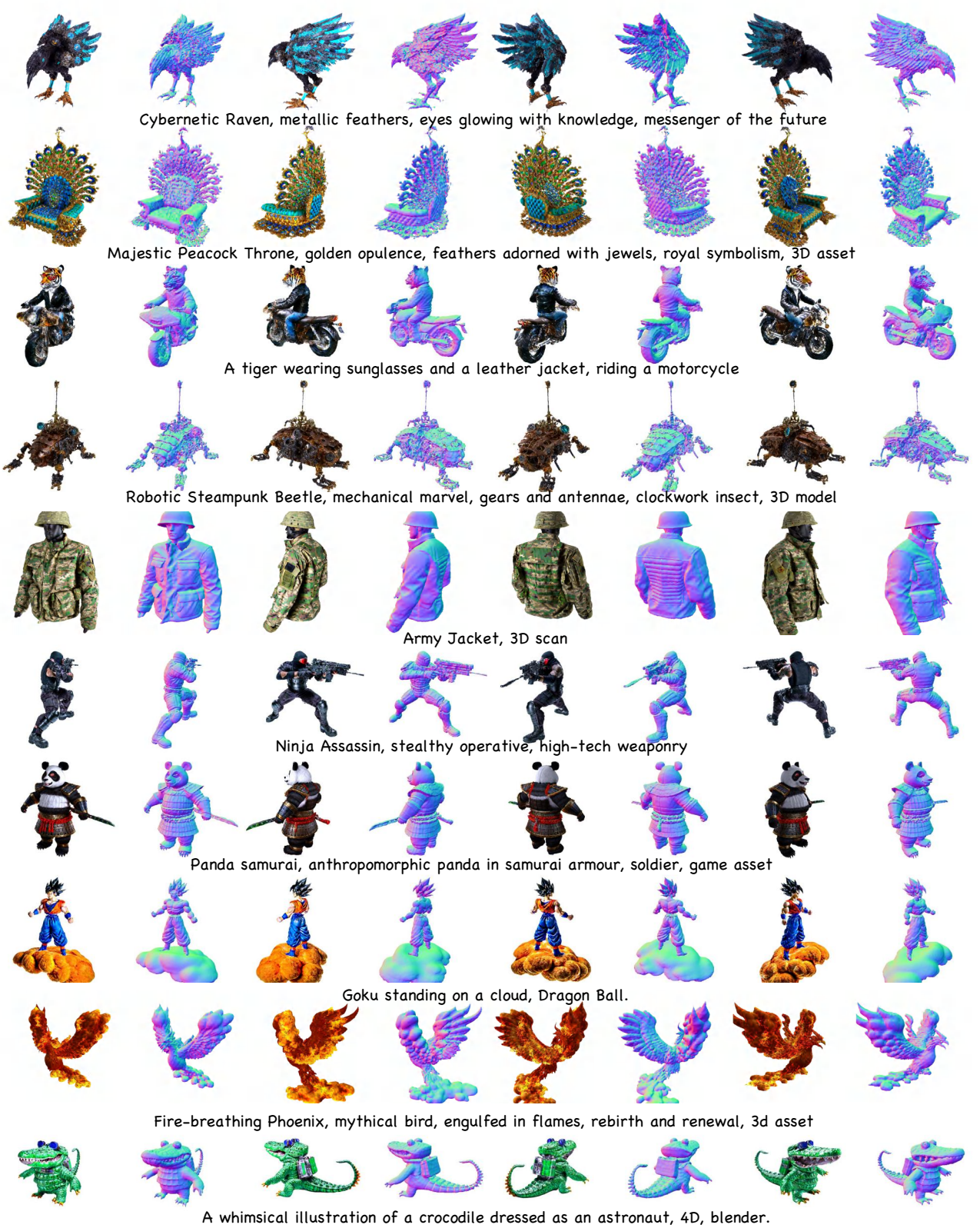


Figure S16. Visual results of *Ours (Sphere)* (Part III).

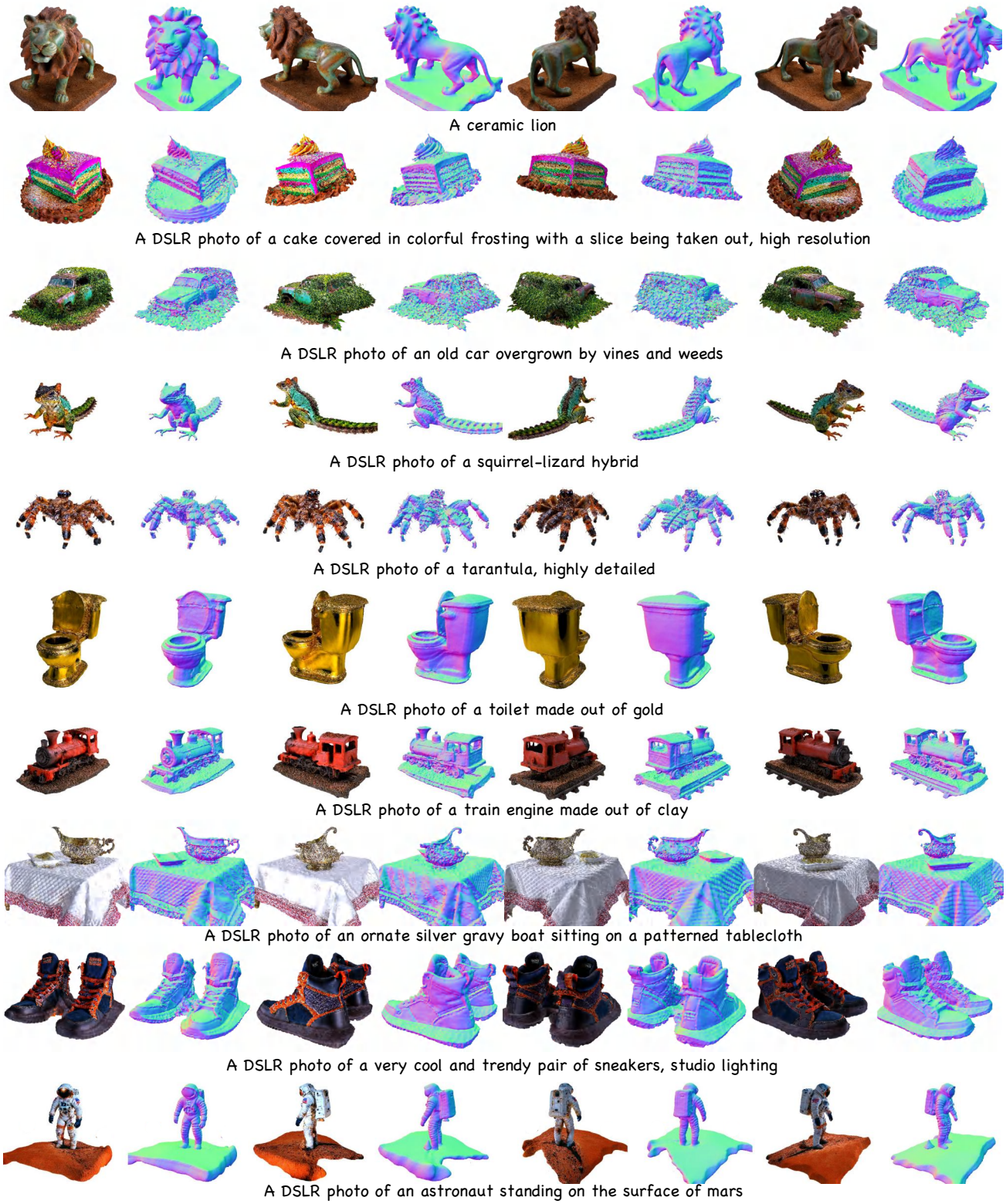


Figure S17. Visual results of *Ours (Sphere)* (Part I).

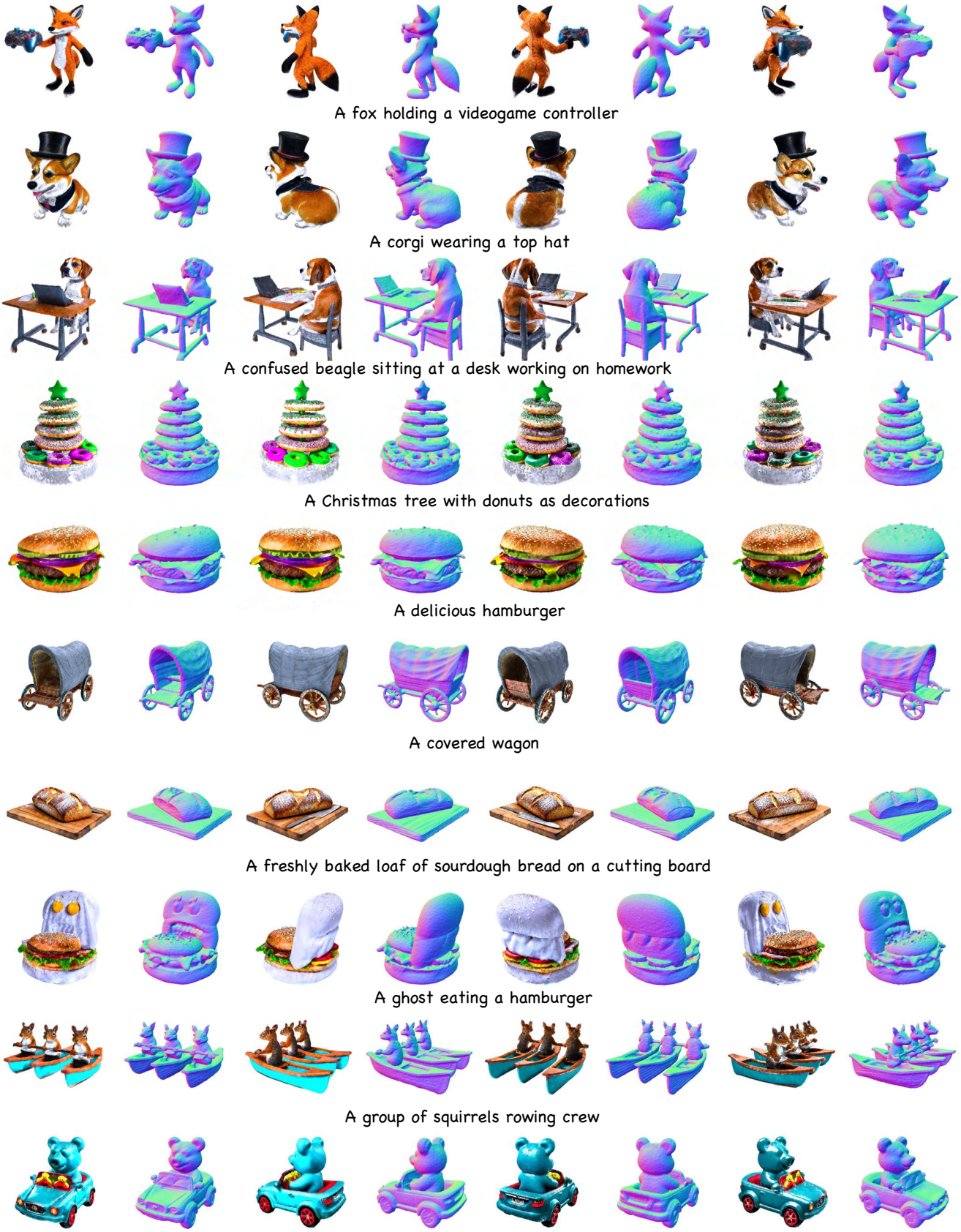


Figure S18. Visual results of *Ours (NeRF)* (Part I).

Figure S19. Visual results of *Ours (NeRF)* (Part II).

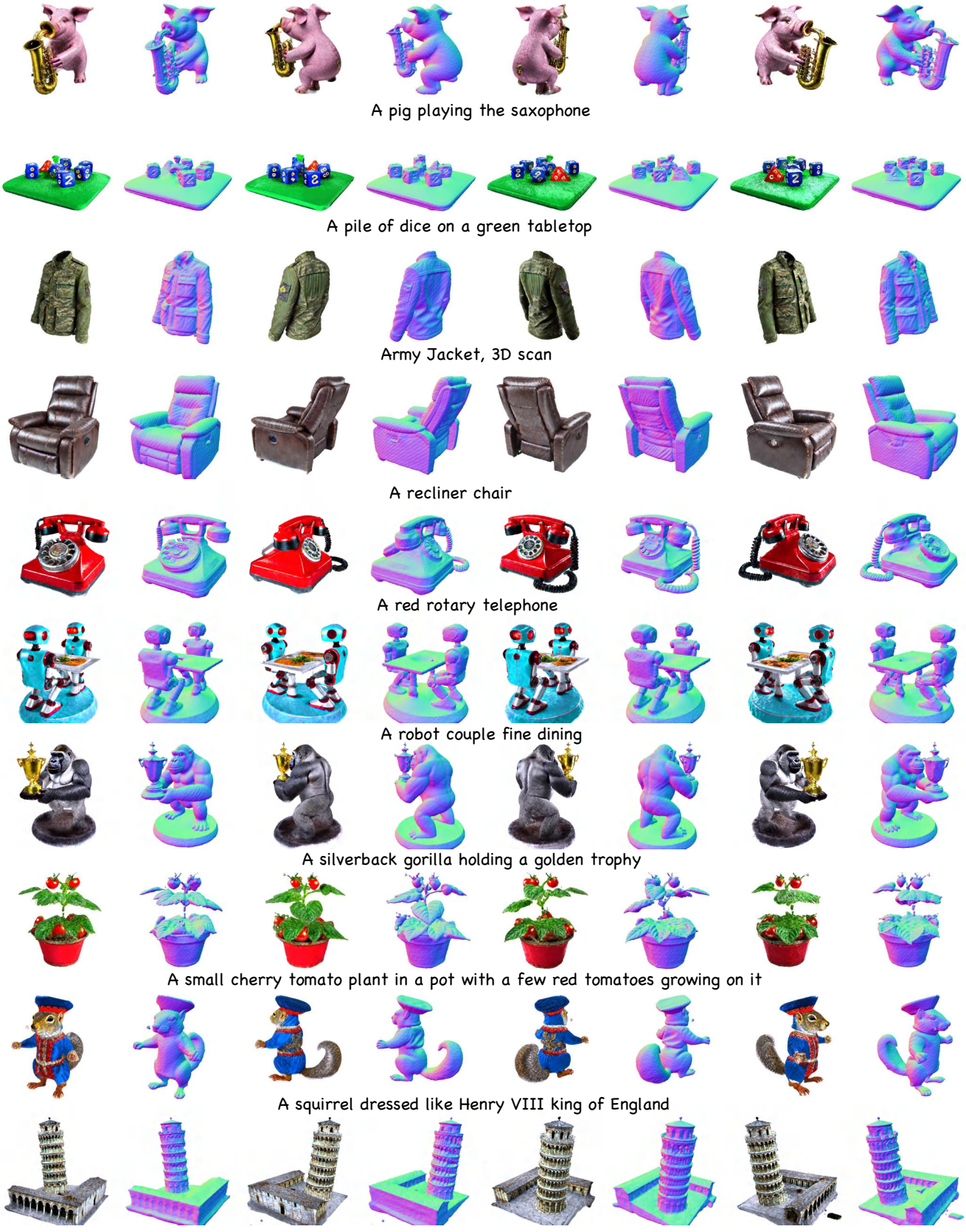
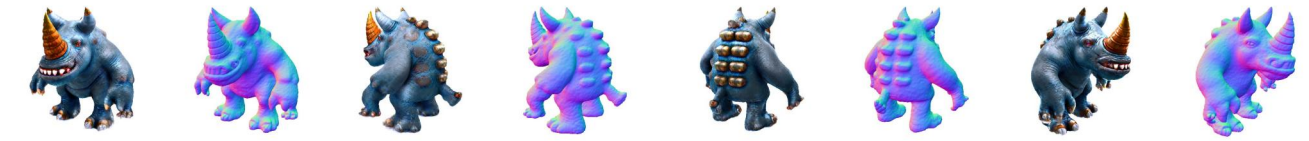


Figure S20. Visual results of *Ours (NeRF)* (Part III).

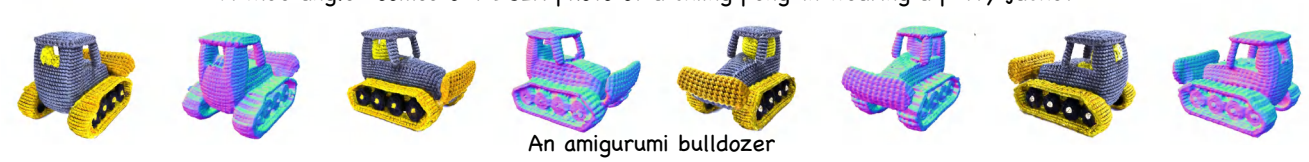


An monster that looks like a rhinoceros, cute, boss, game, character, highly detailed, photorealistic, 4K, HD

A tiger waiter at a fancy restaurant

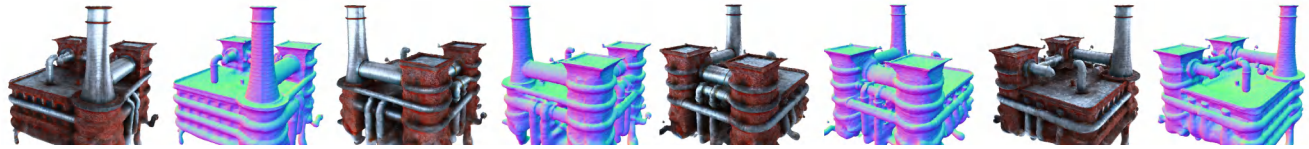
A wide angle DSLR photo of a colorful rooster

A wide angle zoomed out DSLR photo of a skiing penguin wearing a puffy jacket

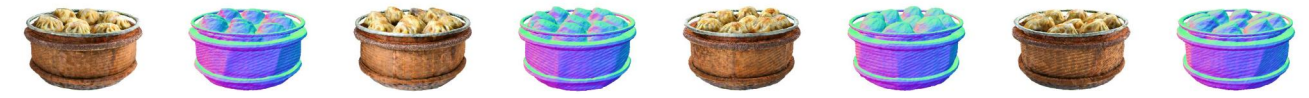


An amigurumi bulldozer

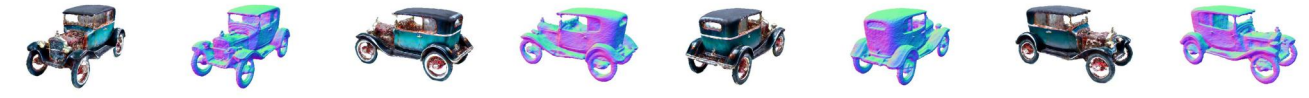
An exercise bike



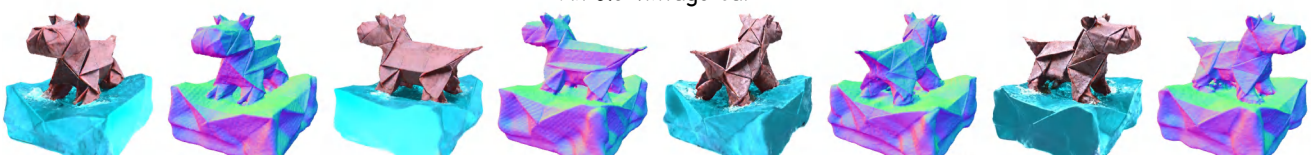
An intricate complex with steampowered machinery, twisting pipes, and brick warehouses, shrouded in a foggy, industrial atmosphere, 8K, blender 3d.



A steaming basket full of dumplings



An old vintage car



An origami hippo in a river

Figure S21. Visual results of *Ours (NeRF)* (Part IV).

References

[1] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Estimating and exploiting the aleatoric uncertainty in surface normal estimation. In *ICCV*, 2021. 1

[2] 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 *CVPR*, 2023. 2

[3] Weiyu Li, Rui Chen, Xuelin Chen, and Ping Tan. Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. *arXiv preprint arXiv:2310.02596*, 2023. 7

[4] Songyou Peng, Björn Häfner, Yvain Quéau, and Daniel Cremers. Depth super-resolution meets uncalibrated photometric stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017. 3

[5] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020. 1

[6] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1, 2

[7] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *NeurIPS*, 2022. 1

[8] 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, 5

[9] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 2