

GTR: Improving Large 3D Reconstruction Models through Geometry and Texture Refinement

Peiye Zhuang

Songfang Han

Chaoyang Wang

Aliaksandr Siarohin

Jiaxu Zou

Michael Vasilkovsky

Vladislav Shakhrai

Sergey Korolev

Sergey Tulyakov

Hsin-Ying Lee

Snap Inc.

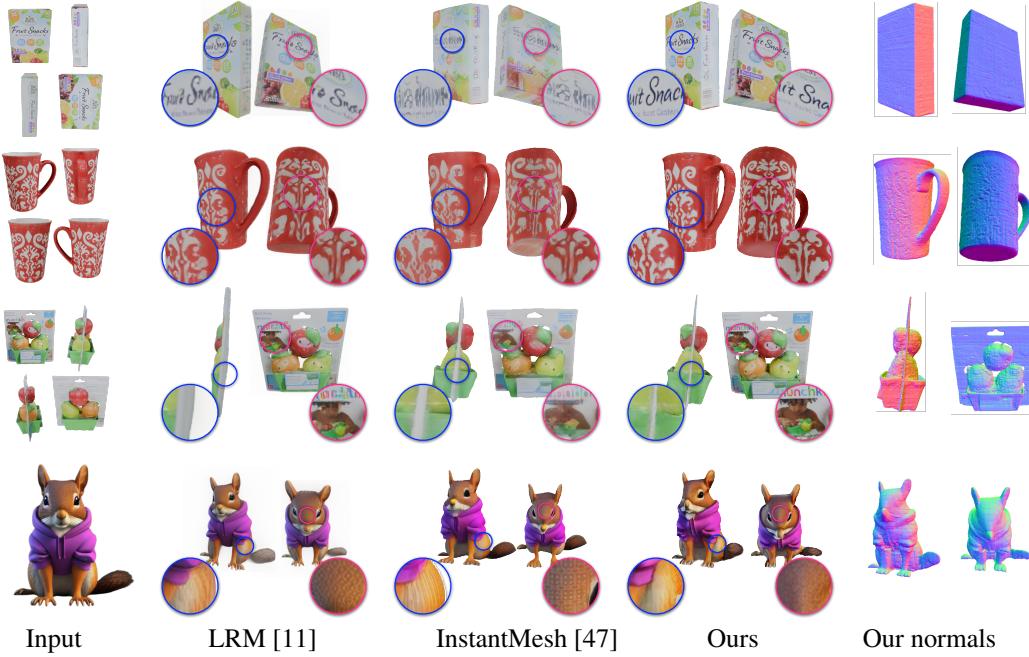


Figure 1: **Comparison with the baseline methods on the sparse-view reconstruction task.** We visualize the novel view results generated using either ground-truth 4-view images (row 1-3) from the GSO dataset [8], or a single generated image (row 4).

Abstract

We propose a novel approach for 3D mesh reconstruction from multi-view images. Our method takes inspiration from large reconstruction models like LRM [11] that use a transformer-based triplane generator and a Neural Radiance Field (NeRF) model trained on multi-view images. However, in our method, we introduce several important modifications that allow us to significantly enhance 3D reconstruction quality. First of all, we examine the original LRM architecture and find several shortcomings. Subsequently, we introduce corresponding modifications to the LRM architecture, which lead to improved multi-view image representation and more computationally efficient training. Second, in order to improve geometry reconstruction and enable supervision at full image resolution, we extract meshes from the NeRF field in a differentiable manner and fine-tune the NeRF model through mesh rendering. These modifications allow us to achieve state-of-the-art

performance on both 2D and 3D evaluation metrics, such as a PSNR of 28.67 on the Google Scanned Objects (GSO) dataset. Despite these superior results, our feed-forward model still struggles to reconstruct complex textures, such as text and portraits on assets. To address this, we introduce a lightweight per-instance texture refinement procedure. This procedure fine-tunes the triplane representation and the NeRF’s color estimation model on the mesh surface using the input multi-view images in just 4 seconds. This refinement improves the PSNR to 29.79 and achieves faithful reconstruction of complex textures, such as text. Additionally, our approach enables various downstream applications, including text/image-to-3D generation. Our project website is at <https://snap-research.github.io/GTR/>.

1 Introduction

The task of generating 3D assets from text or images has gained increasing attention and demonstrated wide applications in digital content generation and Virtual Reality (VR) [4, 6, 17]. Despite the scarcity of 3D data, efforts have been made to leverage pre-trained large-scale text-to-image diffusion models in different ways. Some works learn 3D asset generation from the pre-trained image diffusion models via score distillation techniques [5, 16, 26, 27, 39, 41, 51], which, however, require from several minutes to several hours for a single asset, due to the optimization nature of these methods. Other works propose to fine-tune image diffusion models into a multi-view image diffusion model [18–20, 35] using 3D asset datasets [7]. These multi-view images serve as an intermediate 3D representation, which is taken as input by large 3D reconstruction models (LRMs) [11, 15, 37, 42, 49] for asset generation. However, these previous approaches struggle to reconstruct faithful textures and high-quality geometry when using Marching Cube (MC) algorithms [21]. Moreover, after extracting meshes using MC, the texture quality degrades even further. In our method, we propose a hybrid approach to improve the quality of 3D reconstruction in terms of both geometry and texture. Specifically, we generate meshes with a feed-forward model and then apply lightweight fine-tuning to enhance the texture details.

We start developing the feed-forward mesh generation model by carefully examining the standard LRM architecture. First, we observed that DINO features tend to discard high-frequency image details, which are important for precise reconstitution, from the input images. Thus we replace the pre-trained DINO transformer [3] used in previous LRMs [11, 15, 37, 42, 49] with a convolutional encoder for the multi-view images. Moreover, because of the high computation requirements of the transformers, previous methods usually run a transformer at 32^2 triplane resolution and use a deconvolution to upsample this triplane. However, we noticed that reconstruction from the standard LRM most of the time exhibits regular grid-shape artifacts (see Fig. 1). We speculate that the nature of these artifacts is similar to grid-shape artifacts observed in 2D deconvolutional generators [25]. To address this, we replace all deconvolution layers with Pixelshuffle layers [34]. Finally, we employ two shallow Multi-layer Perceptrons (MLPs) to separately predict density and colors, which is beneficial for our following fine-tuning stages, which we will explain shortly.

Learning meshes is significantly harder than learning NeRF fields. Thus, we begin by training this modified architecture using NeRF volume rendering. With the trained NeRF model in place, we proceed to fine-tune the pipeline using mesh rendering (also known as rasterization). To achieve this, we employ Differentiable Marching Cubes (DiffMC) [43] to extract meshes from the NeRF density fields by transferring the densities to a signed distance function (SDF) representation. This enables us to render full-resolution images for supervision. Additionally, we employ a depth loss to guide the geometry extraction. Our feed-forward mesh generation pipeline significantly boosts the quality of the reconstructions compared to the results extracted from NeRFs using MC.

While the feed-forward mesh generation pipeline achieves advanced 3D reconstruction quality, it still faces challenges in accurately reconstructing intricate textures, such as text and complex patterns. To address this limitation and further enhance texture quality, we introduce a straightforward yet highly effective texture refinement procedure. Specifically, we fine-tune both the triplane representation and the color estimation model for each instance using sparse multi-view input images. As aforementioned, the shallow and separate density and color estimation models enable efficient updating of colors for surface points on the extracted meshes. In other words, the heavy triplane generator remains fixed in this texture refinement stage. This enables us to achieve rapid optimization, reaching 5 iterations per

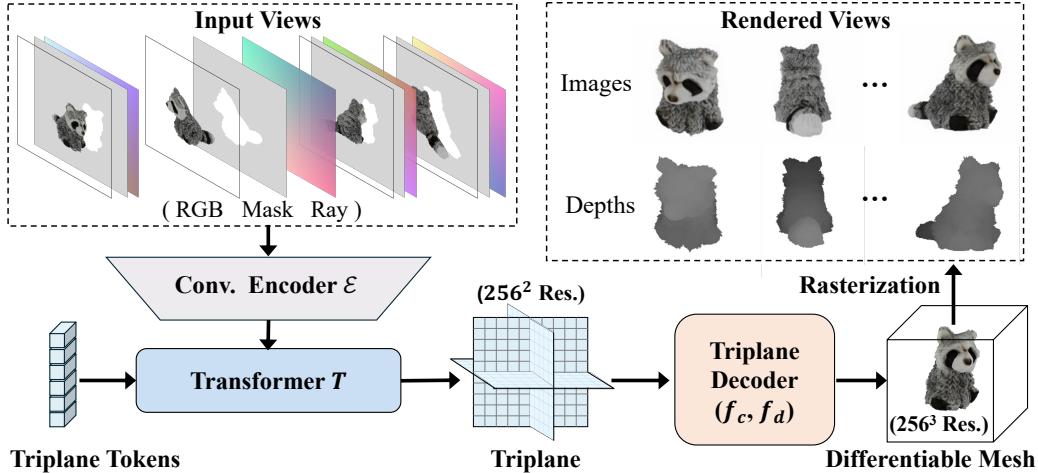


Figure 2: **Overview of our proposed approach for 3D reconstruction.** Our pipeline consists of a convolutional decoder \mathcal{E} , a transformer-based triplane generator, T , and a NeRF-based triplane decoder that contains two MLPs, f_c and f_d , for color and density prediction, respectively. In practice, the triplane resolution is set to 256, and the mesh representation has a grid size resolution of 256.

second on an A100 GPU. Remarkably, our method achieves faithful texture reconstruction with just 20 steps of fine-tuning on 4-view images, requiring a mere 4 seconds on an A100 GPU.

Our proposed approach enables faithful 3D reconstruction from the multi-view input images, as shown in Fig. 1. We conduct a comprehensive comparison of our method with multiple concurrent works [11, 37, 47] using Google Scanned Object (GSO) [8] and OmniObject3D [46] dataset, employing various evaluation metrics, including Peak signal-to-noise ratio (PSNR), Structural Similarity Index (SSIM), Learned Perceptual Image Patch Similarities (LPIPS) in 2D image space, and Chamfer Distance (CD) and Intersection over Union (IoU) in 3D geometry space. Extensive experiments show that our feed-forward model achieves superior results compared to the concurrent baseline approaches, while our per-instance refinement approach enables further texture improvement on text and complex patterns. We show **18%** improvement in PSNR, **30%** improvement in LPIPS, and **33%** better score in CD comparing to previous state-of-the-art. See Fig. 1 for visual examples.

In summary, we introduce a holistic design for multi-view image to 3D reconstruction to enhance generation quality for both geometry and texture. This is achieved through several modifications to the existing LRM model, fine-tuning the NeRF model using the differentiable mesh representation, and introducing an efficient per-instance texture refinement procedure using input images. We demonstrate that our model design effectively addresses existing shortcomings and improves the quality of object-centric 3D asset reconstruction from multi-view images. Furthermore, our model can be adapted to various downstream applications, such as text/image-to-3D generation tasks.

2 Related Work

Optimization-based 3D generation aims to use pre-trained large-scale text-to-image diffusion models [28, 29] for 3D generation, given the insufficient scale and diversity of existing 3D datasets. To distill 3D knowledge from the text-to-image models, a Score Distillation Sampling (SDS) approach and its variants [5, 26, 39, 41, 51] have been proposed. In these methods, noise is added to an image rendered from 3D models like NeRFs and subsequently denoised by a pre-trained text-to-image generative model [28]. The SDS approach aims to minimize the Kullback-Leibler (KL) divergence between a prior noise distribution and the estimated noise distribution from the text-to-image model. However, these SDS-based methods are time-consuming, usually taking up to hours to generate a single instance. Alternatively, feed-forward 3D generation models have been proposed to achieve faster generation.

Feed-forward 3D generation has gained increasing attention recently due to its speed advantage. Most existing feed-forward 3D generation pipelines consist of two stages: (i) a text prompt (or a

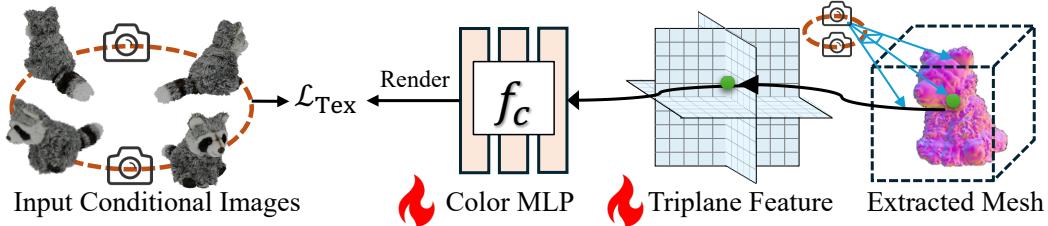


Figure 3: **Texture refinement for extracted meshes.** We refine the texture by fine-tuning the triplane feature of the asset and the color model, f_c , using the input images. The learnable components are marked as 🔥 . We use an L2 loss on the images, defined as \mathcal{L}_{Tex} .

single image) to multi-view generation and (ii) multi-view images to 3D shape reconstruction. In the first stage, multi-view images are generated from text or image input using a multi-view generator [18–20, 35], which are usually fine-tuned from image or video diffusion generation models [1, 28]. In the second stage, a 3D instance is reconstructed from input multi-view images. In this case, various 3D representations are used such as neural implicit fields [11, 15, 49], Gaussian Splatting [37, 48], or meshes [42, 47]. However, existing methods encounter challenges to either faithfully reconstruct textures when using implicit field representations [11, 15, 49], or difficult to extract explicit geometries when using Gaussian Splatting [37, 48]. In this work, we focus on the second stage of the pipeline, i.e., reconstructing 3D shapes from multi-view images. To address those challenges, we propose a novel 3D reconstruction approach that improves the 3D quality through geometry and texture refinement.

Differentiable mesh is a hybrid 3D representation that combines implicit and explicit surface representations, i.e., SDFs and meshes, and is suitable for 3D optimization. Recent popular representations include DMTet, Flexicubes, Differentiable Marching Cubes (DiffMC) [31, 32, 43]. In our work, we chose DiffMC [43] as it doesn’t require additional components beyond our existing model pipeline, in contrast to using DMTet and Flexicubes [31, 32], where a deformation and a weight prediction net are required.

MVS-based 3D reconstruction aims to generate novel views from sparse-view input images using Multi-view Stereo (MVS) techniques. Classic MVS methods leverage cost volumes [13, 30], point clouds [14, 36], or depth maps [2, 9] to learn blending weights for input sparse-view pixels for generating novel views. Recently, learning-based methods [10, 23, 38, 45, 50] have been proposed that can generalize to novel scenes. However, these methods require input views to have dense local overlap and struggle to generate 360-degree views of 3D assets. It is even more challenging when input multi-view images are generated without precise pixel-level alignment. Alternatively, in our work, we propose a simple yet effective texture refinement procedure that enables high-quality texture reconstruction from sparse-view input and is robust to synthetic images.

3 Method

We separate the technical details of our method into three parts. In Sec. 3.1, we explain the modifications made to the existing LRM architecture. In Sec. 3.2, we present a training procedure for our feed-forward mesh generation model. The overview of our feed-forward mesh generation pipeline is illustrated in Fig. 2. Finally, in Sec. 3.3, we introduce our per-instance texture refinement procedure. This procedure is highlighted in Fig. 3.

3.1 Improving the large 3D reconstruction architecture

A standard LRM [11] architecture consists of an image encoder, a transformer-based triplane generator with a deconvolutional triplane upsampler, and a NeRF-based triplane decoder. In our work, we propose several modifications to the LRM architecture outlined next.

Convolutional image encoder \mathcal{E} . LRM [11, 15, 49] models typically utilize pre-trained transformer network DiNO ViT [3]. However, we observe that since DiNO ViT [3] was designed for semantic understanding, it tends to ignore some local details irrelevant to image semantics but required for accurate reconstruction. Thus we propose to replace the DiNO ViT [3] architecture with a convolutional encoder. Since this encoder is trained from scratch along with other components it does



Figure 4: **Comparison with the baseline methods on OmniObject3D dataset [46]**. LRM [11] takes the front-view image as input while Instantmesh [47] and our method take the 4-view images as input. We present the novel view reconstruction results using these methods.

not exhibit the bias of pre-trained DiNO ViT [3]. An additional advantage of this encoder is that it does not require any modifications to consume additional inputs useful for 3D reconstruction. To this end, we complement the input images with binary foreground masks and Plücker coordinates for camera rays. We show a comparison of the training process using different encoders in Appendix A.

Triplane upsampler T . One of the LRM architecture shortcomings is the tendency to generate grid-shaped artifacts. We attribute this issue to the deconvolution operation utilized in a triplane upsampler. Indeed, to reduce the computational requirements, the original LRM proposed to run a transformer-based triplane generator on 32^2 resolution and later utilize deconvolution operation to upsample the triplane. The deconvolution operation is widely studied in GAN literature, for example, Odena et al. [25] show that 2D deconvolution generators are the main source of grid-shaped artifacts. To this end, we replace the deconvolution upsampling with a linear layer followed by a pixelshuffle layer [34]. This simple modification helps to alleviate grid-shaped texture artifacts.

NeRF decoders f_c, f_d . Unlike previous LRMs [11, 15], we utilize two separate MLPs, defined as f_c and f_d , to estimate colors and density, respectively. This modification does not impact performance; however, it serves a more practical purpose. For instance, we can train the color model f_c and freeze the density model f_d when fine-tuning asset texture, or vice versa if fine-tuning asset geometry.

3.2 Feed forward mesh generation model

The optimization through mesh representation may pose a significant challenge. Indeed, the gradients for backpropagation through mesh exist only in a small local neighborhood and, thus, convergence heavily depends on accurate initialization. To tackle this, we develop a two-stage training procedure, where in the first stage we utilize the volumetric rendering and optimize NeRF, and, in the second stage, we perform geometry refinement by optimizing through a mesh representation.

NeRF training stage. In this stage, we simply optimize our modified architecture (see Sec. 3.1) using a mean squared error (MSE) loss \mathcal{L}_{rgb} and an LPIPS loss $\mathcal{L}_{\text{LPIPS}}$ on images, defined as

$$\mathcal{L}_{\text{NeRF}} = \mathcal{L}_{\text{rgb}} + \lambda_p \mathcal{L}_{\text{LPIPS}}, \quad (1)$$

where λ_p denotes the loss weight.

Geometry refinement with NeRF initialization. In the geometry refinement stage, we fine-tune the entire pipeline using mesh rendering. Specifically, we transfer the density field to an SDF field:

$$\text{sdf} = -(d - s), \quad (2)$$

where d is the estimated density, s is a pre-defined level set for DiffMC, and $s = 10$, in practice.

Next, we render images, depths, and masks for training via mesh rendering (a.k.a rasterization). We fine-tune the pipeline using an MSE loss and an LPIPS loss on images, MSE losses on depths and masks, and a regularization loss on opacity [44]. Formally, we define the loss as:

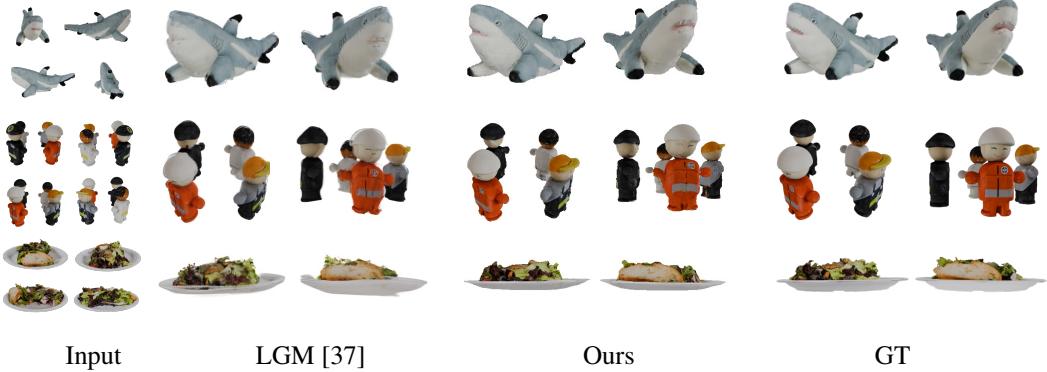


Figure 5: **Comparison with LGM [37] on GSO dataset [8]**. Both LGM [37] and our method take 4 images (*column 1*) as input and reconstruct novel views.

$$\mathcal{L}_{\text{Mesh}} = \mathcal{L}_{\text{rgb}} + \lambda_p \mathcal{L}_{\text{LPIPS}} + \lambda_d \mathcal{L}_{\text{depth}} + \lambda_m \mathcal{L}_{\text{mask}} + \lambda_r \mathcal{L}_{\text{reg}}, \quad (3)$$

where λ_p , λ_d , λ_m and λ_r are the loss weights.

3.3 Texture refinement for mesh representations

Inspired by previous works that utilize the Gaussian Splatting [37, 48] representation, where colors of the input images can be easily retained in Gaussian features, we notice a disparity in our pipeline, which lacks this color memorization scheme. Thus, we refine the triplane feature of an asset and the color model, f_c , using the input multi-view images, \mathbf{I}_{cond} , for surface points on the extracted mesh. We illustrate the refinement procedure in Fig. 3. The separated density and color models benefit the texture refinement procedure, as we only fine-tune a single MLP. We use an MSE loss on input images for texture refinement:

$$\mathcal{L}_{\text{Tex}} = \mathcal{L}_2(\mathbf{I}_{\text{cond}}, \hat{\mathbf{I}}_{\text{cond}}), \quad (4)$$

where \mathbf{I}_{cond} and $\hat{\mathbf{I}}_{\text{cond}}$ denote the ground-truth and predicted input images, respectively. Note that at this stage, the image encoder \mathcal{E} and the triplane generator T are fixed and used to generate the initial triplane features of assets. The density model, f_d , is also fixed and used to extract meshes.

4 Experiments

In Sec. 4.1, we explain training datasets, implementation details, and evaluation with the baseline methods. We present both quantitative and qualitative results in Sec. 4.2 and the ablation study in Sec. 4.3. Additional implementation details and comparisons are presented in Appendix.

4.1 Experiment Settings

Dataset: Our model is trained on a 140k asset dataset, which merges the filtered Objaverse dataset [7] with an internal 3D asset dataset. We filter the Objaverse dataset to retain 26k high-quality assets. We render 32 random views for each training asset. In Appendix A, we also provide additional ablation studies with our model trained solely on the Objaverse dataset [7] containing 100k assets.

Implementation Details. In practice, the loss weights are set to $\lambda_p = 0.5$, $\lambda_d = 0.5$, $\lambda_m = 1$ and $\lambda_r = 0.5$. Input multi-view images are of 512 resolution. The triplane transformer T contains 24 attention blocks with a hidden dimension of 1024. Each attention layer has 16 attention heads and each head has a dimension of 64. During the NeRF training stage, images are rendered at 512 resolution, and the NeRF model is trained using a patch size of 128^2 . We uniformly sample 256 points along each camera ray. The density and color models consist of a 3 and 4-layer MLP, respectively, with a hidden size of 512.

Table 1: Quantitative comparison on Google Scanned Objects (GSO) dataset [8].

Method	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)	CD (\downarrow)	IoU (\uparrow)
LRM [11]	20.4485	0.9044	0.1257	6.3825	0.3515
LGM (GS) [37]	25.2269	0.9254	0.0662	1.3725	0.6008
InstantMesh (NeRF) [47]	24.7401	0.9226	0.0802	1.1014	0.6353
InstantMesh (Mesh) [47]	24.4439	0.9197	0.0799	1.1147	0.6457
Ours (Feed-forward)	28.6734	0.9456	0.0549	0.7404	0.7077
Ours (Tex. refine)	29.7876	0.9603	0.0468		

In the NeRF training stage, we use an AdamW optimizer [22] with a learning rate $1e - 4$ and a weight decay of 0.05. Cosine scheduling is employed to gradually reduce the learning rate to 0 after 150k training iterations. We use a batch size of 512 on 32 A100 GPUs. For each asset, we randomly choose 4 views as input and another 4 views for supervision. In the geometry refinement stage, we choose a grid size of 256 during mesh extraction using DiffMC [43]. We use another AdamW optimizer [22] with a learning rate $5e - 5$. The batch size is 192 on 32 A100 GPUs. For each asset, we randomly choose 4 views as input and another 8 views for supervision. In the per-instance texture refinement stage, the learning rates for the triplane feature and the color model, f_c , are 0.15 and $1e - 4$, respectively.

Evaluation. We evaluate our method alongside baseline methods, including LRM [11], InstantMesh [47], and LGM [37] using the Google Scanned Objects (GSO) [8] and the OmniObject3D [46] datasets. We use the identical data lists of the GSO and OmniObject3D datasets and render camera orbits as outlined in Instantmesh [47]. Specifically, 300 GSO assets and 130 OmniObject3D assets (from 30 classes) are used for evaluation. We render 20 images for each asset in a trigonometric orbiting trajectory, i.e., maintaining uniform azimuths and elevations in $\{-30^\circ, 0^\circ, 30^\circ\}$. We use PSNR, SSIM, and LPIPS as image evaluation metrics, while CD and IoU are utilized for 3D geometry evaluation. To evaluate 3D geometry, we follow the mesh processing steps in InstantMesh [47]. Specifically, we reposition the generated meshes to the origin and align the coordinate system with the ground-truth meshes. We then rescale all meshes into a $[-1, 1]^3$ cube. We also use Iterative Closest Point (ICP) registration to align the generated meshes to the ground truth meshes. We sample 16000 points on the asset surface to compute the CD and IoU scores.

4.2 Results

Qualitative evaluation. We compare our method with baseline approaches using the GSO [8] and the OmniObject3D [46] datasets in Fig.1 and Fig.4, respectively. We observe that our method achieves more faithful texture reconstruction with finer details and more accurate geometry. For instance, in Fig. 1, our model can generate clear text on the first example and complex texture patterns on the second example. In the third example, our model enables the reconstruction of portraits printed on the asset. In Fig.5, we compare our method with LGM [37], which uses Gaussian Splatting [12] as a 3D representation. We observe that while LGM [37] can generate high-quality textures, it often tends to generate inaccurate floating Gaussian points in some regions, even when the input images are ground-truth multi-view images with precise pixel alignment. We present additional visual results in Appendix B.

Quantitative evaluation. We present the evaluation scores for the GSO [8] and the OmniObject3D [46] dataset in Tab. 1 and Tab. 2, respectively. The CD scores are presented by multiplying a rescale factor of 100. Note that both LGM [37] and InstantMesh [47] baselines are concurrent works. For InstantMesh [47], we compare both their results using either neural rendering or mesh rendering. Since LRM [11] takes a single image as input, we provide a front-view image to it. Our full approach achieves the best evaluation results in 2D and 3D evaluations. We also present the evaluation scores using our feed-forward model, i.e., without the texture refinement procedure. These results still show significant improvements in texture and geometry quality.

Additionally, the LGM [37] method serves as a strong baseline as we directly compare against their generated images rather than rendered images from extracted meshes. We note that the baselines that do not incorporate some explicit geometry representation during training typically yield inferior

Table 2: Quantitative comparison on OmniObject3D dataset [46].

Method	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)	CD (\downarrow)	IoU (\uparrow)
LRM [11]	18.0817	0.8880	0.1249	4.3467	0.4479
LGM (GS) [37]	22.8261	0.9128	0.0679	0.8933	0.6262
InstantMesh (NeRF) [47]	22.6087	0.9144	0.0761	0.6601	0.6711
InstantMesh (Mesh) [47]	22.1407	0.9097	0.0785	0.6026	0.6746
Ours (Feed-forward)	25.3724	0.9308	0.06	0.5042	0.7279
Ours (Tex. refine)	25.3996	0.9367	0.059		

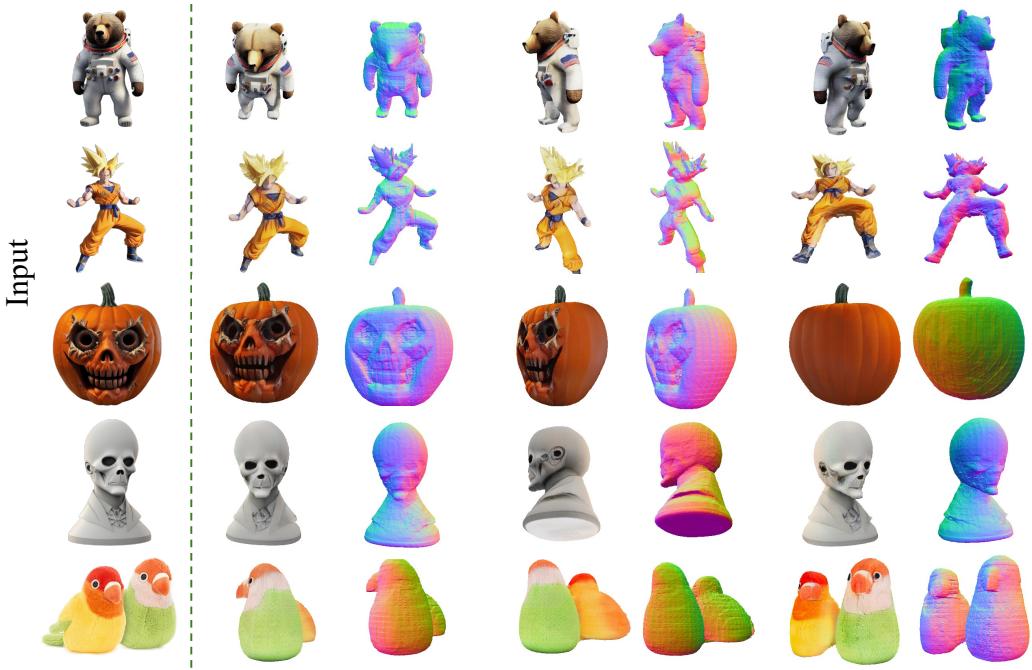


Figure 6: **Image-to-3D generation.** Our method can adapt to the text/image-to-3D generation tasks for both generated images from text (row 1-4) and real-world images (row 5). We visualize the input (column 1) and generated RGB and normal images (column 2-7).

rendering results when extracted to meshes. However, our method achieves better results than their directly generated images. Furthermore, it takes 1 minute to extract meshes using LGM [37] from Gaussian Splatting [12]. In contrast, our method enables the generation of meshes within 1 second, plus an additional 4 seconds for texture refinement, which in total is still faster than LGM [37] inference.

Applications. Our method enables downstream tasks such as text/image-to-3D generation. In this case, we generate multi-view images using pre-trained text-to-image and/or image-to-multiview diffusion models [28, 33]. In practice, we use Zero123++ [33] to generate 6-view images as the input for our model. We show the generated results in Fig. 6 and additional results in the Appendix B.

4.3 Ablation study

Geometry refinement. In Fig. 7, we compare results generated using NeRF+MC and NeRF with geometry refinement. We observe that directly extracting meshes from the NeRF field using MC leads to blurry texture results and a significant drop in 2D evaluation scores. In contrast, after fine-tuning at the geometry refinement stage, the rendered images show improved high-frequency details.



Figure 7: **Ablation study of geometry refinement.** On the left, we visualize the comparison between NeRF + MC, NeRF + Geometry refinement, and the ground-truth images. On the right, we present the evaluation scores using NeRF + MC.

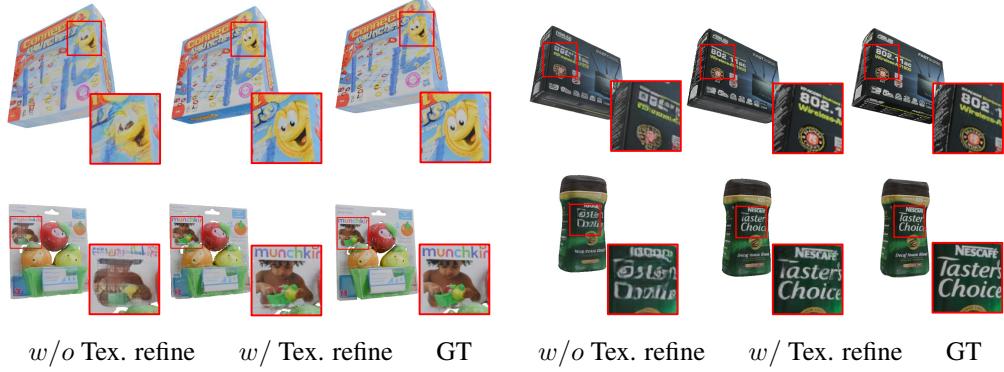


Figure 8: **Ablation study of the texture refinement procedure.** We visualize the mesh rendering results without (column 1, 4) or with (column 2, 5) the texture refinement procedure, and corresponding ground-truth images (column 3, 6).

Texture refinement. We visualize the novel view mesh-rendering results generated without and with per-instance texture refinement in Fig. 8. The results of the texture refinement appear to have superior detailed textures on mesh surfaces.

4.4 Limitations and Discussion

While our approach enables generating high-quality meshes with superior textures, there are several limitations to the current pipeline. First, we train the convolutional encoder \mathcal{E} from scratch. Using pre-trained models, such as the auto-encoder in Stable Diffusion models (SD) [28] may help speed up the convergence. We show preliminary results using a pre-trained VAE encoder from SD [28] in Appendix A. Moreover, adding a normal loss may help achieve smoother asset surfaces. Beyond that, we initialize the mesh rendering stage using NeRFs [24]. Alternatively, NeuS [40] generates SDF, which may be potentially suitable for mesh initialization, although, recent papers point out that NeuS may lose geometry details [43], which we leave for future exploration.

5 Conclusion

In this work, we introduce GTR, a large 3D reconstruction model that takes multi-view images as input. Our approach enables the generation of high-quality meshes with faithful texture reconstruction within seconds. We achieve this through three key contributions: modifications to the current LRM model architecture, the integration of end-to-end geometry refinement with NeRF initialization, and the implementation of a per-instance texture refinement procedure. Extensive experiments and evaluations conducted in both 2D and 3D spaces demonstrate the state-of-the-art performance of our approach. Additionally, our approach can be applied to various downstream applications, such as text/image-to-3D generation.

References

- [1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.
- [2] Neill DF Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *ECCV*, 2008.
- [3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.
- [4] Dave Zhenyu Chen, Haoxuan Li, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Scenetex: High-quality texture synthesis for indoor scenes via diffusion priors. In *CVPR*, 2024.
- [5] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *ICCV*, 2023.
- [6] Yen-Chi Cheng, Chieh Hubert Lin, Chaoyang Wang, Yash Kant, Sergey Tulyakov, Alexander Schwing, Liangyan Gui, and Hsin-Ying Lee. Virtual pets: Animatable animal generation in 3d scenes. *arXiv preprint arXiv:2312.14154*, 2023.
- [7] 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.
- [8] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *ICRA*, 2022.
- [9] David Gallup, Jan-Michael Frahm, Philippos Mordohai, Qingxiong Yang, and Marc Pollefeys. Real-time plane-sweeping stereo with multiple sweeping directions. In *CVPR*, 2007.
- [10] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, 2020.
- [11] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. In *ICLR*, 2024.
- [12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023.
- [13] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *IJCV*, 2000.
- [14] Maxime Lhuillier and Long Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *TPAMI*, 2005.
- [15] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. In *ICLR*, 2024.
- [16] 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 *CVPR*, 2023.
- [17] Chieh Hubert Lin, Hsin-Ying Lee, Willi Menapace, Menglei Chai, Aliaksandr Siarohin, Ming-Hsuan Yang, and Sergey Tulyakov. Infinicity: Infinite-scale city synthesis. In *ICCV*, 2023.
- [18] 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 *ICCV*, 2023.
- [19] 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. In *ICLR*, 2024.
- [20] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. In *CVPR*, 2023.
- [21] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

- [22] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.
- [23] Xinjun Ma, Yue Gong, Qirui Wang, Jingwei Huang, Lei Chen, and Fan Yu. Epp-mvsnet: Epipolar-assembling based depth prediction for multi-view stereo. In *ICCV*, 2021.
- [24] 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. *Communications of the ACM*, 2021.
- [25] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. doi: 10.23915/distill.00003. URL <http://distill.pub/2016/deconv-checkerboard>.
- [26] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023.
- [27] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, et al. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. In *ICLR*, 2024.
- [28] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.
- [29] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022.
- [30] Steven M Seitz and Charles R Dyer. Photorealistic scene reconstruction by voxel coloring. *IJCV*, 1999.
- [31] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *NeurIPS*, 2021.
- [32] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *TOG*, 2023.
- [33] 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.
- [34] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016.
- [35] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *ICLR*, 2024.
- [36] Robust Multiview Stereopsis. Accurate, dense, and robust multiview stereopsis. *TPAMI*, 2010.
- [37] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024.
- [38] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, Pablo Speciale, and Marc Pollefeys. Patchmatchnet: Learned multi-view patchmatch stereo. In *CVPR*, 2021.
- [39] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *CVPR*, 2023.
- [40] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.
- [41] 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 *NeurIPS*, 2023.
- [42] Zhengyi Wang, Yikai Wang, Yifei Chen, Chendong Xiang, Shuo Chen, Dajiang Yu, Chongxuan Li, Hang Su, and Jun Zhu. Crm: Single image to 3d textured mesh with convolutional reconstruction model. *arXiv preprint arXiv:2403.05034*, 2024.
- [43] Xinyue Wei, Fanbo Xiang, Sai Bi, Anpei Chen, Kalyan Sunkavalli, Zexiang Xu, and Hao Su. Neumanifold: Neural watertight manifold reconstruction with efficient and high-quality rendering support. *arXiv preprint arXiv:2305.17134*, 2023.

- [44] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Meshlrm: Large reconstruction model for high-quality mesh. *arXiv preprint arXiv:2404.12385*, 2024.
- [45] Zizhuang Wei, Qingtian Zhu, Chen Min, Yisong Chen, and Guoping Wang. Aa-rmvsnet: Adaptive aggregation recurrent multi-view stereo network. In *ICCV*, 2021.
- [46] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, et al. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *CVPR*, 2023.
- [47] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.
- [48] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arXiv preprint arXiv:2403.14621*, 2024.
- [49] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, et al. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. In *ICLR*, 2024.
- [50] Hongwei Yi, Zizhuang Wei, Mingyu Ding, Runze Zhang, Yisong Chen, Guoping Wang, and Yu-Wing Tai. Pyramid multi-view stereo net with self-adaptive view aggregation. In *ECCV*, 2020.
- [51] Joseph Zhu and Peiyie Zhuang. Hifa: High-fidelity text-to-3d with advanced diffusion guidance. In *ICLR*, 2024.

A Ablation study

DiNO encoder. We conduct experiments using different encoders. In Fig. 9 we present the validation PSNR curves during training, when using either our convolutional image encoder or a pre-trained DiNO ViT [3]. Specifically, our convolutional encoder is a single layer that downsamples input images from 512 to 32. The triplane generator is a self-attention transformer, identical in both settings. We train both models on 8 80G A100 GPUs. We observe that the DiNO experiment did not show improved convergence during the initial iterations. Alternatively, more careful designs could optimize the use of DiNO ViT [3], which we leave for future study.

Objaverse training dataset. In Fig. 9, we also show the training process with a dataset consisting solely of 100k Objaverse [7] images (*the yellow curve*). We did not observe a performance drop in the early stage compared to the other experiments in the figure, which were trained on our mixed dataset.

Vae encoder. In Fig.10, we show preliminary results using a pretrained VAE encoder¹ from an SD model [28]. To enable the VAE encoder to handle multi-channel input, we separately provide images, masks, and the camera rays to the encoder, then assemble the output features using a convolution layer. Experiments are run on 32 80G A100 GPUs. We observe that using a pretrained VAE encoder leads to better convergence in the early training stage. We attribute this to the good initialization provided by VAEs compared to training the convolutional encoder from scratch.

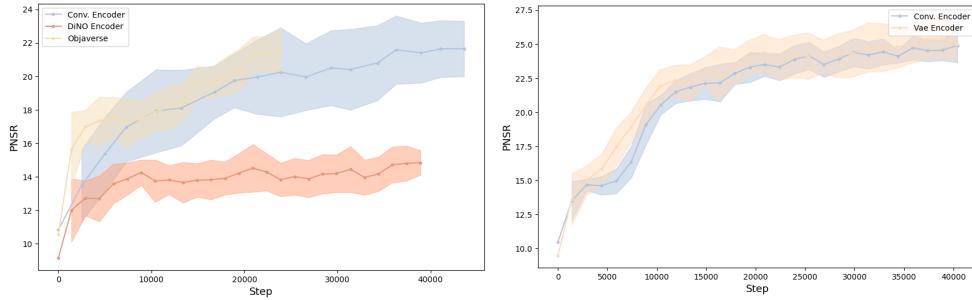


Figure 9: **Ablation study on image encoders** (Conv. vs DiNO) and dataset.

Figure 10: **Ablation study on image encoders** (Conv. vs VAE).

Texture refinement. We evaluate the texture refinement procedure by either learning the color model alone or the triplane feature alone. The 2D scores are presented in Tab. 3, and image comparisons are visualized in Fig. 11. Experiments show that jointly optimizing the triplane feature and the color model performs best compared to optimizing each component individually. The joint optimization produces superior textures with better details.

Table 3: **Ablation study for texture refinement on GSO dataset [8].** In the texture refinement procedure, we experiment with fine-tuning only the color model (*row 1*), fine-tuning only the triplane feature (*row 2*), and fine-tuning both (*row 3*).

Learning	PSNR (\uparrow)	SSIM (\uparrow)	LPIPS (\downarrow)
Color model	29.4176	0.9505	0.0558
Triplane	29.4911	0.9577	0.0486
Triplane + Color model	29.7876	0.9603	0.0468

B Additional results

We show additional results generated by our approach in Fig. 12- 17.

¹In practice, we use the pretrained SD VAE from <https://huggingface.co/madebyollin/taesd>



Figure 11: **Ablation study for texture refinement.** We evaluate the texture refinement procedure by either fine-tuning the color model alone (*column 1 and 4*), or fine-tuning the triplane feature alone (*column 2 and 5*), or jointly fine-tuning both (*column 3 and 6*).



Figure 12: **Additional visual comparison results with LGM [37] on GSO dataset [8].** We present the novel view rendering results using LGM [37] (*column 1-2*) and our approach (*column 3-4*), and ground-truth images (*column 5-6*).



Figure 13: **Additional visual comparison results with InstantMesh [47] on GSO dataset [8].** We present the novel view rendering results using InstantMesh [47] (*column 1-2*) and our approach (*column 3-4*), and ground-truth images (*column 5-6*).

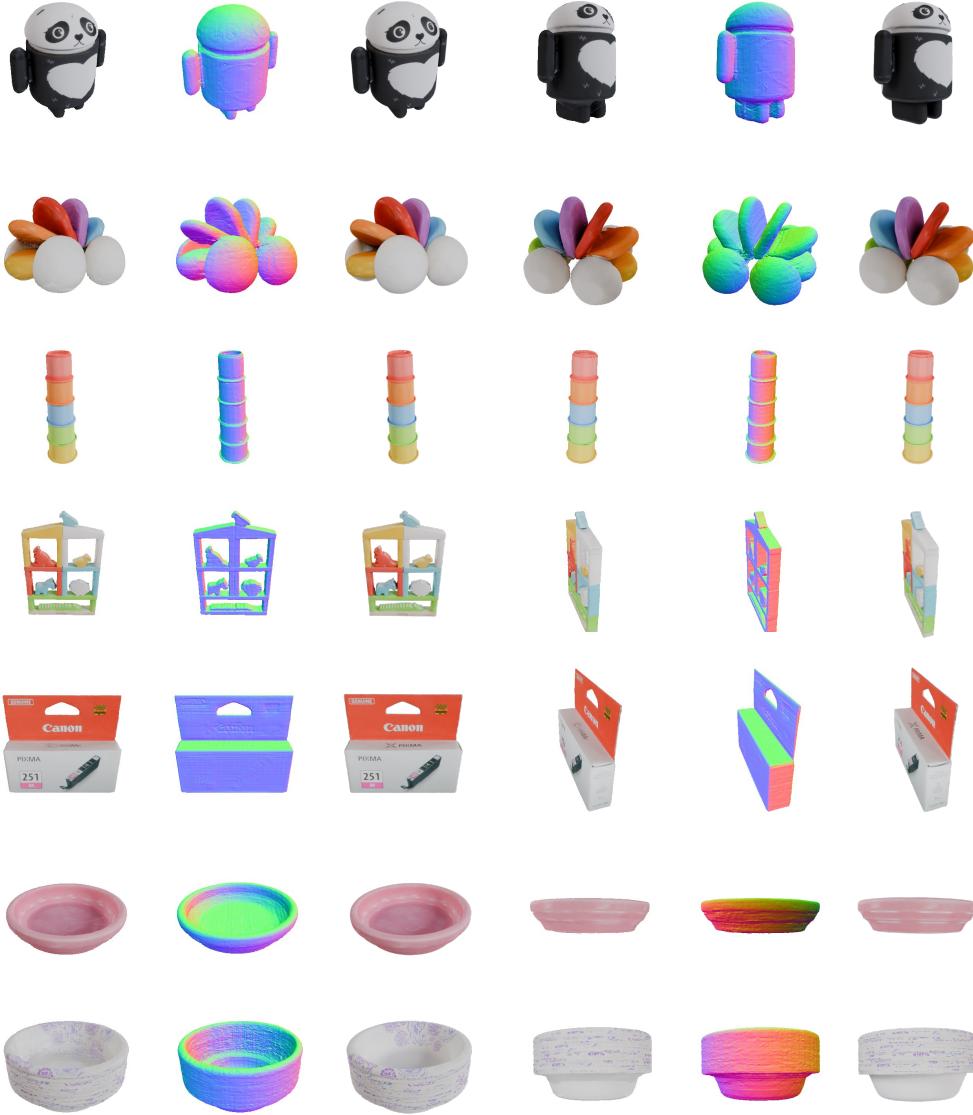


Figure 14: **Additional visual 3D reconstruction results on GSO dataset [8].** The input of our model is 4 orthogonal views. We show the novel view rendering RGB (*column 1, 4*) and normal images (*column 2, 5*), and the ground-truth images (*column 3, 6*).

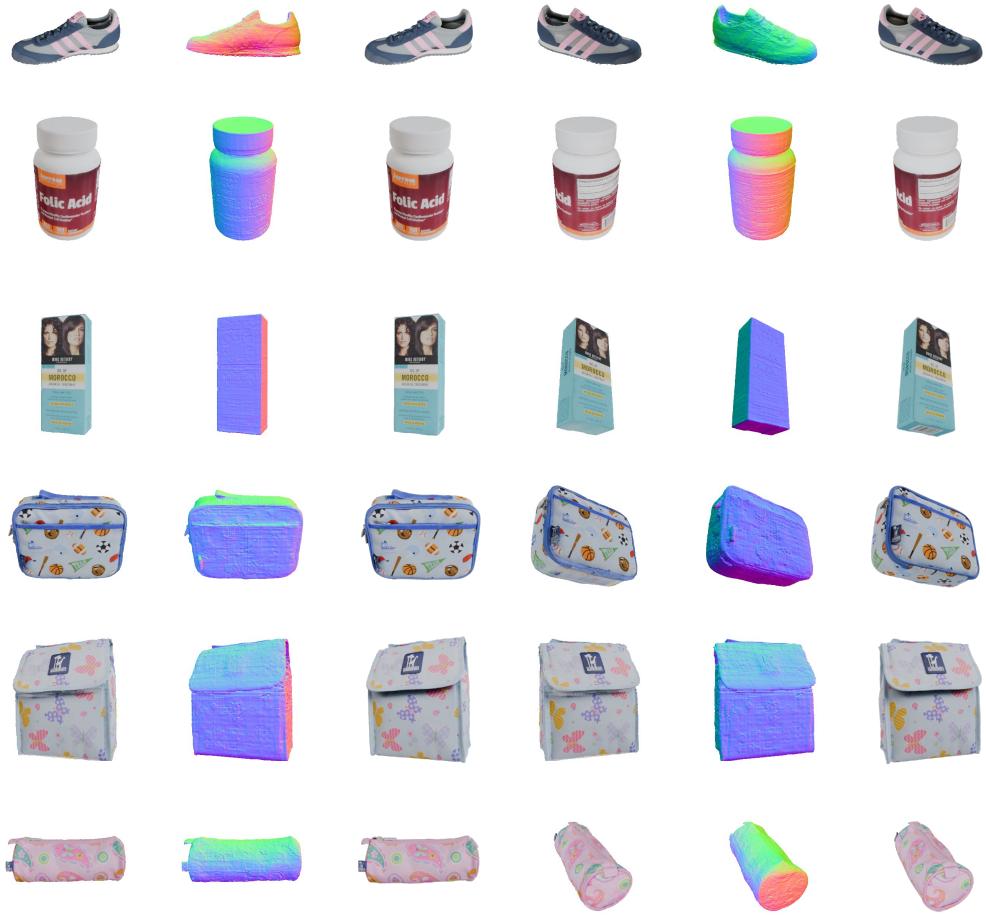


Figure 15: **Additional visual 3D reconstruction results on GSO dataset [8].** The input of our model is 4 orthogonal views. We show the novel view rendering RGB (*column 1, 4*) and normal images (*column 2, 5*), and the ground-truth images (*column 3, 6*).



Figure 16: **Additional visual 3D asset generation results.** The input images (*column 1*) are either generated from text using pre-trained text-to-image diffusion models [28] or online generated images.

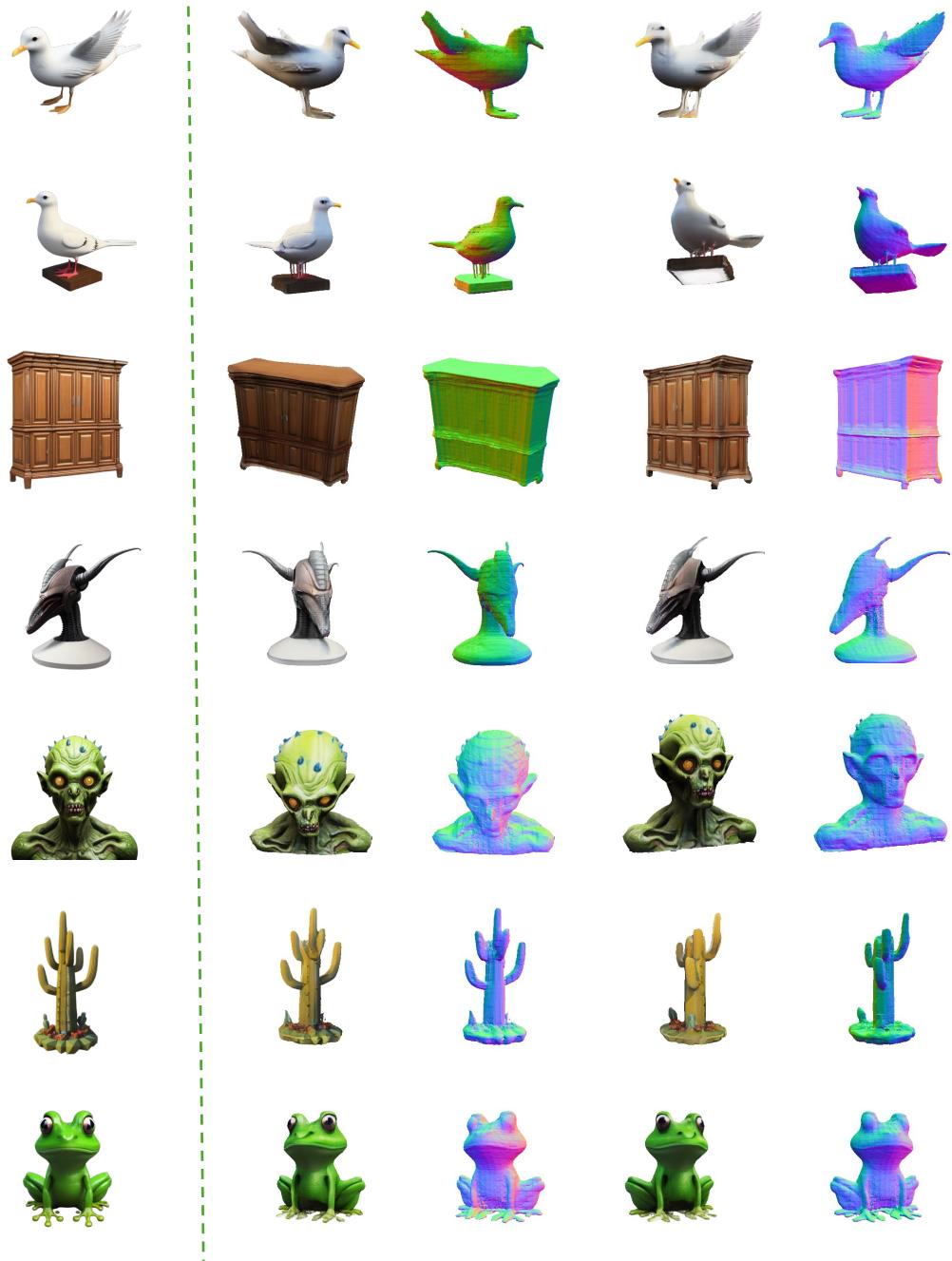


Figure 17: **Additional visual 3D asset generation results.** The input images (*column 1*) are either generated from text using pre-trained text-to-image diffusion models [28] or online generated images.