3D-ADAPTER: GEOMETRY-CONSISTENT MULTI-VIEW DIFFUSION FOR HIGH-QUALITY 3D GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-view image diffusion models have significantly advanced open-domain 3D object generation. However, most existing models rely on 2D network architectures that lack inherent 3D biases, resulting in compromised geometric consistency. To address this challenge, we introduce 3D-Adapter, a plug-in module designed to infuse 3D geometry awareness into pretrained image diffusion models. Central to our approach is the idea of *3D feedback augmentation*: for each denoising step in the sampling loop, 3D-Adapter decodes intermediate multi-view features into a coherent 3D representation, then re-encodes the rendered RGBD views to augment the pretrained base model through feature addition. We study two variants of 3D-Adapter: a fast feed-forward version based on Gaussian splatting and a versatile training-free version utilizing neural fields and meshes. Our extensive experiments demonstrate that 3D-Adapter not only greatly enhances the geometry quality of text-to-multi-view models such as Instant3D and Zero123++, but also enables high-quality 3D generation using the plain text-to-image Stable Diffusion. Furthermore, we showcase the broad application potential of 3D-Adapter by presenting high quality results in text-to-3D, image-to-3D, text-to-texture, and text-to-avatar tasks.

1 INTRODUCTION

Diffusion models (Ho et al., 2020; Song et al., 2021) have recently made significant strides in visual synthesis, achieving production-quality results in image generation (Rombach et al., 2022). However, the success of 2D diffusion does not easily translate to the 3D domain due to the scarcity of large-scale datasets and the lack of a unified, neural-network-friendly representation (Po et al., 2024). To bridge the gap between 2D and 3D generation, novel-view or multi-view diffusion models (Liu et al., 2023b; Long et al., 2024; Shi et al., 2023; Li et al., 2024; Chen et al., 2024; Voleti et al., 2024) have been finetuned from pretrained image or video models, facilitating 3D generation via a 2-stage paradigm involving multi-view generation followed by 3D reconstruction (Liu et al., 2023a; 2024a; Li et al., 2024; Wang et al., 2024; Xu et al., 2024a). While these models generally exhibit good *global semantic consistency* across different view angles, a pivotal challenge lies in achieving *local geometry consistency*. This entails ensuring precise 2D–3D alignment of local features and maintaining geometric plausibility. Consequently, these two-stage methods often suffer from floating artifacts or produce blurry, less detailed 3D outputs (Fig. 1 (c)).

To enhance local geometry consistency, previous works have explored inserting 3D representations and rendering operations into the denoising sampling loop, synchronizing either the denoised outputs (Gu et al., 2023; Xu et al., 2024b; Zuo et al., 2024; Zhang et al., 2024) or the noisy inputs (Liu et al., 2023c; Gao et al., 2024) of the network, a process we refer to as *I/O sync*. However, we observe that I/O sync generally leads to less detailed, overly smoothed textures and geometry (Fig. 1 (a)). This phenomenon can be attributed to two factors:

- Diffusion model sampling is sensitive to error accumulations (Li & van der Schaar, 2024). I/O sync methods insert 3D reconstruction and rendering operations into the denoiser in a way that disrupts the original model topology and introduces errors during each denoising step (unless reconstruction and rendering are perfect).

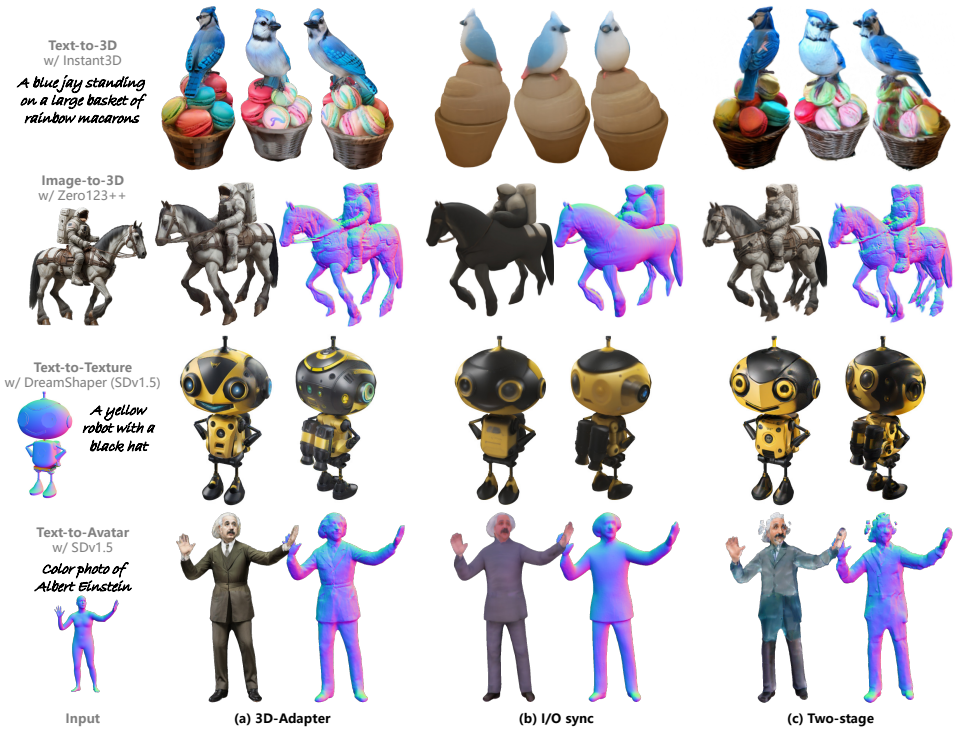

Figure 1: Comparison between the results generated by different architectures. Texture refinement is enabled for text-to-3D, image-to-3D, and text-to-avatar.

- For texture generation methods in Liu et al. (2023c); Gao et al. (2024); Zhang et al. (2024), I/O sync is equivalent to multi-view score averaging, which theoretically leads to mode collapse, causing the loss of fine details in the generated outputs. More theoretical analysis can be found in Appendix A.1.

To overcome the limitations of I/O sync, we propose a novel approach termed *3D feedback augmentation*, which attaches a 3D-aware parallel branch to the base model, while preserving the original network topology and avoiding score averaging. Essentially, this branch decodes intermediate features from the base model to reconstruct an intermediate 3D representation, which is then rendered, encoded, and fed back into the base model through feature addition, thus augmenting 3D awareness. Specifically, when using a denoising U-Net as the base model, we implement 3D feedback augmentation as *3D-Adapter*, which reuses a copy of the original U-Net with an additional 3D reconstruction module to build the parallel branch. Thanks to its ControlNet-like (Zhang et al., 2023) model reuse, 3D-Adapter requires minimal or, in cases where suitable off-the-shelf ControlNets are available, zero training.

To thoroughly evaluate its performance and demonstrate its flexibility, we have tested multiple variants of 3D-Adapter using various base models and reconstruction methods. The base models include Instant3D (Li et al., 2024), Zero123++ (Shi et al., 2023), Stable Diffusion(Rombach et al., 2022) v1.5 and its customizations. The reconstruction methods include GRM (Xu et al., 2024a), texture backprojection, Instant-NGP neural radiance field (NeRF) (Müller et al., 2022; Mildenhall et al., 2020) and DMTet mesh(Shen et al., 2021) optimization. This wide range of possible combinations makes 3D-Adapter models capable of many applications, as shown in Fig. 1. Extensive evaluations show that 3D-Adapter improves geometry consistency compared to the two-stage methods, without suffering from the quality degradation observed with I/O sync.

We summarize the main contributions of this paper as follows:

- We propose 3D-Adapter, which enables high-quality 3D generation with enhanced multi-view geometry consistency by integrating a 3D feedback module into a base image diffusion model.

- We demonstrate that 3D-Adapter is compatible with various base models and reconstruction methods, making it highly adaptable to a range of tasks.
- We conduct extensive experiments to show that 3D-Adapter improves geometry consistency while preserving visual quality, outperforming previous methods on text-to-3D, image-to-3D, and text-to-texture tasks.

2 Related Work

3D-native diffusion models. We define 3D-native diffusion models as injecting noise directly into the 3D representations (or their latents) during the diffusion process. Early works (Bautista et al., 2022; Dupont et al., 2022) have explored training diffusion models on low-dimensional latent vectors of 3D representations, but are highly limited in model capacity. A more expressive approach is training diffusion models on triplane representations (Chan et al., 2022), which works reasonably well on closed-domain data (Chen et al., 2023b; Shue et al., 2023; Gupta et al., 2023; Wang et al., 2023). Directly working on 3D grid representations is more challenging due to the cubic computation cost (Müller et al., 2023), so an improved multi-stage sparse volume diffusion model is proposed in Zheng et al. (2023). In general, 3D-native diffusion models face the challenge of limited data, and sometimes the extra cost of preprocessing the training data into 3D representations (e.g., NeRF), which limit their scalability.

Novel-/multi-view diffusion models. Trained on multi-view images of 3D scenes, view diffusion models inject noise into the images (or their latents) and thus benefit from existing 2D diffusion research. Watson et al. (2023) have demonstrated the feasibility of training a conditioned novel view generative model using purely 2D architectures. Subsequent works (Shi et al., 2024; 2023; Liu et al., 2023b; Long et al., 2024) achieve open-domain novel-/multi-view generation by fine-tuning the pre-trained 2D Stable Diffusion model (Rombach et al., 2022). However, 3D consistency in these models is generally limited to global semantic consistency because it is learned solely from data, without any inherent architectural bias to support detailed local alignment.

Two-stage 3D generation. Two-stage methods (Fig. 2 (a)) link view diffusion with multi-view 3D reconstruction models, offering a significant speed advantage over score distillation sampling (SDS) (Poole et al., 2023). Liu et al. (2023a) initially combine Zero-1-to-3 (Liu et al., 2023b) with SparseNeuS (Long et al., 2022), and subsequent works (Liu et al., 2024a; Xu et al., 2024a; Long et al., 2024; Tang et al., 2024a) have further explored more effective multi-view diffusion models and enhanced reconstruction methods. A common issue with two-stage approaches is that existing reconstruction methods, often designed for or trained under conditions of perfect consistency, lack robustness to local geometric inconsistencies. This may result in floaters and texture seams.

3D-aware view diffusion and I/O sync. To introduce 3D-awareness in single-image diffusion models, Anciukevicius et al. (2023); Tewari et al. (2023) elevate image features into 3D NeRFs to render denoised views. Extending this concept to multi-view diffusion, DMV3D (Xu et al., 2024b) represents an end-to-end feed-forward output sync approach. However, these methods often produce blurry outputs due to the information loss in the absence of residual connections. Liu et al. (2024b) attempt to preserve these connections through input sync and attention-based feature fusion, yet it lacks a robust architecture, leading to subpar quality as noted in Liu et al. (2023a). Alternatively, employing feed-forward Gaussian reconstruction models (Xu et al., 2024a; Tang et al., 2024a) for output sync preserves more information but often struggles to coherently fuse inconsistent views, resulting in only marginal improvements in 3D consistency as seen in Zuo et al. (2024). On the other hand, optimization-based I/O sync methods in Gu et al. (2023); Liu et al. (2023c); Gao et al. (2024) either require strong local conditioning or suffer from the pitfalls of score averaging, resulting in overly smoothed textures.

3 Preliminaries

Let $p(\boldsymbol{x}|\boldsymbol{c})$ denote the real data distribution, where $\boldsymbol{c}$ is the condition (e.g., text prompts) and $\boldsymbol{x} \in \mathbb{R}^{V\times 3\times H\times W}$ denotes the V-view images of a 3D object. A Gaussian diffusion model defines a diffusion process that progressively perturb the data point by adding an increasing amount of Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$, yielding the noisy data point $\boldsymbol{x}_t := \alpha_t \boldsymbol{x}_i + \sigma_t \boldsymbol{\epsilon}$ at diffusion timestep t, with pre-defined noise schedule scalars α_t, σ_t. A denoising network D is then tasked with removing

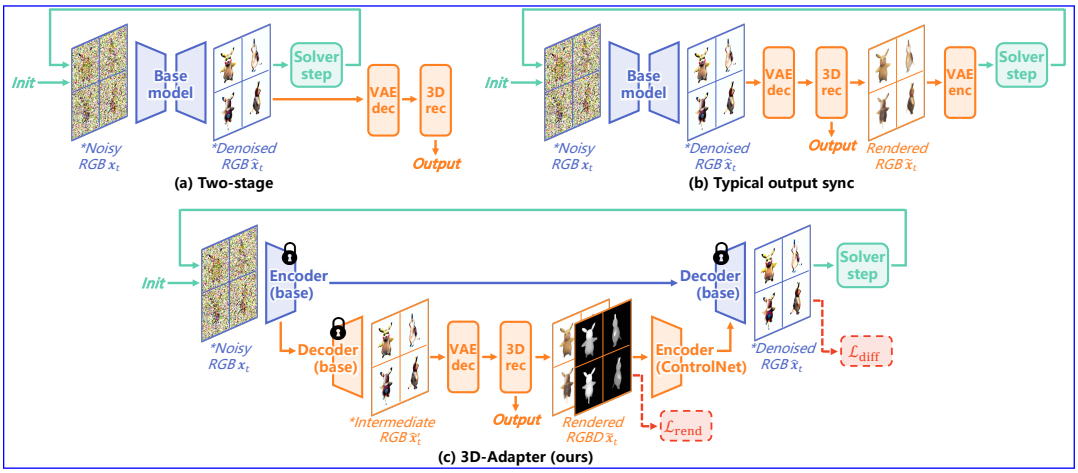

Figure 2: Comparison between different architectures. For brevity, we omit the condition encoders (e.g., text encoders), the rendered alpha channel, and the noisy RGB input for the ControlNet. For LDMs Rombach et al. (2022), VAE encoders and decoders are required, and * denotes RGB latents.

the noise from $\boldsymbol{x}_t$ to predict the denoised data point. The network is typically trained with an L2 denoising loss:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{t,\boldsymbol{c},\boldsymbol{x},\boldsymbol{\epsilon}}\left[\frac{1}{2} w_t^{\text{diff}} \| D(\boldsymbol{x}_t, \boldsymbol{c}, t) - \boldsymbol{x} \|^2\right], \tag{1}$$

where $t \sim \mathcal{U}(0, T)$, and w_t^{diff} is an empirical time-dependent weighting function (e.g., SNR weighting $w_t^{\text{diff}} = (\alpha_t/\sigma_t)^2$). At inference time, one can sample from the model using efficient ODE/SDE solvers (Lu et al., 2022) that recursively denoise $\boldsymbol{x}_t$, starting from an initial noisy state $\boldsymbol{x}_{t_{\text{init}}}$, until reaching the denoised state $\boldsymbol{x}_0$. Note that in latent diffusion models (LDM) (Rombach et al., 2022), both diffusion and denoising occur in the latent space. For brevity, we do not differentiate between latents and images in the equations, assuming VAE encoding and decoding as necessary.

I/O sync baseline. We broadly define I/O sync as inserting a 3D representation and a rendering/projecting operation at the input or output end of the denoising network to synchronize multiple views. Input sync is primarily used for texture generation, and it is essentially equivalent to output sync, assuming linearity and synchronized initialization (detailed in the Appendix A.1). Therefore, for simplicity, this paper considers only output sync as the baseline. As depicted in Fig. 2 (b), a typical output sync model can be implemented by reconstructing a 3D representation from the denoised outputs $\hat{\boldsymbol{x}}_t$, and then re-rendering the views from 3D to replace the original outputs.

4 3D-Adapter

To overcome the limitations of I/O sync, our key idea is the 3D feedback augmentation architecture, which involves reconstructing a 3D representation midway through the denoising network and feeding the rendered views back into the network using ControlNet-like feature addition. This architecture preserves the original flow of the base model while effectively leveraging its inherent priors.

Based on this idea, we propose the 3D-Adapter, as illustrated in Fig. 2 (c). For each denoising step, after passing the input noisy views $\boldsymbol{x}_t$ through the base U-Net encoder, we use a copy of the base U-Net decoder to first output intermediate denoised views $\hat{\boldsymbol{x}}'_t$. A 3D reconstruction model then lifts these intermediate views to a coherent 3D representation, from which consistent RGBD views $\tilde{\boldsymbol{x}}_t$ are rendered and fed back into the network through a ControlNet encoder. The output features from this encoder are added to the base encoder features, which are then processed again by the base

decoder to produce the final denoised output $\hat{\boldsymbol{x}}_t$. The full denoising step can be written as:

$$\hat{\boldsymbol{x}}_t = D_{\text{aug}}(\boldsymbol{x}_t, \boldsymbol{c}, t, \underbrace{R(\overbrace{D(\boldsymbol{x}_t, \boldsymbol{c}, t)}^{\hat{\boldsymbol{x}}'_t})}_{\hat{\boldsymbol{x}}_t}). \tag{2}$$

where R denotes 3D reconstruction and rendering, and D_{aug} denotes the augmented U-Net with feedback ControlNet.

Various 3D-Adapters can be implemented depending on the choice of base model and 3D reconstruction method, as described in the following subsections.

4.1 3D-Adapter Using Feed-Forward GRM

GRM (Xu et al., 2024a) is a feed-forward sparse-view 3D reconstruction model based on 3DGS. In this section, we describe the method to train GRM-based 3D-Adapters for the text-to-multi-view model Instant3D (Li et al., 2024) and image-to-multi-view model Zero123++ (Shi et al., 2023).

Training phase 1: finetuning GRM. GRM is originally trained on consistent ground truth input views, and is not robust to low-quality intermediate views, which are often highly inconsistent and blurry. To overcome this challenge, we first finetune GRM using the intermediate images $\hat{\boldsymbol{x}}'_t$ as inputs, where the time t is randomly sampled just like in the diffusion loss. In this training phase, we freeze the base encoder and decoder of the U-Net, and initialize GRM with the official checkpoint for finetuning. As shown in Fig. 2 (c), a rendering loss $\mathcal{L}_{\text{rend}}$ is employed to supervise GRM with ground truth novel views. Specifically, both the appearance and geometry are supervised using the combination of an L1 loss L_1^{RGBAD} on RGB/alpha/depth maps, and an LPIPS loss $L_{\text{LPIPS}}^{\text{RGB}}$ (Zhang et al., 2018) on RGB only. The loss is computed on 16 rendered views $\tilde{\boldsymbol{X}}_t \in \mathbb{R}^{16\times5\times512\times512}$ and the corresponding ground truth views $\boldsymbol{X}_{\text{gt}}$, given by:

$$\mathcal{L}_{\text{rend}} = \mathbb{E}_{t,\boldsymbol{c},\boldsymbol{x},\epsilon}\left[w_t^{\text{rend}}\left(L_1^{\text{RGBAD}}\left(\tilde{\boldsymbol{X}}_t, \boldsymbol{X}_{\text{gt}}\right) + L_{\text{LPIPS}}^{\text{RGB}}\left(\tilde{\boldsymbol{X}}_t, \boldsymbol{X}_{\text{gt}}\right)\right)\right], \tag{3}$$

where w_t^{rend} is a time-dependent weighting function. We use $w_t^{\text{rend}} = \alpha_t / \sqrt{{\alpha_t}^2 + {\sigma_t}^2}$. The L1 RGBAD loss also employs channel-wise weights, which are detailed in our code.

Training phase 2: finetuning feedback ControlNet. In this training phase, we freeze all modules except the feedback ControlNet encoder, which is initialized with the base U-Net weights for finetuning. Following standard ControlNet training method, we employ the diffusion loss in Eq. (1) to finetune the RGBD feedback ControlNet. To accelerate convergence, we feed rendered RGBD views of a less noisy timestep $\tilde{\boldsymbol{x}}_{0.1t}$ to the ControlNet during training.

Inference: guided 3D feedback augmentation. One potential issue is that the ControlNet encoder may overfit the finetuning dataset, resulting in an undesirable bias that persists even if the rendered RGBD $\tilde{\boldsymbol{x}}_t$ is replaced with a zero tensor. To mitigate this issue, inspired by classifier-free guidance (CFG) (Ho & Salimans, 2021), we replace $\hat{\boldsymbol{x}}_t$ with the guided denoised views $\hat{\boldsymbol{x}}_t^{\text{G}}$ during inference to cancel out the ControlNet bias:

$$\hat{\boldsymbol{x}}_t^{\text{G}} = \lambda_{\text{aug}}(D_{\text{aug}}(\boldsymbol{x}_t, \boldsymbol{c}, t, \tilde{\boldsymbol{x}}_t) - D_{\text{aug}}(\boldsymbol{x}_t, \boldsymbol{c}, t, \boldsymbol{0})) + \lambda_{\boldsymbol{c}} D(\boldsymbol{x}_t, \boldsymbol{c}, t) + (1 - \lambda_{\boldsymbol{c}}) D(\boldsymbol{x}_t, \boldsymbol{0}, t), \tag{4}$$

where $\lambda_{\boldsymbol{c}}$ is the regular condition CFG scale, and λ_{aug} is our feedback augmentation guidance scale. During training, we feed zero tensors to the ControlNet with a 20% probability, so that $D_{\text{aug}}(\boldsymbol{x}_t, \boldsymbol{c}, t, \boldsymbol{0})$ learns a meaningful dataset bias.

Training details. We adopt various techniques to reduce the memory footprint, including mixed precision training, 8-bit AdamW (Dettmers et al., 2022; Loshchilov & Hutter, 2019), gradient checkpointing, and deferred back-propagation (Xu et al., 2024a; Zhang et al., 2022). The adapter is trained with a total batch size of 16 objects on 4 A6000 GPUs (VRAM usage peaks at 39GB). In phase 1, GRM is finetuned with a small learning rate of 5×10^{-6} for 2k iterations (for Instant3D, taking 3 hours) or 4k iterations (for Zero123++, taking 9 hours). In phase 2, ControlNet is finetuned with a learning rate of 1×10^{-5} for 5k iterations (taking 8 hours for Instant3D and 5 hours for Zero123++).

47k (for Instant3D) or 80k (for Zero123++) objects from a high-quality subset of Objaverse (Deitke et al., 2023) are rendered as the training data.

4.2 3D-ADAPTER USING 3D OPTIMIZATION/TEXTURE BACKPROJECTION

Feed-forward 3D reconstruction methods, like GRM, are typically constrained by specific camera layouts. In contrast, more flexible reconstruction approaches, such as optimizing an Instant-NGP NeRF (Müller et al., 2022; Mildenhall et al., 2020) and DMTet mesh (Shen et al., 2021), can accommodate diverse camera configurations and achieve higher-quality results, although they require longer optimization times.

In this subsection, we design an optimization-based 3D adapter, using Stable Diffusion v1.5 (Rombach et al., 2022) as the base model, where each view is denoised independently, with the adapter being the only component that shares information across views.

During the sampling process, the adapter performs NeRF optimization for the first 60% of the denoising steps. It then converts the color and density fields into a texture field and DMTet mesh, respectively, to complete the remaining 40% denoising steps. All optimizations are incremental, meaning the 3D state from the previous denoising step is retained to initialize the next. As a result, only 96 optimization steps are needed per denoising step. Alternatively, for texture generation only, multi-view aggregation can be achieved by backprojecting the views into UV space and blending the results according to visibility. For feedback augmentation, the off-the-shelf "tile" ControlNet, originally trained for superresolution, performs effectively for RGB feedback due to its robustness to blurry inputs, while the off-the-shelf depth ControlNet can naturally handle depth feedback.

It should be noted that, when using single-image diffusion as the base model, 3D-Adapter alone cannot provide the necessary global semantic consistency for 3D generation. Therefore, it should be complemented with other sources of consistency, such initialization with partial noise like SDEdit (Meng et al., 2022) or extra conditioning from ControlNets. For text-to-avatar generation, we use rendered views of a human template for SDEdit initialization with the initial timestep t_{init} set to $0.88T$, and employ an extra pose ControlNet for conditioning. For text-to-texture generation, global consistency is usually good due to ground truth depth conditioning.

Optimization loss functions. For NeRF/mesh optimization, we employ L1 and LPIPS losses on RGB and alpha maps, and total variation (TV) loss on normal maps. Additionally, we enforce stronger geometry regularization using ray entropy loss for NeRF, and Laplacian smoothing loss (Sorkine et al., 2004) plus normal consistency loss for mesh, making the optimization more robust to imperfect intermediate views $\hat{\boldsymbol{x}}_t'$. More details can be found in Appendix C.

4.3 TEXTURE POST-PROCESSING

To further enhance the visual quality of objects generated from text, we implement an optional texture refinement pipeline as a post-processing step. First, when using the GRM-based 3D-Adapter, we convert the generated 3DGS into a textured mesh via TSDF integration. With the initial mesh, we render six surrounding views and apply per-view SDEdit refinement ($t_{\text{init}} = 0.5T$) using Stable Diffusion v1.5 with "tile" ControlNet. Finally, the refined views are aggregated into the UV space using texture backprojection. For fair comparisons in the experiments, this refinement step is not used by default unless specified otherwise.

5 EXPERIMENTS

5.1 EVALUATION METRICS

To evaluate the results generated by 3D-Adapter and compare them to various baselines and competitors, we compute the following metrics based on the rendered images of the generated 3D representations:

- **CLIP score** (Radford et al., 2021; Jain et al., 2022): Evaluates image–text alignment in text-to-3D, text-to-texture, and text-to-avatar tasks. We use CLIP-ViT-L-14 for all CLIP-related metrics.
- **Aesthetic score** (Schuhmann et al., 2022): Assesses texture details. The user study in Wu et al. (2024) revealed that this metric highly correlates with human preference in texture details.
- **FID** (Heusel et al., 2017): Measures the visual quality when reference test set images are available, applicable to text-to-3D models trained on common dataset and all image-to-3D models.

Table 1: Text-to-3D: comparison with baselines, parameter sweep, and ablation studies, on the validation set.

ID	Method	CLIP↑	Aesthetic↑	FID↓	MDD $/10^{-7}$↓
A0	Two-stage (original GRM)	27.02	4.48	34.19	232.4
A1	I/O sync (original GRM)	24.62	4.35	63.22	1239.7
A2	A1 + GRM finetuning	22.57	4.16	70.35	**1.7**
A3	A2 + dynamic blending	25.95	4.39	44.62	2.8
B0	3D-Adapter λ_{aug}=1	**27.31**	4.54	**32.81**	4.7
B1	3D-Adapter λ_{aug}=2	27.22	4.52	33.46	3.9
B2	3D-Adapter λ_{aug}=4	26.99	4.45	34.34	3.2
B3	3D-Adapter λ_{aug}=8	25.47	4.28	39.36	25.3
C0	B0 w/o feedback	27.18	**4.55**	33.13	7.6
C1	B0 w/o bias canceling	25.49	4.36	42.20	3.6

Table 2: Text-to-3D: comparison with previous SOTAs.

Method	CLIP↑	Aesthetic↑
Shap-E	19.4	4.07
LGM	22.5	4.31
Instant3D	25.5	4.24
MVDream-SDS	26.9	4.49
GRM	26.6	4.54
3D-Adapter (ours)	27.7	4.61
3D-Adapter + tex refine (ours)	**28.0**	**4.71**

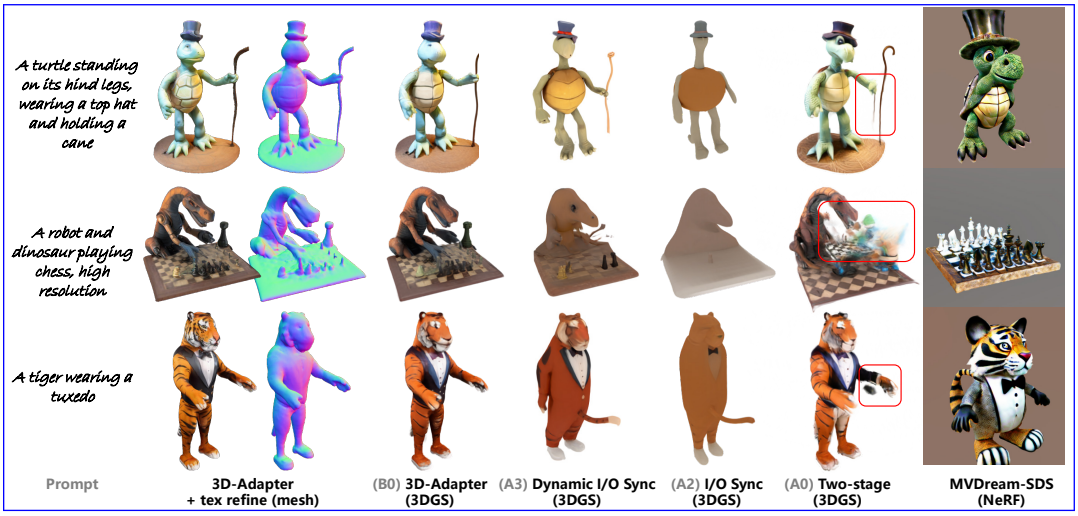

Figure 3: Comparison on text-to-3D generation. Both 3D-Adapter and I/O sync fix the broken geometry and floaters present in the two-stage method, but I/O sync suffers from blurriness.

- **CLIP similarity** (Radford et al., 2021), **LPIPS** (Zhang et al., 2018), **SSIM** (Wang et al., 2004), **PSNR**: Evaluates novel view fidelity in image-to-3D.
- **Mean depth distortion (MDD)** (Yu et al., 2024; Huang et al., 2024): Assesses the geometric quality of generated 3DGS. Lower depth distortion indicates less floaters or fuzzy surfaces, reflecting better geometry consistency. More details can be found in Appendix D.
- **CLIP t-less score**: Assesses the geometric quality of generated meshes by computing the CLIP score between shaded textureless renderings and texts appended with "textureless 3D model".

5.2 Text-to-3D Generation

For text-to-3D generation, we adopt the GRM-based 3D-Adapter with Instant3D U-Net as the base model. All results are generated using EDM Euler ancestral solver (Karras et al., 2022) with 30 denoising steps and mean latent initialization (Appendix B.2). The inference time is around 0.7 sec per step, and detailed inference time analysis is presented in Appendix B.3. For evaluation, we first compare 3D-Adapter with the baselines and conduct ablation studies on a validation set of 379 BLIP-captioned objects sampled from a high-quality subset of Objaverse (Li et al., 2022; Deitke et al., 2023). The results are shown in Table 1, with the rendered images from the validation set used as real samples when computing the FID metric. Subsequently, we benchmark 3D-Adapter on the same test set as GRM (Xu et al., 2024a), consisting of 200 prompts, to make fair comparisons to the previous SOTAs in Table 2. Qualitative results are shown in Fig. 3.

Baselines. The two-stage GRM (A0) exhibits good visual quality, but the MDD metric is magnitudes higher than that of our 3D-Adapter (B0–B3) due to the highly ambiguous geometry caused by local misalignment. Naively rewiring it into an I/O sync model (A1) worsens the results, as the

Table 3: Image-to-3D: comparison with previous SOTAs.

Method	Type	PSNR↑	SSIM↑	LPIPS↓	CLIP sim↑	FID↓
One-2-3-45	Mesh	17.84	0.800	0.199	0.832	89.4
TriplaneGaussian	GS	16.81	0.797	0.257	0.840	52.6
Shap-E	Mesh	15.45	0.772	0.297	0.854	56.5
LGM	GS	16.90	0.819	0.235	0.855	42.1
DreamGaussian	Mesh	19.19	0.811	0.171	0.862	57.6
Wonder3D	Mesh	17.29	0.815	0.240	0.871	55.7
One-2-3-45++	Mesh	17.79	0.819	0.219	0.886	42.1
GRM	GS	20.10	0.826	0.136	0.932	27.4
3D-Adapter (ours)	GS	**20.38**	**0.840**	**0.135**	**0.936**	**20.2**
3D-Adapter + TSDF (ours)	Mesh	20.34	**0.840**	**0.135**	0.933	21.7

Table 4: Text-to-avatar: comparison with baselines.

Methods	CLIP↑	Aesthetic↑	CLIP t-less↑
Two-stage baseline	23.90	4.79	24.60
I/O sync baseline	22.01	4.53	25.98
3D-Adapter	23.67	4.97	**26.07**
3D-Adapter + tex refine	**24.07**	**5.11**	**26.07**

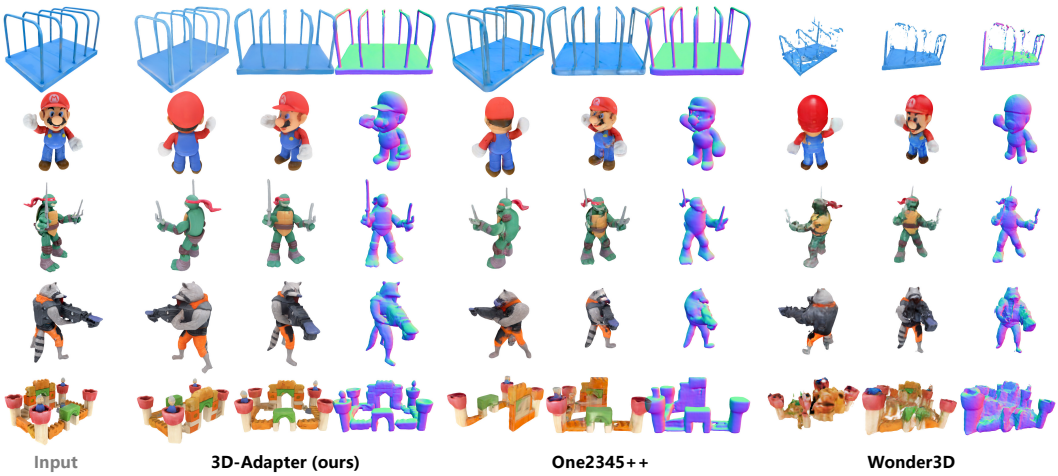

Figure 4: Comparison of mesh-based image-to-3D methods on the GSO test set.

original GRM is trained only on rendered ground truths $\boldsymbol{x}$ and cannot handle the imperfections of the denoised views $\hat{\boldsymbol{x}}$. However, when using the GRM model fine-tuned according to our method (Eq. 3), the model (A2) achieves the lowest possible MDD with nearly perfect geometry consistency, but it suffers significantly from mode collapse and struggles to generate meaningful content. Adopting a dynamic blending technique (Appendix A.2) for I/O sync (A3) alleviates this issue, but the appearance metrics are still worse than two-stage (A0).

Parameter sweep on λ_{aug} and ablation studies. The 3D-Adapter with a feedback augmentation guidance scale $\lambda_{\text{aug}} = 1$ (B0) achieves the best visual quality among all variants and significantly better geometry quality than A0. As λ_{aug} increases, the MDD metric continues to improve, but at the expense of visual quality. A very large λ_{aug} (B3) unsurprisingly worsens the results, similar to a large CFG scale. Disabling feedback augmentation (C0, equivalent to $\lambda_{\text{aug}} = 0$) notably impacts the geometric quality, as evidenced by the worse MDD metric, although it still outperforms the baseline (A0) thanks to our robust GRM fine-tuning. Additionally, we ablated the bias canceling technique (C1), observing significant degradation in all visual metrics, which substantiates the effectiveness of 3D feedback guidance (Eq. (4)). Qualitative results are all presented in Fig. 9.

Comparison with other competitors. Built on top of GRM, our 3D-Adapter (λ_{aug}=1) further advances the benchmark, outperforming previous SOTAs in text-to-3D (Jun & Nichol, 2023; Tang et al., 2024a; Li et al., 2024; Shi et al., 2024; Xu et al., 2024a) as shown in Table 2 and Fig. 3.

5.3 Image-to-3D Generation

For image-to-3D generation, we adopt the same approach used for text-to-3D generation, except for employing Zero123++ U-Net as the base model. We follow the same evaluation protocol as in Xu et al. (2024a), using 248 GSO objects (Downs et al., 2022) as the test set. As shown in Table 3, 3D-Adapter (λ_{aug}=1) outperforms the two-stage GRM and other competitors (Liu et al., 2023a; Zou

Table 5: Text-to-texture: comparison with baselines.

Methods	CLIP↑	Aesthetic↑
Two-stage baseline	25.82	**4.85**
I/O sync baseline	26.05	4.68
3D-Adapter	26.40	**4.85**
3D-Adapter + I/O sync	**26.41**	4.61

Table 6: Text-to-texture: comparison with previous SOTAs.

Methods	CLIP↑	Aesthetic↑
TexPainter	25.36	4.55
TEXTure	25.39	4.66
Text2Tex	24.44	4.72
SyncMVD	25.65	4.76
3D-Adapter (ours)	**26.40**	**4.85**

Figure 5: Comparison on text-to-texture generation.

et al., 2024; Jun & Nichol, 2023; Tang et al., 2024a;b; Long et al., 2024; Liu et al., 2024a; Xu et al., 2024a) on all metrics. Moreover, the quality loss in converting the generated 3DGS to mesh via TSDF is almost negligible. We present qualitative comparisons of the meshes generated by 3D-Adapter and other methods in Fig. 4.

5.4 Text-to-Texture Generation

For text-to-texture evaluation, 3D-Adapter employs fast texture backprojection to blend multiple views for intermediate timesteps, and switches to high-quality texture field optimization (similar to NeRF) for the final timestep. A community Stable Diffusion v1.5 variant, DreamShaper 8, is adopted as the base model. During the sampling process, 32 surrounding views are used initially, and this number is gradually reduced to 7 views during the denoising process to reduce computation in later stages. We adopt the EDM Euler ancestral solver with 24 denoising steps. The total inference time is around 1.5 minutes per object on a single RTX A6000 GPU (including UV unwrapping), which is faster than the competitors (SyncMVD (Liu et al., 2023c): ∼1.9 min, TEXTure (Richardson et al., 2023): ∼2.0 min, Text2Tex (Chen et al., 2023a): ∼11.2 min, TexPainter (Zhang et al., 2024): ∼11.6 min). 92 BLIP-captioned objects are sampled from a high-quality subset of Objaverse as our test set.

Comparison with baselines. As shown in Table 5 and Fig. 5, the two-stage baseline has good texture details but notably worse CLIP score due to poor consistency. The I/O sync baseline has much better consistency, but it sacrifices details, resulting in the worst aesthetic score. In comparison, 3D-Adapter excels in both metrics, producing detailed and consistent textures. Additionally, we demonstrate that 3D-Adapter and I/O sync should not be used simultaneously, as I/O sync consistently compromises texture details, as evidenced by the Aesthetic score and qualitative results.

Figure 6: Comparison on text-to-avatar generation using the same pose template.

Comparison with other competitors. Quantitatively, Table 6 demonstrates that 3D-Adapter significantly outperforms previous SOTAs on both metrics. Interestingly, even our two-stage baseline in Table 5 surpasses the competitors, which can be attributed to our use of texture field optimization and community-customized base model. Qualitative results in Fig. 5 reveal that previous methods are generally less robust compared to 3D-Adapter and may produce artifacts in some cases.

5.5 Text-to-Avatar Generation

For text-to-avatar generation, the optimization-based 3D-Adapter is adopted with a custom pose ControlNet for Stable Diffusion v1.5, which provides extra conditioning given a human pose template. 32 full-body views and 32 upper-body views are selected for denoising, capturing both the overall figure and face details. These are later reduced to 12 views during the denoising process. We use the EDM Euler ancestral solver with 32 denoising steps, with an inference time of approximately 7 minutes per object on a single RTX A6000 GPU (including 0.5 minutes on CPU-based UV unwrapping). Texture editing (using text-to-texture pipeline and SDEdit with $t_{\text{init}} = 0.3T$) and refinement can be optionally applied to further improve texture details, which costs 1.4 minutes. For evaluation, we compare 3D-Adapter with baselines using 21 character prompts on the same pose template. As shown in Table 4, 3D-Adapter achieves the highest scores across all three metrics, indicating superior appearance and geometry. Fig. 6 reveals that I/O sync produces overly smoothed texture and geometry due to mode collapse, while the two-stage baseline results in noisy, less coherent texture and geometry. These observations also align with the quantitative results in Table 4.

6 Conclusion

In this work, we have introduced 3D-Adapter, a plug-in module that effectively enhances the 3D geometry consistency of existing multi-view diffusion models, bridging the gap between high-quality 2D and 3D content creation. We have demonstrated two variants of 3D-Adapter: the fast 3D-Adapter using feed-forward Gaussian reconstruction, and the flexible training-free 3D-Adapter using 3D optimization and pretrained ControlNets. Experiments on text-to-3D, image-to-3D, text-to-texture, and text-to-avatar tasks have substantiated its all-round competence, suggesting great generality and potential in future extension.

Limitations. 3D-Adapter introduces substantial computation overhead, primarily due to the VAE decoding process before 3D reconstruction. In addition, we observe that our finetuned ControlNet for 3D feedback augmentation strongly overfits the finetuning data, which may limit its generalization despite the proposed guidance method. Future work may focus on developing more efficient, easy-to-finetune networks for 3D-Adapter.

REFERENCES

Titas Anciukevicius, Zexiang Xu, Matthew Fisher, Paul Henderson, Hakan Bilen, Niloy J. Mitra, and Paul Guerrero. RenderDiffusion: Image diffusion for 3D reconstruction, inpainting and generation. In *CVPR*, 2023.

Miguel Angel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, Afshin Dehghan, and Josh Susskind. Gaudi: A neural architect for immersive 3d scene generation. In *NeurIPS*, 2022.

Arwen Bradley and Preetum Nakkiran. Classifier-free guidance is a predictor-corrector, 2024. URL `https://arxiv.org/abs/2408.09000`.

Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *CVPR*, 2022.

Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *ICCV*, 2023a.

Hansheng Chen, Jiatao Gu, Anpei Chen, Wei Tian, Zhuowen Tu, Lingjie Liu, and Hao Su. Single-stage diffusion nerf: A unified approach to 3d generation and reconstruction. In *ICCV*, 2023b.

Zilong Chen, Yikai Wang, Feng Wang, Zhengyi Wang, and Huaping Liu. V3d: Video diffusion models are effective 3d generators, 2024.

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.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *ICLR*, 2022.

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*, pp. 2553–2560, 2022.

Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In *ICML*, 2022.

Chenjian Gao, Boyan Jiang, Xinghui Li, Yingpeng Zhang, and Qian Yu. Genesistex: Adapting image denoising diffusion to texture space. In *CVPR*, 2024.

Jiatao Gu, Alex Trevithick, Kai-En Lin, Josh Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion. In *ICML*, 2023.

Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation, 2023.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields, 2024.

Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *CVPR*, 2022.

Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions, 2023.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *CVPR*, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Min Seok Lee, Wooseok Shin, and Sung Won Han. Tracer: Extreme attention guided salient object tracing network. In *AAAI*, 2022.

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. URL `https://openreview.net/forum?id=2lDQLiH1W4`.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

Yangming Li and Mihaela van der Schaar. On error propagation of diffusion models. In *ICLR*, 2024. URL `https://openreview.net/forum?id=RtAct1E2zS`.

Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Zexiang Xu, Hao Su, et al. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. In *NeurIPS*, 2023a.

Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *CVPR*, 2024a.

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*, 2023b.

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*, 2024b.

Yuxin Liu, Minshan Xie, Hanyuan Liu, and Tien-Tsin Wong. Text-guided texturing by synchronized multi-view diffusion, 2023c.

Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *ECCV*, 2022.

Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, and Wenping Wang. Wonder3d: Single image to 3d using cross-domain diffusion. In *CVPR*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *NeurIPS*, 2022.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022.

Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *CVPR*, 2023.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *CVPR*, 2023.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15, July 2022. doi: 10.1145/3528223.3530127. URL https://doi.org/10.1145/3528223.3530127.

Ryan Po, Wang Yifan, Vladislav Golyanik, Kfir Aberman, Jonathan T. Barron, Amit H. Bermano, Eric Ryan Chan, Tali Dekel, Aleksander Holynski, Angjoo Kanazawa, C. Karen Liu, Lingjie Liu, Ben Mildenhall, Matthias Nießner, Björn Ommer, Christian Theobalt, Peter Wonka, and Gordon Wetzstein. State of the art on diffusion models for visual computing. In *Eurographics STAR*, 2024.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023.

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pp. 8748–8763, 2021.

Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *SIGGRAPH*, 2023.

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

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models. In *NeurIPS Workshop*, 2022.

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.

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, 2023.

Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *ICLR*, 2024.

J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *CVPR*, 2023.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pp. 175–184, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 3905673134. doi: 10.1145/1057432.1057456. URL https://doi.org/10.1145/1057432.1057456.

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, 2024a.

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *ICLR*, 2024b.

Ayush Tewari, Tianwei Yin, George Cazenavette, Semon Rezchikov, Joshua B. Tenenbaum, Frédo Durand, William T. Freeman, and Vincent Sitzmann. Diffusion with forward models: Solving stochastic inverse problems without direct supervision. In *NeurIPS*, 2023.

Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitrii Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. SV3D: Novel multi-view synthesis and 3D generation from a single image using latent video diffusion. *arXiv*, 2024.

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. In *NeurIPS*, pp. 27171–27183, 2021a.

Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. PF-LRM: Pose-free large reconstruction model for joint pose and shape prediction. In *ICLR*, 2024. URL `https://openreview.net/forum?id=noe76eRcPC`.

Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, and Baining Guo. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *CVPR*, 2023.

Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *ICCV Workshop*, 2021b.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. In *ICLR*, 2023.

Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. Gpt-4v(ision) is a human-aligned evaluator for text-to-3d generation. In *CVPR*, 2024.

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, 2024a.

Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. In *ICLR*, 2024b.

Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient and compact surface reconstruction in unbounded scenes, 2024.

Hongkun Zhang, Zherong Pan, Congyi Zhang, Lifeng Zhu, and Xifeng Gao. Texpainter: Generative mesh texturing with multi-view consistency. In *SIGGRAPH*, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705250. doi: 10.1145/3641519.3657494. URL `https://doi.org/10.1145/3641519.3657494`.

Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *ECCV*, 2022.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023.

Richard Zhang, Phillip Isola, Alexei Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Xin-Yang Zheng, Hao Pan, Peng-Shuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *ACM Transactions on Graphics*, 42(4), 2023.

Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *CVPR*, 2024.

Qi Zuo, Xiaodong Gu, Lingteng Qiu, Yuan Dong, Zhengyi Zhao, Weihao Yuan, Rui Peng, Siyu Zhu, Zilong Dong, Liefeng Bo, and Qixing Huang. Videomv: Consistent multi-view generation based on large video generative model, 2024.

A Details on the I/O Sync Baseline

A.1 Theoretical Analysis

When performing diffusion ODE sampling using the common Euler solver, a linear input sync operation (e.g., linear blending or optimizing using the L2 loss) is equivalent to syncing the output $\hat{\boldsymbol{x}}_t$ as well as the initialization $\boldsymbol{x}_{t_{\text{init}}}$. This is because the input $\boldsymbol{x}_t$ can be expressed as a linear combination of all previous outputs $\{\boldsymbol{x}_{t-\Delta t}, \boldsymbol{x}_{t-2\Delta t}, \dots\}$ and the initialization $\boldsymbol{x}_{t_{\text{init}}}$ by expanding the recursive Euler steps.

Furthermore, linear I/O sync is also equivalent to linear score sync, since the learned score function $\boldsymbol{s}_t(\boldsymbol{x}_t)$ can also be expressed as a linear combination of the input $\boldsymbol{x}_t$ and output $\hat{\boldsymbol{x}}_t$:

$$\boldsymbol{s}_t(\boldsymbol{x}_t) = -\frac{\hat{\boldsymbol{\epsilon}}_t}{\sigma_t} = \frac{\alpha_t \hat{\boldsymbol{x}}_t - \boldsymbol{x}_t}{{\sigma_t}^2} \tag{5}$$

However, synchronizing the score function, a.k.a. score averaging, is theoretically problematic. Let $p(\boldsymbol{x}|\boldsymbol{c}_1), p(\boldsymbol{x}|\boldsymbol{c}_2)$ be two independent probability density functions of a corresponding pixel $\boldsymbol{x}$ viewed from cameras $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, respectively. A diffusion model is trained to predict the score function $\boldsymbol{s}_t(\boldsymbol{x}_t|\boldsymbol{c}_v)$ of the noisy distribution at timestep t, defined as:

$$\boldsymbol{s}_t(\boldsymbol{x}_t|\boldsymbol{c}_v) = \nabla_{\boldsymbol{x}_t} \log \int p(\boldsymbol{x}_t|\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{c}_v) d\boldsymbol{x}, \tag{6}$$

where $p(\boldsymbol{x}_t|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}_t; \alpha_t \boldsymbol{x}, \sigma_t^2 \boldsymbol{I})$ is a Gaussian perturbation kernel. Ideally, assuming $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are independent, combining the two conditional PDFs $p(\boldsymbol{x}|\boldsymbol{c}_1)$ and $p(\boldsymbol{x}|\boldsymbol{c}_2)$ yields the product $p(\boldsymbol{x}|\boldsymbol{c}_1, \boldsymbol{c}_2) = \frac{1}{Z} p(\boldsymbol{x}|\boldsymbol{c}_1) p(\boldsymbol{x}|\boldsymbol{c}_2)$, where Z is a normalization factor. The corresponding score function should then become $\boldsymbol{s}_t(\boldsymbol{x}_t|\boldsymbol{c}_1, \boldsymbol{c}_2) = \nabla_{\boldsymbol{x}_t} \log \int p(\boldsymbol{x}_t|\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{c}_1, \boldsymbol{c}_2) d\boldsymbol{x}$. However, the average of $\boldsymbol{s}_t(\boldsymbol{x}_t|\boldsymbol{c}_1)$ and $\boldsymbol{s}_t(\boldsymbol{x}_t|\boldsymbol{c}_2)$ is generally not proportional to $\boldsymbol{s}_t(\boldsymbol{x}_t|\boldsymbol{c}_1, \boldsymbol{c}_2)$, i.e.:

$$\begin{aligned} \frac{1}{2}\boldsymbol{s}_t(\boldsymbol{x}_t|\boldsymbol{c}_1) + \frac{1}{2}\boldsymbol{s}_t(\boldsymbol{x}_t|\boldsymbol{c}_2) &= \frac{1}{2}\nabla_{\boldsymbol{x}_t} \log \left(\int p(\boldsymbol{x}_t|\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{c}_1) d\boldsymbol{x} \right) \left(\int p(\boldsymbol{x}_t|\boldsymbol{x}) p(\boldsymbol{x}|\boldsymbol{c}_2) d\boldsymbol{x} \right) \\ &\not\propto \nabla_{\boldsymbol{x}_t} \log \int p(\boldsymbol{x}_t|\boldsymbol{x}) \left(\frac{1}{Z} p(\boldsymbol{x}|\boldsymbol{c}_1) p(\boldsymbol{x}|\boldsymbol{c}_2) \right) d\boldsymbol{x} = \boldsymbol{s}_t(\boldsymbol{x}_t|\boldsymbol{c}_1, \boldsymbol{c}_2). \end{aligned} \tag{7}$$

In Fig. 7, we illustrate a simple 1D simulation, showing that score averaging leads to mode collapse, when compared to the real product distribution. This explains the blurry, mean-shaped results produced by the I/O sync baselines. This problem is also noted in a concurrent work (Bradley & Nakkiran, 2024) in the context of classifier-free guidance.

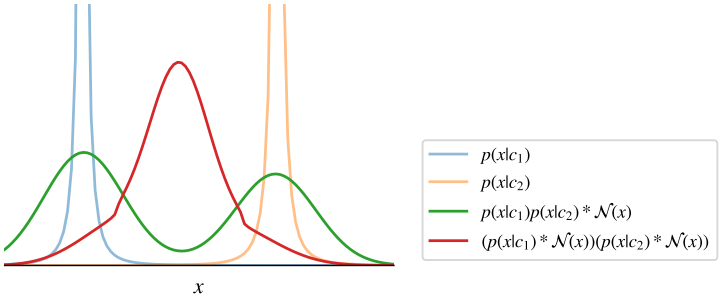

Figure 7: A simple 1D simulation illustrating the difference between the **score averaged distribution** and the actual **perturbed product distribution**. $*$ denotes convolution, and $\mathcal{N}(x)$ denotes the Gaussian perturbation kernel.

A.2 Dynamic I/O Sync

While I/O sync works reasonably on our texture generation benchmark, our text-to-3D model using I/O sync (A2 in Table 1 and Fig. 3) exhibits significant quality degradation due to mode collapse.

Table 7: GRM-based 3D-Adapter: Inference times (sec) with guidance on a single RTX A6000.

Encode	Adapter Decode	VAE Decode	GRM	Render	Adapter Encode	Decode	**Adapter total**	**Overall total**
0.055	0.120	0.215	0.091	0.023	0.082	0.121	0.531	0.707

We believe the main reasons are twofold. First, the base model Instant3D generates a very sparse set of only four views, which are hard to synchronize. Second, our finetuned GRM reconstructor is trained using the depth loss to suppress surface fuzziness, which has a negative impact when its sharp renderings $\tilde{\boldsymbol{x}}_t$ are used as diffusion output. This is because a well-trained diffusion model should actually predict blurry outputs $\hat{\boldsymbol{x}}_t$ in the early denoising stage as the mean of the distribution $p(\boldsymbol{x}_0|\boldsymbol{x}_t)$. Only in the late stage should $\hat{\boldsymbol{x}}_t$ be sharp and crisp, as shown in Fig. 8.

To make the I/O sync baseline more competitive on the text-to-3D benchmark, we adopt a simple technique called *dynamic blending* or dynamic I/O sync. The idea is that, since I/O sync mainly corrupts fine-grained details, its influence should be reduced during the late denoising stages when details are being generated. Therefore, we perform a weighted blending of the denoised views before synchronization $\hat{\boldsymbol{x}}_t$ and the rendered views $\tilde{\boldsymbol{x}}_t$:

$$\tilde{\boldsymbol{x}}_t^{\text{blend}} = (1 - \lambda_t^{\text{sync}})\hat{\boldsymbol{x}}_t + \lambda_t^{\text{sync}}\tilde{\boldsymbol{x}}_t, \tag{8}$$

where λ_t^{sync} is a time-dependent blending weight, and $\tilde{\boldsymbol{x}}_t^{\text{blend}}$ is the blended output that is fed to the diffusion solver. We set $\lambda_t^{\text{sync}} = \frac{1-\alpha_t}{\sqrt{\alpha_t^2+\sigma_t^2}}$, so that λ_t^{sync} decreases over the denoising process.

As shown in Table 1 and Fig. 3, dynamic I/O sync demonstrates significant improvements in visual quality over vanilla I/O sync. Its MDD score indicates that its geometry consistency lies between vanilla I/O sync and 3D-Adapter. However, the visual quality of dynamic I/O sync is still clearly below that of the two-stage method and 3D-Adapter. While it is possible to tune a better blending weight λ_t^{sync}, we believe it is very difficult to reduce the gap due to the aforementioned challenges brought by our model setup.

B Details on GRM-Based 3D-Adapter

B.1 ControlNet

The GRM-based 3D-Adapter trains a ControlNet (Zhang et al., 2023) for feedback augmentation, which has very large model capacity and can easily overfit our relatively small finetuning dataset (e.g., 47k objects for Instant3D). Therefore, using the CFG-like bias subtraction technique (Eq. (4)) is extremely important to the generalization performance, which is already validated in our ablation studies. Additionally, we disconnect the text prompt input from the ControlNet to further alleviate overfitting.

B.2 Mean Latent Initialization

Instant3D's 4-view UNet is sensitive to the initialization method, as noted in the original paper (Li et al., 2024), which develops an empirical Gaussian blob initialization method to stabilize the background color. In contrast, this paper adopts a more principled mean latent initialization method by computing the mean value $\bar{\boldsymbol{x}}$ of the VAE-encoded latents of 10K objects in the training set. The initial state is then sampled by perturbing the mean latent with Gaussian noise ϵ:

$$\boldsymbol{x}_{t_{\text{init}}} = \alpha_{t_{\text{init}}}\bar{\boldsymbol{x}} + \sigma_{t_{\text{init}}}\boldsymbol{\epsilon}. \tag{9}$$

B.3 Inference Time

Detailed module-level inference times per denoising step is shown in Table 7 (with classifier-free guidance and guided 3D feedback augmentation enabled). Apparently, the SDXL VAE decoder is the most expensive module within 3D-Adapter, which may be replaced by a more efficient decoder in future work.

C DETAILS ON OPTIMIZATION-BASED 3D-ADAPTER

The optimization-based 3D-Adapter faces the challenge of potentially inconsistent multi-view inputs, especially at the early denoising stage. Existing surface optimization approaches, such as NeuS (Wang et al., 2021a), are not designed to address the inconsistency. Therefore, we have developed various techniques for the robust optimization of InstantNGP NeRF (Müller et al., 2022) and DMTet mesh (Shen et al., 2021), using enhanced regularization and progressive resolution.

Rendering. For each NeRF optimization iteration, we randomly sample a 128×128 image patch from all camera views. Unlike Poole et al. (2023) that computes the normal from NeRF density gradients, we compute patch-wise normal maps from the rendered depth maps, which we find to be faster and more robust. For mesh rendering, we obtain the surface color by querying the same InstantNGP neural field used in NeRF. For both NeRF and mesh, Lambertian shading is applied in the linear color space prior to tonemapping, with random point lights assigned to their respective views.

RGBA losses. For both NeRF and mesh, we employ RGB and alpha rendering losses to optimize the 3D parameters so that the rendered views $\tilde{\boldsymbol{x}}_t$ match the intermediate denoised views $\hat{\boldsymbol{x}}'_t$. For RGB, we employ a combination of pixel-wise L1 loss and patch-wise LPIPS loss (Zhang et al., 2018). For alpha, we predict the target alpha channel from $\hat{\boldsymbol{x}}'_t$ using an off-the-shelf background removal network (Lee et al., 2022) as in Magic123 (Qian et al., 2024). Additionally, we soften the predicted alpha map using Gaussian blur to prevent NeRF from overfitting the initialization.

Normal losses. To avoid bumpy surfaces, we apply an L1.5 total variation (TV) regularization loss on the rendered normal maps:

$$\mathcal{L}_{\text{N}} = \sum_{chw} \left\| w_{hw} \cdot \nabla_{hw} n^{\text{rend}}_{chw} \right\|^{1.5}, \tag{10}$$

where $n^{\text{rend}}_{chw} \in \mathbb{R}$ denotes the value of the $C \times H \times W$ normal map at index (c, h, w), $\nabla_{hw} n^{\text{rend}}_{chw} \in \mathbb{R}^2$ is the gradient of the normal map w.r.t. (h, w), and $w_{hw} \in [0, 1]$ is the value of a foreground mask with edge erosion.

Ray entropy loss for NeRF. To mitigate fuzzy NeRF geometry, we propose a novel ray entropy loss based on the probability of sample contribution. Unlike previous works (Kim et al., 2022; Metzer et al., 2023) that compute the entropy of opacity distribution or alpha map, we consider the ray density function:

$$p(\tau) = T(\tau)\sigma(\tau), \tag{11}$$

where τ denotes the distance, $\sigma(\tau)$ is the volumetric density and $T(\tau) = \exp - \int_0^s \sigma(\tau) d\tau$ is the ray transmittance. The integral of $p(\tau)$ equals the alpha value of the pixel, i.e., $a = \int_0^{+\inf} p(\tau) d\tau$, which is less than 1. Therefore, the background probability is $1 - a$ and a corresponding correction term needs to be added when computing the continuous entropy of the ray as the loss function:

$$\mathcal{L}_{\text{ray}} = \sum_r \int_0^{+\inf} -p_r(\tau) \log p_r(\tau) d\tau - \underbrace{(1 - a_r) \log \frac{1 - a_r}{d}}_{\text{background correction}}, \tag{12}$$

where r is the ray index, and d is a user-defined "thickness" of an imaginative background shell, which can be adjusted to balance foreground-to-background ratio.

Mesh smoothing losses As per common practice, we employ the Laplacian smoothing loss (Sorkine et al., 2004) and normal consistency loss to further regularize the mesh extracted from DMTet.

Implementation details The weighted sum of the aforementioned loss functions is utilized to optimize the 3D representation. At each denoising step, we carry forward the 3D representation from the previous step and perform additional iterations of Adam (Kingma & Ba, 2015) optimization. During the denoising sampling process, the rendering resolution progressively increases from 128 to 256, and finally to 512 when NeRF is converted into a mesh (for texture generation the resolution is consistently 512). When the rendering resolution is lower than the diffusion resolution 512, we employ RealESRGAN-small (Wang et al., 2021b) for efficient super-resolution.

D Details on the Mean Depth Distortion (MDD) Metric

The MDD metric is inspired by the depth distortion loss in Yu et al. (2024), which proves effective in removing floaters and improving the geometry quality. The depth distortion loss of a pixel is defined as:

$$\mathcal{L}_{\mathrm{D}} = \sum_{m,n} \omega_m \omega_n |\tau_m - \tau_n|, \tag{13}$$

where m, n index over Gaussians contributing to the ray, ω_m is the blending weight of the m-th Gaussian and τ_m is the distance of the intersection point.

To compute the mean depth distortion of a view, we take the sum of depth distortion losses across all pixels and divide it by the sum of alpha values across all pixels:

$$MDD = \frac{\sum_r \mathcal{L}_{\mathrm{D}r}}{\sum_r a_r}, \tag{14}$$

where r is the pixel index.

E More Results

We present more qualitative comparisons in Fig. 8, 9, 10, 11, 12, 13, 14.

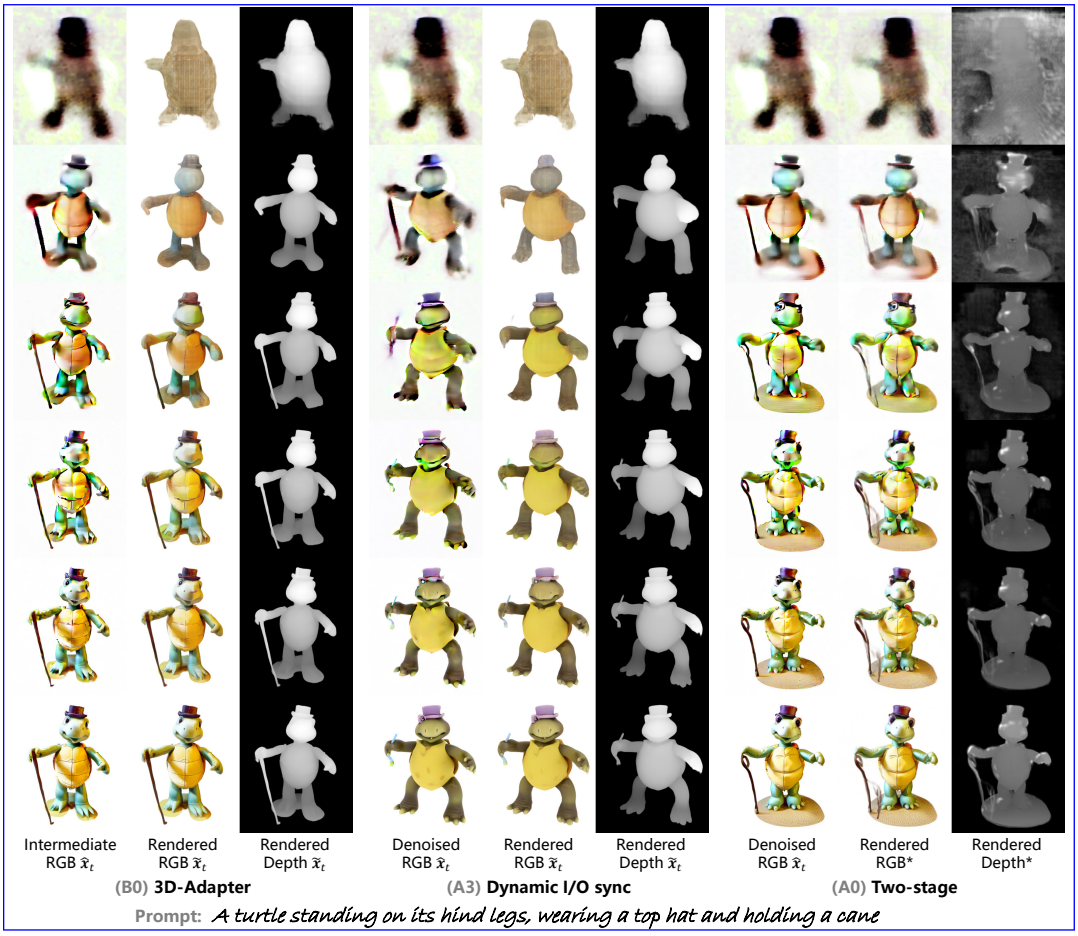

Figure 8: Text-to-3D: visualization of the multi-step sampling process. *For the two-stage method, the rendered RGB and depth maps (using the original GRM reconstructor before finetuning) are NOT a part of the sampling process, and are presented here solely for visualization.

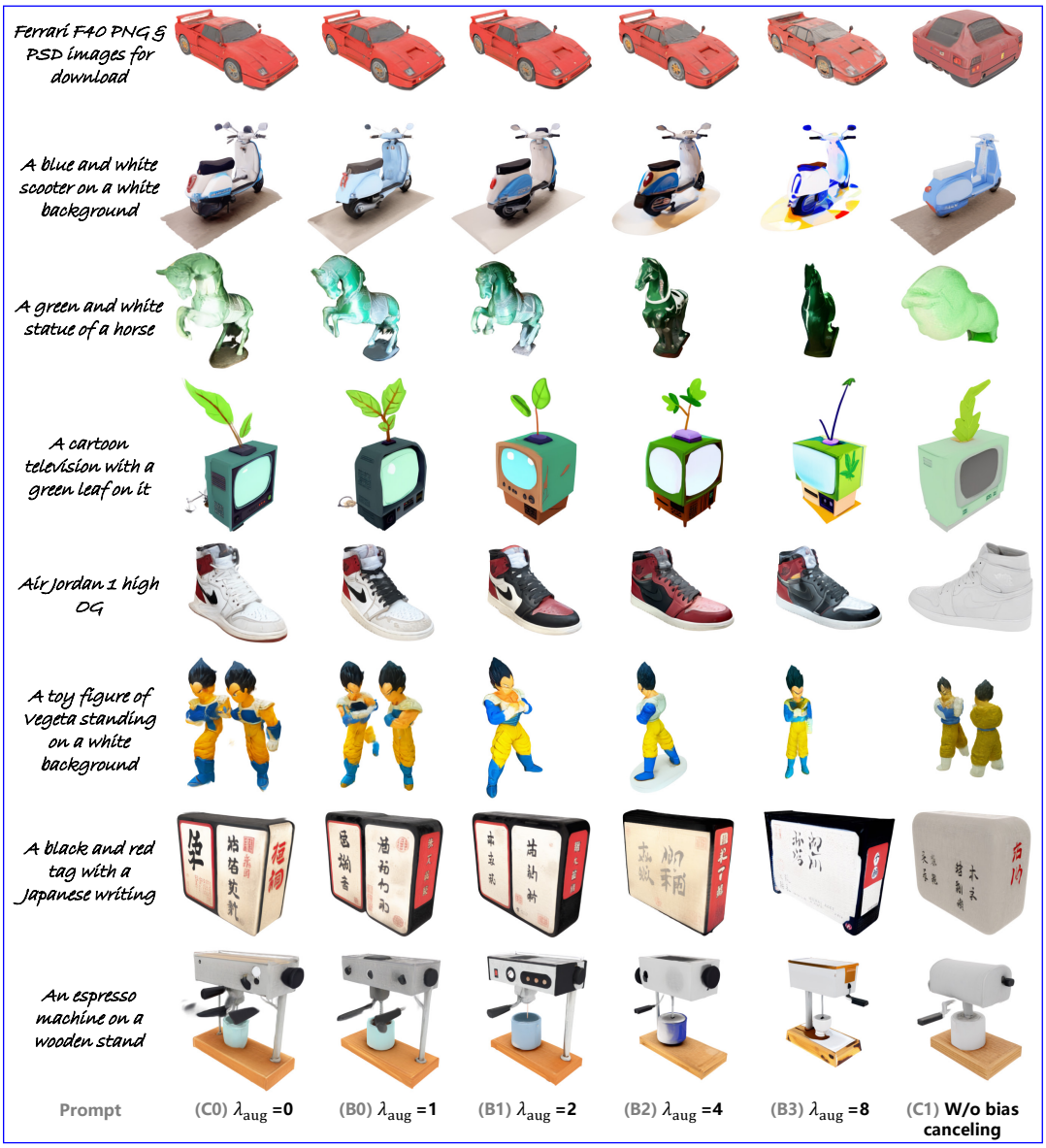

Figure 9: Text-to-3D: qualitative results from the parameter sweep on λ_{aug} and the ablation studies.

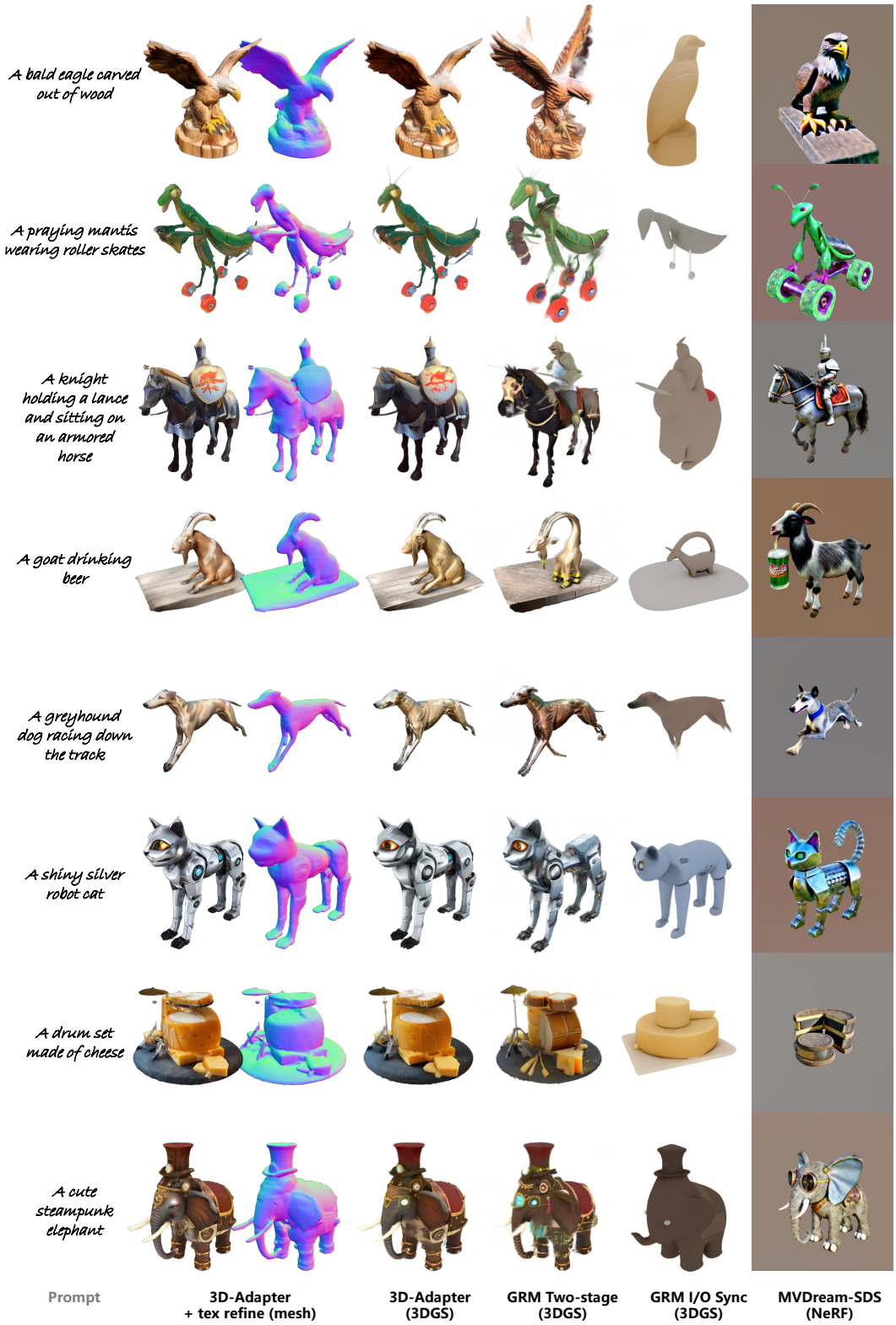

Figure 10: More comparisons on text-to-3D generation (part 1).

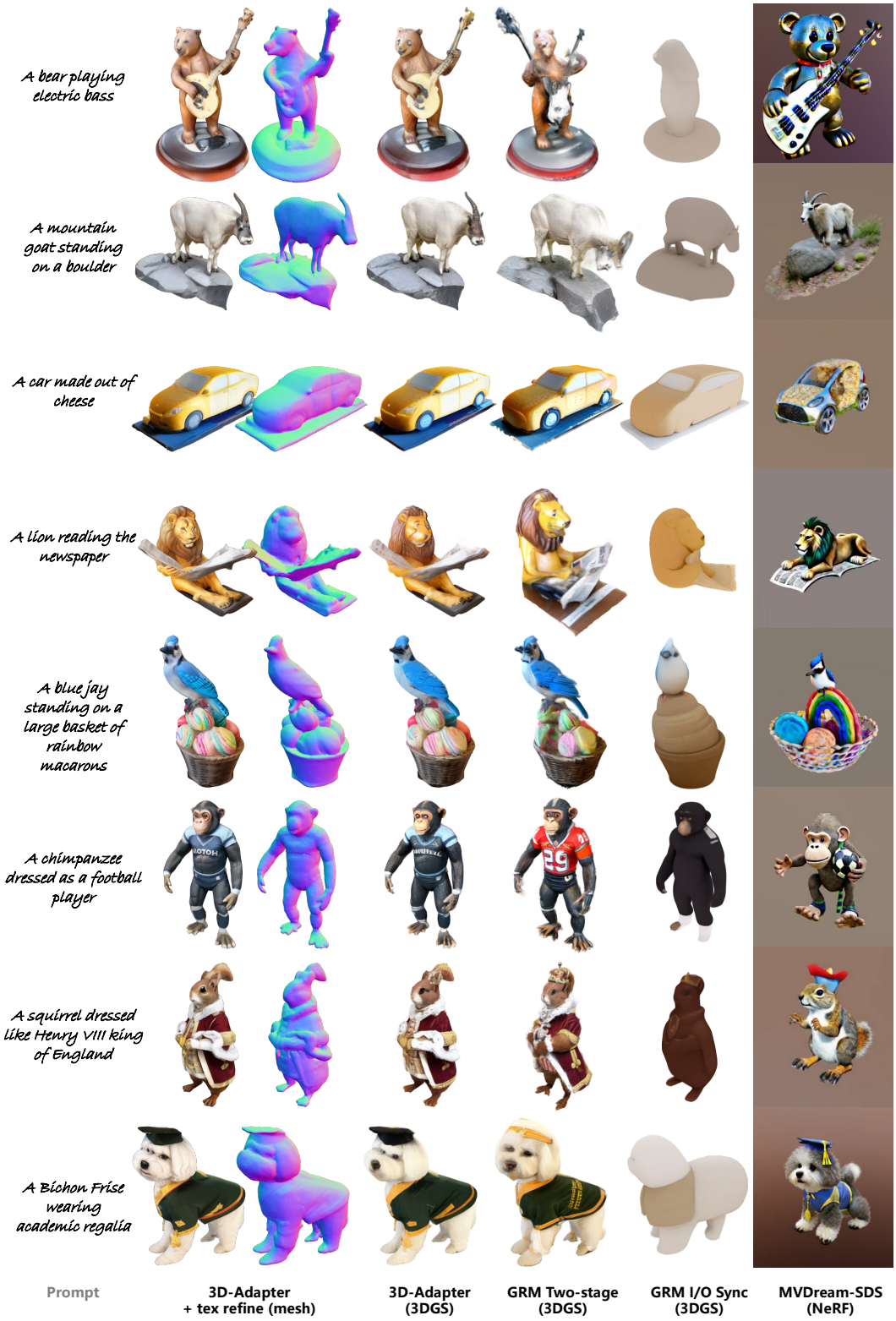

Figure 11: More comparisons on text-to-3D generation (part 2).

Figure 12: More comparisons on image-to-3D generation.

Figure 13: More comparisons on text-to-texture generation.

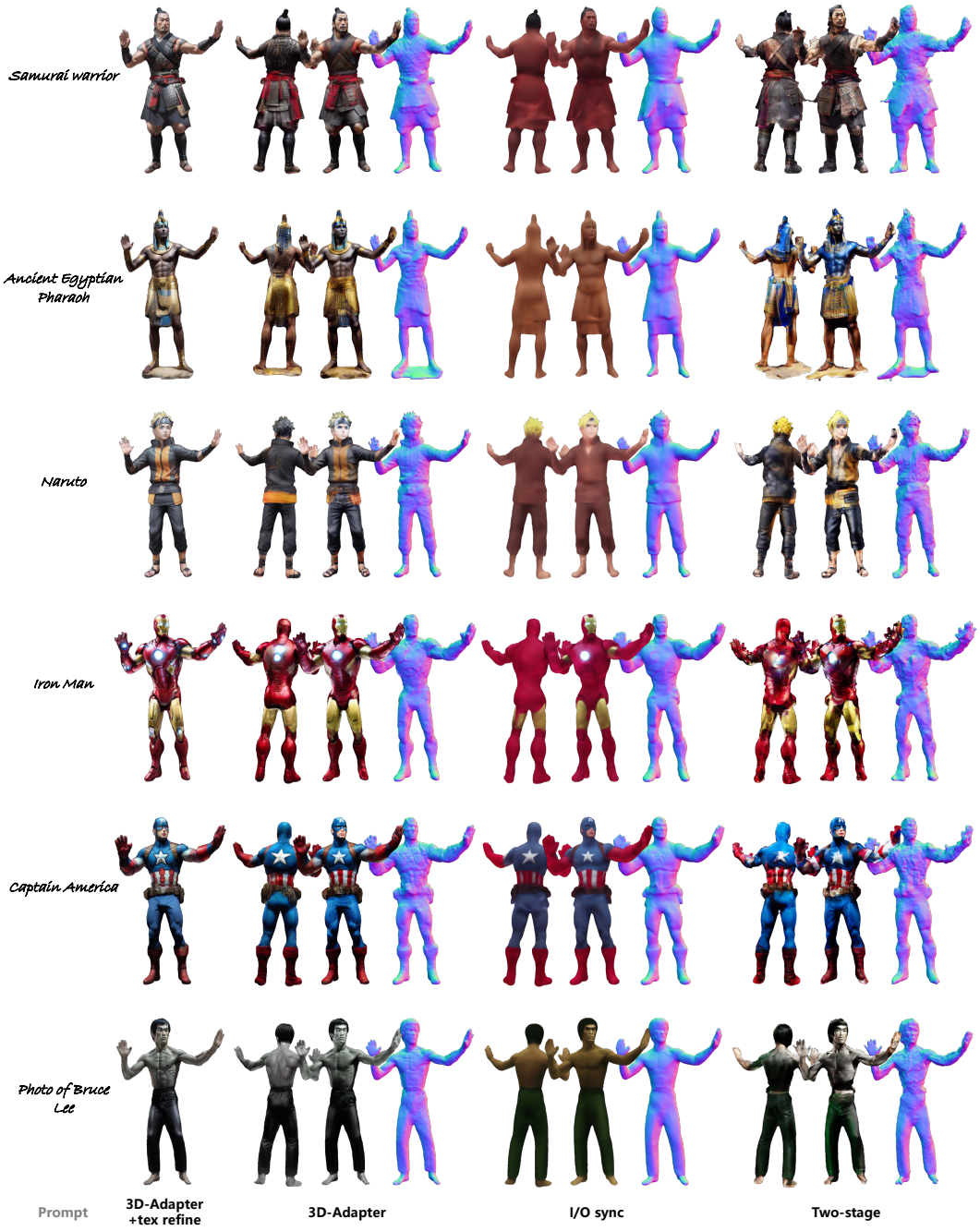

Figure 14: More comparisons on text-to-avatar generation.