Text-to-3D Generation with Bidirectional Diffusion using both 2D and 3D priors

Supplementary Material

In the supplementary material, we first introduce the data processing pipeline in (§ 5.1), then provide more implementation details of the model architecture (§ 5.2), more training details in (§ 5.3), and give more ablation results in (§ 5.4).

5.1. Data Processing

As mentioned in the main paper, we use 6k ShapNet-Chair [1] and LVIS Objaverse 40k [4] as our training datasets. We obtain the Objaverse 40k dataset by filtering objects with LVIS category labels in the 800k Objaverse data. To process data for the 2D diffusion process, we use Blender to render each 3D object into 8 images with a fixed elevation of 30° and evenly distributed azimuth from -180° to 180°. These fixed view images serve as the ground truth multi-view image set $\mathcal{V}$. In addition, we also randomly render 16 views to supervise the novel view rendering of the denoised radiance field $\mathcal{F}_0'$. All the images are rendered at a resolution of 256×256. Since we adopt the DeepFloyd as our 2D foundation model which runs at a resolution of 64×64, the rendered images are downsampled to 64×64 during training. To process data for the 3D diffusion, we compute the signed distance of each 3D object at each $N \times N \times N$ grid point within a $[-1, 1]$ cube, where N is set to 128 in our experiments. To obtain the latent code $\mathcal{C}$ for each object, we use the encoder in Shap-E [12] to encode each object and apply $t_0 = 0.4$ level Gaussian noise to $\mathcal{C}$ to get noisy $\mathcal{C}_{t_0}$, and then decode the condition radiance field during training.

Furthermore, both the ShapNet-Chair and Objaverse dataset contains no text prompts, so we use Blip-2 [14] to generate labels for the Objaverse object by rendering the image from a positive view. For evaluation, we manually choose 50 text prompts from the Objaverse dataset without LVIS label, ensuring the text prompts have not been trained during training.

5.2. Model Architecture Details

Our framework contains a 3D denoising network built upon 3D SparseConv U-Net and a 2D denoising network built upon 2D U-Net. Below we provide more details for each of them.

5.2.1 3D Denoising Network

Given the input feature volume

$$\begin{aligned}\mathcal{S}_{\text{in}} = \text{Concat}(\mathcal{M}, \text{Sp3DConv}(\mathcal{N}),\\ \text{Sp3DConv}(\mathcal{G}_{t_0}))\end{aligned} \tag{8}$$

as discussed in Section 3.2 of the main paper, we use a 3D sparse U-Net $\mathcal{U}$ to denoise the signed distance field. Specifically, we first use a $1\times1\times1$ convolution to adjust the number of input channels to 128. Then we stack four $3\times3\times3$ sparse 3D convolution blocks to extract hierarchical features while obtaining downsampled $8 \times 8 \times 8$ feature grids. It is noteworthy that we inject the timestep and text embeddings into each sparse convolution block to make the network aware of the current noise level and text information. In practice, we first use an MLP to project the scalar timestep t to high-dimensional features and fuse it with the text embeddings with another MLP to get the fused embeddings as follows:

$$\text{emb} = \text{MLP}_2(\text{Concat}(\text{emb}_{\text{text}}, \text{MLP}_1(t))), \tag{9}$$

where emb_{text} denotes the text embeddings. Then in each sparse convolution block, we project the fused embeddings to scale β and shift γ:

$$\beta, \gamma = \text{Chunk}(\text{MLP}_{\text{proj}}(\text{GeLU}(\text{emb}))), \tag{10}$$

where GeLU is activated function, Chunk operation splits the projected features into two equal parts along the channel dimension. After that, we introduce modulation to the sparse convolution by:

$$\mathcal{S}_{k+1} = (1+\beta)(\text{SparseConv}(\text{GroupNorm}(\mathcal{S}_k)))+\gamma, \tag{11}$$

where k denotes the feature level, $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ are the input and output of the k-th level sparse convolution block. Subsequently, we use 4 sparse deconvolution blocks to upsample the bottleneck feature grids with residuals linked from the extracted hierarchical features:

$$\mathcal{S}_k' = \text{SparseDeConv}(\mathcal{S}_{k+1}') + \mathcal{S}_k, \tag{12}$$

where $\mathcal{S}_{k+1}'$ and $\mathcal{S}_k'$ are the input and output of the k-th level sparse de-convolution block, and obtain the output features $\mathcal{S}$ of the 3D U-Net.

To obtain the denoised signed distance field, we first query each 3D position p in the fused feature grid $\mathcal{S}$ to fetch its feature $\mathcal{S}(p)$ by Trilinear Interpolation. Then we apply several MLPs (we adopt the ResNetFC blocks in [44]) to predict the signed distance at position p:

$$\mathcal{F}_0' = \text{MLP}(\mathcal{S}(p), \lambda(p)), \tag{13}$$

where $\lambda(p)$ is the positional encoding:

$$\begin{aligned}\lambda(p) =&(\sin(2^0\omega p), \cos(2^0\omega p), \sin(2^1\omega p), \cos(2^1\omega p),\\ &..., \sin(2^{L-1}\omega p), \cos(2^{L-1}\omega p)).\end{aligned} \tag{14}$$

L is set to 6 in all experiments.

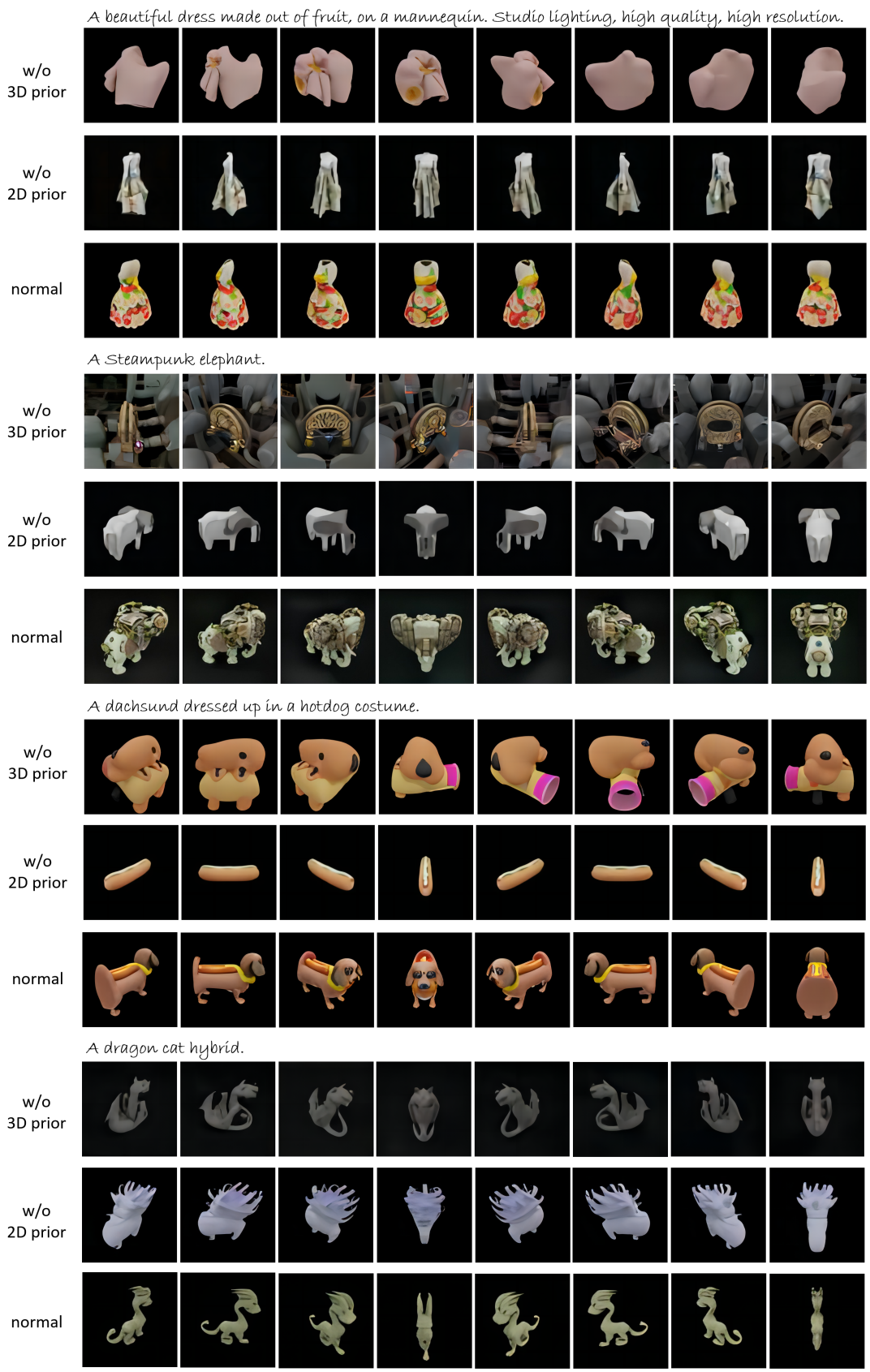

Figure 7. More ablation results showing the importance of both 2D and 3D priors in our model.

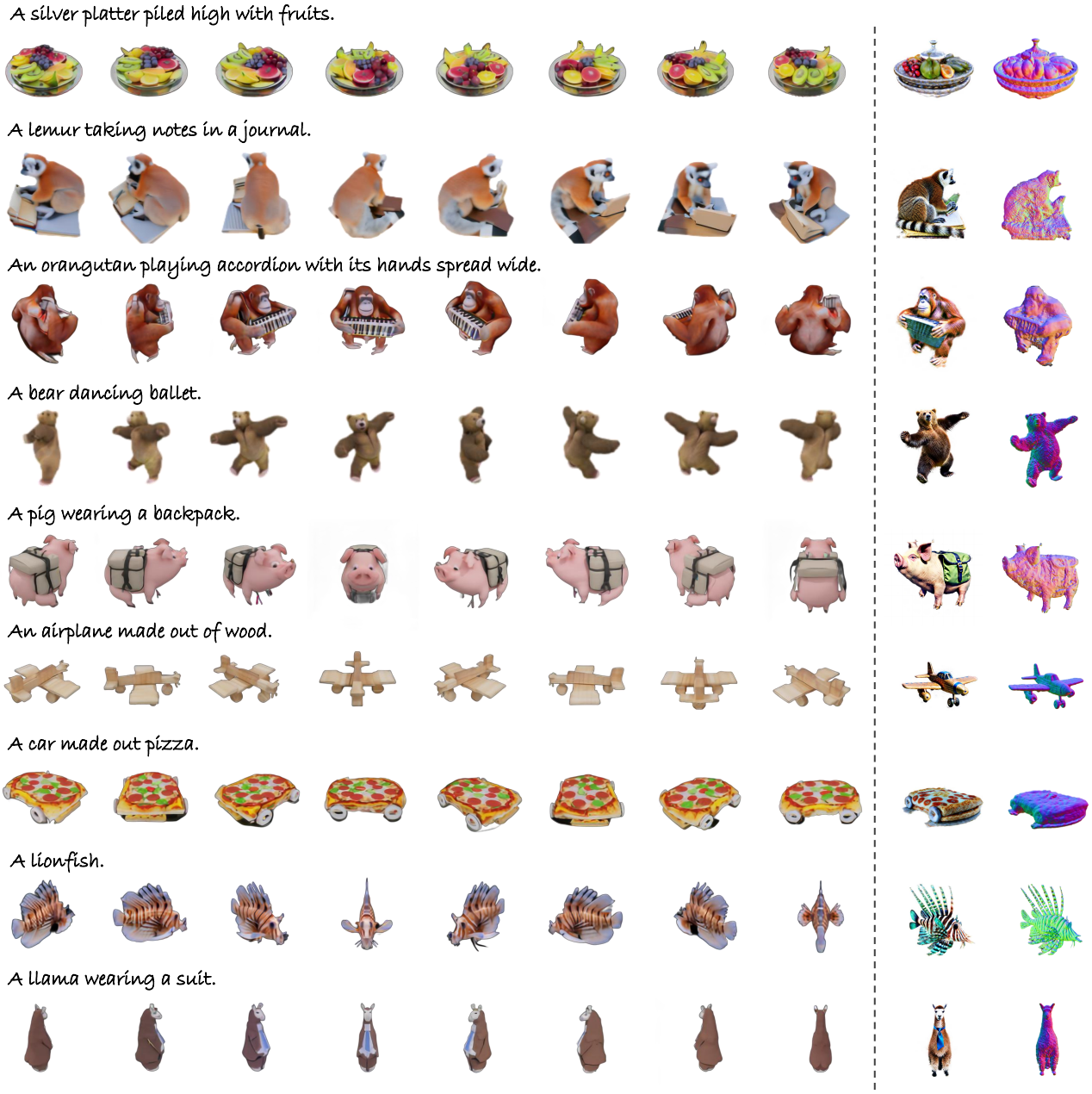

Figure 8. More generated 3D objects by our model. Left side shows the diffusion output and right side shows the 3D object after optimization.

5.2.2 2D Denoising Network

Our 2D denoising network contains a U-Net of the 2D foundation model (DeepFloyd) and a ControlNet [46] modulation module to jointly denoise the multi-view image set. In practice, given the M noisy images $\mathcal{V}_t = \{\mathcal{I}_t^i\}_{i=1}^M$ from the 2D diffusion process and M rendered images $\{\mathcal{H}^i\}_{i=1}^M$ from the 3D diffusion process as mentioned in Section 3.3 of the main paper, we first reshape both of them from $[B, M, C, H, W]$ to $[B \times M, C, H, W]$, where B, C, H, W denote batch size, channel, height, width, respectively. Then we feed the noisy images to the frozen encoder $\mathcal{E}^*$ of DeepFloyd to get encoded features:

$$P = \mathcal{E}^*(\text{Reshape}(\{\mathcal{I}_t^i\}_{i=1}^M), t, \text{emb}_{\text{text}}). \quad (15)$$

$P = \{p^k\}_{k=1}^K$ where p^k denotes the k-th features of the total K feature levels. Simultaneously, we feed the rendered images to the trainable copy encoder $\mathcal{E}$ of ControlNet to obtain the hierarchical 3D consistent condition features:

$$Q = \mathcal{E}(\text{Reshape}(\{\mathcal{H}^i\}_{i=1}^M), t, \text{emb}_{\text{text}}), \quad (16)$$

where $Q = \{q^k\}_{k=1}^{K}$. Subsequently, we decode P with the frozen decoder $\mathcal{D}^*$ of DeepFloyd and the condition residual features Q. Specifically, in the k-th decoding stage, we first apply zero-convolutions to the condition feature q^k and then add it to the original decoded features as residuals:

$$\hat{f}^k = p^k + \mathcal{D}^*_{k-1}(p^{k-1}) + \text{ZeroConv}(q^k), \qquad (17)$$

where $\mathcal{D}^*_{k-1}$ denotes the $k-1$-th frozen decoding layer of DeepFloyd. In this way, we can denoise the multi-view noisy images in a unified manner by introducing the 3D consistent condition signal as guidance. In practice, we set $M = 8$ in our experiments.

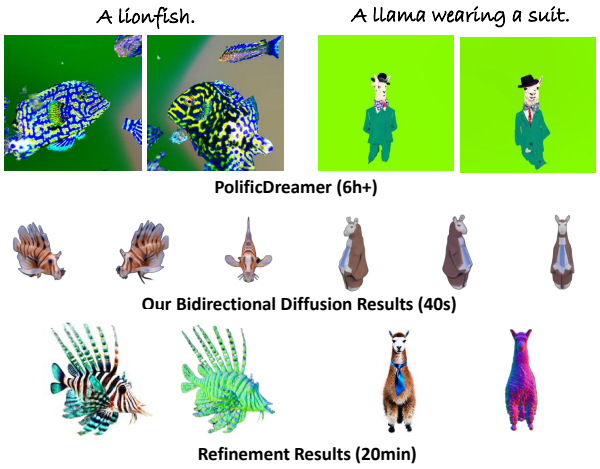

Figure 9. Comparison between our results with the object directly generated by the optimization method (ProlificDreamer).

5.2.3 Discussion of Post-optimization

A key challenge in post-optimization lies in preserving the consistency of the optimization outcomes with the inference results produced by the feedforward network. To address this challenge, our approach extends beyond the mere adjustment of hyperparameters. We have implemented a novel optimization strategy, employing score distillation to refine a residual radiance field. This refined field is subsequently superimposed onto the initialized field, rather than directly optimizing the initial radiance field itself. This method ensures that the optimized result remains closely aligned with the original input, mitigating significant deviations.

5.3. More Training Details

We train our framework on 4 NVIDIA A100 GPUs with a batch size of 4. For ShapeNet-Chair, the training takes about 8 hours to converge. For Objaverse 40k, the training takes 5 days. We use the AdamW optimizer with $\beta = (0.9, 0.999)$ and weight decay $= 0.01$. Notably, we set the learning rate of the 2D diffusion model to 2×10^{-6} while using a much larger learning rate of 5×10^{-5} for the 3D diffusion model.

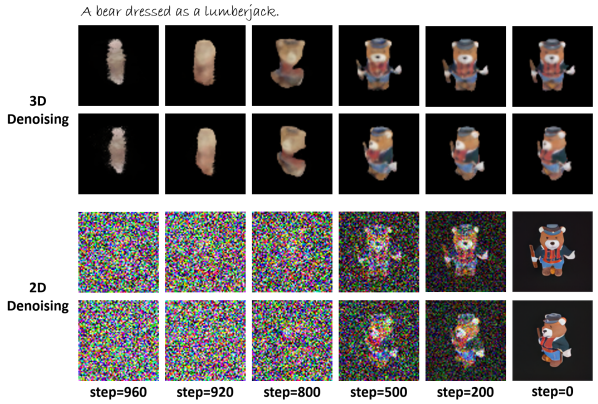

Figure 10. Visualization of our 2D and 3D denoising processes (the maximum diffusion step is 1,000). The top two rows show the rendering views of the implicit field during the 3D denoising process, and the bottom two rows show the 2D sample results during the 2D denoising process.

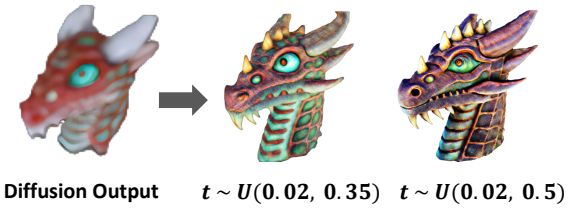

Figure 11. Ablation of range of noise level t for SDS.

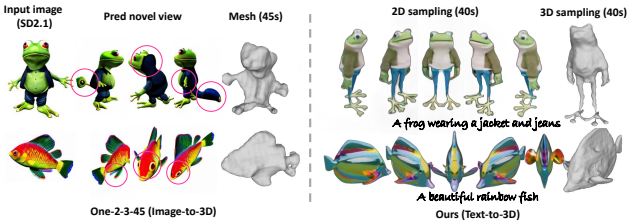

Figure 12. Comparison with One-2-3-45 without post-optimization.

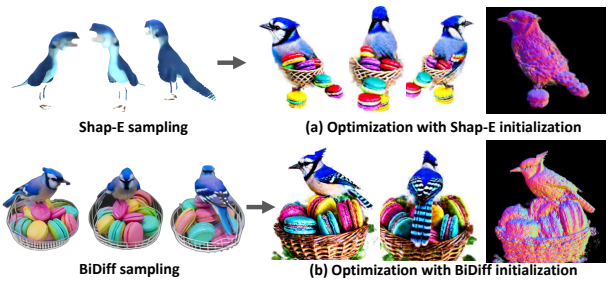

Figure 13. Comparisons with Shap-E initialization.

5.4. More Experiments

5.4.1 Comparison with One-2-3-45

We provide the comparison with One-2-3-45 [17]. We use text-to-image diffusion model to generate the reference images for One-2-3-45 due to it is an image-to-3d model. As shown in Fig. 12, BiDiff performs better in both consistency

and geometry.

5.4.2 Comparison with Shap-E initialization

We demonstrated the results of using different 3D generated feedforward models as post-optimization initialization. As shown in Fig. 13, the Shap-E initialization may fail because the initial shapes from Shap-E can not match the text and the textures lack details. While BiDiff initialization performs well even in complex situations.

5.4.3 Ablation for Priors

In Fig. 7, we provide additional results for the ablation of 3D and 2D priors mentioned in Sec. 4.3. Our method can produce more realistic textures with 2D priors and more robust geometry with 3D priors.

Range of noise level for SDS. The results in Fig. 11 illustrate the impact of the noise level during the entire optimization process, as discussed in Sec. 3.5. The 3D object generated with a smaller noise range is closer to the diffusion output. By adjusting the range of the noise level t_{opt}, we can effectively control the texture similarity between geometries before and after the optimization.

5.4.4 Visualization of 2D-3D Denoising

We also demonstrated the visualization of 2D and 3D denoising processes during bidirectional diffusion sampling as shown in Fig. 10. The top two lines show the rendering views of the implicit field during the 3D denoising process, and the bottom two lines show the 2D sample results during the 2D denoising process. 3D and 2D representations are jointly denoised, and in the early step of diffusion sampling, 3D representations can provide basic geometric shapes, which guides 2D diffusion to generate geometrically reasonable images. In the later step of sampling, texture generation is dominated by 2D diffusion.

5.4.5 More Results

In Fig. 8, we provide more high-quality results generated by our entire framework. And in Fig. 9, we demonstrated a comparison with the previous state-of-the-art optimization method [41]]. Our approach not only significantly reduces time costs but is also more robust in understanding geometry.

5.5. Limitations

A primary limitation of our Bidiff method stems from its dependency on the capabilities of foundational 2D and 3D models. Specifically, its creative potential is constrained by the upper limits of 2D models, especially when interpreting highly complex textual descriptions. Moreover, due to the weaker existing 3D foundational models (e.g., Shap-E), the 3D performance of feedforward inference results is more affected by the quality of training data. Additionally, the post-processing stage, which is based on the score distillation techniques, is subject to their prevalent issues, such as color saturation. Consequently, Bidiff is not exempt from the inherent challenges faced by the technologies it incorporates.