BoostDream: Efficient Refining for High-Quality Text-to-3D Generation from Multi-View Diffusion

Yonghao Yu*, Shunan Zhu¹, Huai Qin¹ and Haorui Li²

¹Waseda University
²Southeast University

yuyonghao@suou.waseda.jp, {shunan-zhu, mizuki_qin}@ruri.waseda.jp, lihaorui.lhr@alibaba-inc.com

Abstract

Witnessing the evolution of text-to-image diffusion models, significant strides have been made in text-to-3D generation. Currently, two primary paradigms dominate the field of text-to-3D: the feed-forward generation solutions, capable of swiftly producing 3D assets but often yielding coarse results, and the Score Distillation Sampling (SDS) based solutions, known for generating high-fidelity 3D assets albeit at a slower pace. The synergistic integration of these methods holds substantial promise for advancing 3D generation techniques. In this paper, we present BoostDream, a highly efficient plug-and-play 3D refining method designed to transform coarse 3D assets into high-quality. The BoostDream framework comprises three distinct processes: (1) We introduce 3D model distillation that fits differentiable representations from the 3D assets obtained through feed-forward generation. (2) A novel multi-view SDS loss is designed, which utilizes a multi-view aware 2D diffusion model to refine the 3D assets. (3) We propose to use prompt and multi-view consistent normal maps as guidance in refinement. Our extensive experiment is conducted on different differentiable 3D representations, revealing that BoostDream excels in generating high-quality 3D assets rapidly, overcoming the Janus problem compared to conventional SDS-based methods. This breakthrough signifies a substantial advancement in both the efficiency and quality of 3D generation processes.

1 Introduction

The significance of 3D assets has grown remarkably due to the expansion of virtual reality and the gaming industry. Creating these assets, however, involves considerable labor costs. Recently, the popularity of methods based on differentiable rendering [Mildenhall et al., 2021; Shen et al., 2021; Wang et al., 2021; Kerbl et al., 2023] and the explosion of text-to-image models [Rombach et al., 2022; Zhang et al., 2023a] provides a new line of methods for generating 3D assets.

Plenty of research [Nichol et al., 2022; Jun and Nichol, 2023a; Poole et al., 2022; Wang et al., 2023b; Lin et al., 2023; Chen et al., 2023b] in the text-to-3D domain aim to lower 3D asset creation costs, demonstrating the efficacy of leveraging 2D generation models for 3D tasks. Currently, there are two primary strategies in the field of text-to-3D generation. The first, exemplified by Point-E and Shap-E [Nichol et al., 2022; Jun and Nichol, 2023a], employs a feed-forward approach, using 3D datasets or latent representation mapped by the 3D datasets to train diffusion models. This enables quick, text-prompt-based 3D asset generation. However, these methods grapple with the limited variety and scope of 3D datasets.

In response to the constraints posed by limited 3D datasets, the SDS-based optimization method was developed. Models like DreamFusion and Magic3D [Poole et al., 2022; Lin et al., 2023] exemplify this approach. They employ an SDS loss function [Poole et al., 2022], enabling text-to-image diffusion models to train 3D assets predominantly represented by methods based on differentiable rendering, such as NeRF [Mildenhall et al., 2021]. This approach allows for the generation of high-quality 3D assets, leveraging the strong text comprehension and 2D generation abilities of diffusion models. However, this method is hindered by slow generation speeds, as each text prompt requires training a 3D representation from scratch and suffering from the Janus (multi-head) problem. How to efficiently generate high-quality 3D assets remains an urgent problem to be solved.

With this in mind, we propose a generation method suitable for various 3D representations, utilizing the rapid generation of the feed-forward method and the high-quality generation characteristics of the SDS-based method. Our method does not just simply combine the feed-forward approach with the SDS-based method. Specifically, we have developed an innovative three-stage method named BoostDream. In the first stage, we designed a rapid initialization technique that transforms explicitly represented 3D assets generated by the feed-forward method into 3D representations based on differentiable rendering. In the second stage, we introduced a multi-view rendering system and a multi-view SDS under the control of the original 3D input condition. In the third stage, we solely rely on self-supervision to achieve refined generation results. Extensive experiments are conducted, including...
Figure 1: Comparison of 3D Generation Results of baseline and BoostDream. Provided with a coarse 3D asset and text prompt pair, BoostDream can refine it into a high-quality 3D asset efficiently. In each set of images, the image on the left is the coarse 3D asset generated by Shap-E [Jun and Nichol, 2023a], and the three images on the right are our refined 3D asset.

refinement and comparison experiments. As shown in Figure 1, our method can efficiently refine the coarse 3D for high-quality results. In summary, the contributions of this paper include the following:

- We propose a novel method that integrates the advances in feed-forward and SDS-based methods, enabling efficient and high-quality refinement of 3D assets.
- We innovatively propose the multi-view SDS with the multi-view render system to refine 3D assets under multi-view conditions and address the Janus problem.
- Our BoostDream can generate high-quality 3D assets based on a variety of 3D differentiable representations, showcasing strong generalizability.

2 Related Work

In this section, we discuss the related work on text-to-3D generation using diffusion models. One mainstream approach is the feed-forward method using 3D datasets for training, and another is SDS-based optimization. In addition, the multi-view perceptual diffusion method provides ideas for solving the Janus problem and has proved effective.

2.1 Feed-Forward Generation Method

Feed-forward generation method tries to directly utilize the diffusion models in text-to-3D tasks, which trained diffusion models with 3D assets as datasets [Liu et al., 2023c; Liu et al., 2023a; Sanghi et al., 2023; Nichol et al., 2022; Zhou et al., 2021] or perform diffusion with a latent representation of 3D assets [Gupta et al., 2023; Ntavelis et al., 2023; Chen et al., 2023a; Zeng et al., 2022]. Shap-E [Jun and Nichol, 2023a] is a typical example of this type of approach. It trains a 3D encoder to transform 3D assets to a set of parameters of an implicit function and then trains a diffusion model on these parameters. Compared to the methods using frozen 2D diffusion models, these approaches can generate 3D assets at a very fast speed due to their feed-forward nature. However, these methods were trained with the 3D datasets, while the 3D dataset available today is relatively small and low-quality. Consider the dataset used in 2D image generation tasks, Laion5B [Schuhmann et al., 2022] contains more than 5 billion image-text pairs while the largest 3D dataset available, Objaverse-XL [Deitke et al., 2023] can only carry 10 million 3D assets with worse quality captions. This causes the 3D generation results to be less sharp, have lower fidelity, and be unable to produce assets with complex semantics.

2.2 SDS-Based Optimization Generation Method

Inspired by the great success of using diffusion models in text-to-image tasks, one mainstream of works focuses on using 2D diffusion models along with differentiable 3D representing techniques like NeRF [Mildenhall et al., 2021] to facilitate 3D generation. Dreamfields [Jain et al., 2022] used pre-trained CLIP models to optimize NeRF, hoping to get a visually real result. But it takes much time and computing resources to generate. DreamFusion [Poole et al., 2022] proposes SDS loss to optimize NeRF representation. SJC [Wang et al., 2023a] is a concurrent effort of Dreamfusion, with the difference that SJC applies the chain rule to the 2D score. Research on providing supervision of novel views via SDS loss is becoming a trend [Lin et al., 2023; Chen et al., 2023b; Metzer et al., 2023; Liu et al., 2023b; Qian et al., 2023; Melas-Kyriazi et al., 2023; Lorraine et al., 2023]. Among them, ProlificDreamer [Wang et al., 2023b] models the 3D parameters as variable, proposes variable fraction distillation
(VSD) to address the shortcomings in SDS. These works indicate the capability to produce high-quality outputs enriched with detailed semantic information derived from 2D diffusion models. However, these SDS-based methods are impeded by prolonged optimization periods and the occurrence of Janus problems.

2.3 Multi-View in 3D Generation

The multi-view aware diffusion model in image generation has made great progress recently. MVDiffusion [Tang et al., 2023] is capable of generating consistent multi-view images from text prompts given pixel-to-pixel correspondences with global awareness. The success of such a method also draws attention to 3D generation tasks. EfficientDreamer [Zhao et al., 2023] used a 2D diffusion model that can generate an image consisting of four orthogonal-view sub-images and use it in the 3D generation. It has been proven to be efficient in improving the quality of 3D content and alleviating the Janus problem. MVDream [Shi et al., 2023] further discusses the situation of building a multi-view diffusion model from both 2D and 3D data. It can benefit from the generalizability of the 2D diffusion model while keeping the consistency of 3D rendering, solving the Janus problem and content drift problem across different views. However, after training by the 3D datasets, the realistic texture information that the 2D diffusion model contained can be degraded. In our work, a multi-view rendering method is designed and a multi-view SDS loss function is formulated to guide the differentiable 3D representation, mitigating the Janus problem while obtaining high-quality generated results.

3 BoostDream

In this section, we propose our BoostDream model. We initially present a brief overview of the background of text-to-3D generation. Subsequently, we introduce our multi-view refining approach, detailing its design and implementation within the context of enhanced 3D asset generation by quick initialization based on differentiable rendering, multi-view render system, and multi-view SDS. Figure 2 illustrates the overview of our method.

3.1 Background

For text-to-image tasks, [Ho et al., 2020] has proposed a simplified training goal of Denoising Diffusion Probabilistic Models (DDPM). It is expressed as:

\[
L_{DDPM} = E_{t,x_0,\epsilon} \left[ \left\| \epsilon - \epsilon_\theta (\sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} \epsilon, t) \right\|^2 \right]
\]

Here the \( x_t \) is latent at time step \( t \), \( \epsilon \) is the actual noise, \( \epsilon_\theta \) represents the predicted noise, \( \alpha_t = 1 - \beta_t \) where \( \beta_t \) is the noise added at time step \( t \) in a diffusion model.

In text-to-3D tasks using text-prompt as the generation condition, DreamFusion [Poole et al., 2022; Wang et al., 2023b] has proposed the SDS loss. For a differentiable 3D rendering represented as \( g(\theta, c) \), where \( \theta \) are its parameters and \( c \) are the camera parameters. Let \( q^\theta_t (x_t | c) \) be the distribution from diffusing \( x_0 = g(\theta, c) \) to moment \( t \). The optimization goal of SDS is by optimizing \( \theta \), make the marginal distribution of the diffusion process of \( q^\theta_t (x_t | c) \), approach the marginal distribution of a pre-trained text-to-image model \( p(x_t | y) \) where \( y \) is text-prompt. The training goal of the SDS loss method can be represented as:

\[
\min_{\theta} L_{SDS} = E_{t,c} \left[ (\sigma_t/\alpha_t) w(t) D_{KL} (q^\theta_t (x_t | c) \| p(x_t | y)) \right]
\]

The SDS method estimates the gradient of the optimization object \( L_{SDS} \) as follows, allowing the text-to-3D task to be
carried out quickly and without losing too much precision:

\[ \nabla_{\theta} L_{SDS} = \mathbb{E}_{t, \epsilon, c} \left[ w(t) (\epsilon_{\theta}(x_t, t, y) - \epsilon) \frac{\partial g(\theta, c)}{\partial \theta} \right] \quad (3) \]

The idea of adding other control conditions in BoostDream is inspired by ControlNet \cite{zhang2023boostdream}, which is a text-to-image task solution allowing more input as task-specific conditions like images. Its training goal can be written as:

\[ L_{Control} = \mathbb{E}_{x_0, t, c, f, \epsilon} \left[ \| \epsilon - \epsilon_{\theta}(x_t, t, c, f) \|^2 \right] \quad (4) \]

In this formulation, the terms \( c_i \) and \( c_f \) are task-specific conditions applied in ControlNet, like text-prompt and images.

### 3.2 3D Representation Initialization

Our method is applicable to a variety of 3D asset generation approaches based on differentiable rendering. In order to achieve this, we propose a generalized model distillation approach to fit a coarse 3D asset into a randomly initialized differential representation.

We define the coarse explicit 3D assets as \( a \) and the parameters of differentiable 3D representation as \( \theta \). In every iteration, the randomly sampled camera parameters \( c \) are passed into the renderer \( g \) to get the rendered images for these two 3D assets separately at the same view. With these pairs of images, we use \( L_1 \) loss function to make the randomly initialized 3D representation look as much like the coarse explicit 3D asset as possible in a very short time. The loss function is defined as:

\[ L_1 = |g(a, c) - g(\theta, c)| \quad (5) \]

### 3.3 Multi-View Render System

To alleviate the Janus problem, we designed a new multi-view rendering method. As shown in the two boost stages of Figure 2, we initialize the default viewing direction in accordance with the input 3D model whose origin coincides with the origin of the coordinate system. At each iteration, we randomly sampled a camera position \( p_0 \) in spherical coordinates, consisting of elevation angle \( \phi_{\text{cam}} \in [-10^\circ, 70^\circ] \), azimuth angle \( \theta_{\text{cam}} \in [0^\circ, 360^\circ] \), field of view \( f \in [0^\circ, 180^\circ] \) and camera distance \( d_{\text{cam}} \). We further create a rotation axis, a random vector passing through the origin denoted as \( a = (a_x, a_y, a_z) \), a rotation angle denoted as \( \alpha \). To get multi-view camera position, a rotation matrix \( R \) is defined as:

\[ R(a, \alpha) = I + (\sin \alpha) K + (1 - \cos \alpha) K^2 \quad (6) \]

Where \( I \) is the identity matrix, and \( K \) is the skew-symmetric matrix generated from the rotation matrix \( a = (a_x, a_y, a_z) \), defined as:

\[ K = \begin{pmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{pmatrix} \quad (7) \]

In our experimental settings, we define rotation angle \( \alpha = 90^\circ \), so the camera at position \( p_0 \) will rotate around axis \( a \) and get four camera positions. In each rotation iteration \( i \), the camera position \( p_i \) after rotation follows:

\[ p_i = R(a, i \times \alpha)p_0 \quad (8) \]

We use separate renderers for each task in different 3D representations. For a renderer \( g \), we use the above four camera positions \( p_i \) along with other parameters to get camera parameters \( c_i \). With a total of four sub-images generated from different camera parameters \( c_i \) and parameters of a 3D representation \( \theta \), stitch these four images together to create one large \( 2 \times 2 \) composite image \( G \). We also capture the normal maps rendered from 3D representation and follow a similar routine to get a large \( 2 \times 2 \) normal map \( N \).

\[ G = \begin{pmatrix} g(\theta, c_0) \\ g(\theta, c_1) \\ g(\theta, c_2) \\ g(\theta, c_3) \end{pmatrix} \quad (9) \]

### 3.4 Multi-View SDS

The multi-view SDS loss function is designed to direct differentiable rendering models, utilizing both text prompt and multi-view conditions generated by the camera parameters we proposed in section 3.3 as the control guidance. We use multi-view normal maps here for multi-view conditions, as they provide a detailed depiction of surface normals, enhancing the capture of finer geometric and textural details. This enhancement is particularly beneficial for accurately preserving complex features like hair or surface irregularities, which are essential for lifelike 3D renderings.

The heart of this loss function is a new method of estimating noise guided by dual task-specific conditions, like normal maps and the text prompt. This dual-conditioned approach ensures that the generated content remains faithful to the surface details outlined by the normal maps while also aligning with the context provided by the text prompt. The noise estimation is formulated as:

\[ \hat{\epsilon}_\phi(x_t; t, y, N) = \epsilon_\phi(x_t; t, y, \lambda N) + s (\epsilon_\phi(x_t; t, y, \lambda N) - \epsilon_\phi(x_t; t)) \quad (10) \]

Here, \( \epsilon_\phi \) denotes the noise predicted by the diffusion model, \( \lambda \in [0, 1] \) balances the normal map conditioned and unconditional noise predictions \cite{chen2023diffuser}, and \( s \) is the scale of classifier-free guidance \cite{ho2022classifier}. \( N \) represents the multi-view normal maps captured following the rule we proposed in section 3.3, for normal map renderer \( g_N \), we use the same camera parameters used by composite image \( G \) in Eq. 9:

\[ N = \begin{pmatrix} g_N(\theta, c_0) \\ g_N(\theta, c_1) \\ g_N(\theta, c_2) \\ g_N(\theta, c_3) \end{pmatrix} \quad (11) \]

Like Eq. 3, the gradient of the multi-view SDS loss concerning the differentiable representations’ parameters is defined as:

\[ \nabla_{\theta} L_{\text{MV-SDS}} = \mathbb{E}_{t, \epsilon} \left[ w(t)(\hat{\epsilon}_\phi(x_t; t, y, N) - \epsilon) \frac{\partial G}{\partial \theta} \right] \quad (12) \]

Furthermore, We enhance the loss with orientation loss \( \mathcal{L}_{\text{orient}} \) and opacity loss \( \mathcal{L}_{\text{opacity}} \) proposed at \cite{poole2022image}, when implement with NeRF.

The final loss function combines these components to ensure accurate surface detail representation, correct surface orientations, and optimal opacity:

\[ \mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MV-SDS}}(\phi, x) + \alpha \mathcal{L}_{\text{orient}} + \beta \mathcal{L}_{\text{opacity}} \quad (13) \]
where $\alpha$ and $\beta$ are the weights for the orientation and opacity losses, respectively. This comprehensive loss function fosters the generation of 3D content that is both semantically and geometrically accurate, exhibiting realistic physical properties.

4 Experiments

In this section, we present a series of experiments designed to evaluate the capabilities of our BoostDream method. In Section 4.1, we detail the implementation specifics. Section 4.2 discusses the results of refinement experiments and explores its performance under various text prompts. In Section 4.3, we compare our method against existing techniques in terms of performance and speed. We conduct some ablation studies to illustrate the effect of each stage and condition setting of our method in Section 4.4. We also defined a user study to quantify the model performance which is shown in Section 4.5. These experiments are crucial in demonstrating the effectiveness of BoostDream in refining and editing 3D assets and the generalisability when applied to different 3D representations.

4.1 Implementation Details

All the experiments in this paper are conducted on a single NVIDIA V100 GPU with 32GB VRAM. We use Shap-E [Jun and Nichol, 2023a] from official implementation [Jun and Nichol, 2023b] to generate coarse 3D assets and fit them into differentiable 3D representations in the first 200 iterations. After initialization, we set 4800 iterations for the refining process, of which the first 1800 iterations are under the guidance of the coarse 3D assets, and the subsequent 3000 iterations are guided by the differentiable rendering result itself. We use ControlNet 1.1 with Stable Diffusion 1.5 [Zhang et al., 2023b] as the diffusion model. In the comparison experiment, we carried out experiments on different differentiable 3D representations, including NeRF [Mildenhall et al., 2021], DMTet [Shen et al., 2021], and 3D Gaussian Splatting [Kerbl et al., 2023]. Please refer to the Appendix [Yonghao et al., 2024] for more details and results.

4.2 Refinement Experiment

We designed a refinement experiment that shows the powerful refining capabilities of our BoostDream. We choose Shap-E as the feed-forward generation method to generate coarse 3D assets. The coarse 3D assets will be sent to our refining method with the same text prompt. We use NeRF implementation here as an example. The result is shown in Figure 1. Our BoostDream can also refine coarse 3D assets with effective prompt-based editing. Experiments demonstrate its ability to refine assets using varied text prompts while retaining original features, as illustrated in Figure 3. Notably, examples like “Hulk” and “Batman” preserve the original posture and red elements from the “Iron Man” asset, showcasing a delicate balance between adaptation and preservation.

4.3 Comparison Experiment

Our BoostDream is a refining method that combines the advantages of the feed-forward generation method and the SDS-based optimization generation method. To demonstrate the effectiveness of our methods, we selected methods from each of the above two categories for the comparison experiment. For the feed-forward generation method, we choose Shap-E, which has an official implementation [Jun and Nichol, 2023b]. The 3D assets generated by Shap-E are also used
<table>
<thead>
<tr>
<th>BoostDream-NeRF (Ours)</th>
<th>Shap-E</th>
<th>DreamFusion</th>
<th>Magic3D</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Image" /></td>
<td><img src="image2" alt="Image" /></td>
<td><img src="image3" alt="Image" /></td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td><img src="image5" alt="Image" /></td>
<td><img src="image6" alt="Image" /></td>
<td><img src="image7" alt="Image" /></td>
<td><img src="image8" alt="Image" /></td>
</tr>
</tbody>
</table>

- a Gundam with a weapon
- a pagoda in Japanese style on the cloud
- a Chinese garden tree
- a brown and white horse is wearing a blue saddle
- a corgi with a red collar
- A 3D model of an adorable cottage with a thatched roof
- an elephant with jewellery
- a fire breathing dragon

Figure 4: Comparison with Shap-E [Jun and Nichol, 2023a], DreamFusion [Poole et al., 2022] and Magic3D [Lin et al., 2023] for the same text-to-3D generation task. Our model has significantly stronger prompt relevancy and much better quality (best viewed by zooming in). See the results of our method on DMTet [Shen et al., 2021] and 3D Gaussian Splatting [Kerbl et al., 2023] in the Appendix [Yonghao et al., 2024].
to initialize our BoostDream method. For the SDS-based optimization generation method, we choose DreamFusion and Magic3D, which are implemented by threestudio [Guo et al., 2023] because no official code is available now. It is also worth noticing there are some differences between the original papers and the threestudio implementation. In this implementation, DreamFusion and Magic3D use DeepFloyd [StabilityAI, 2023] as the diffusion model, while the originals useImagen [Saharia et al., 2022] and eDiff-I [Balaji et al., 2022]. We use the default parameter of these methods. The experiment result is shown in Figure 4. Our method can generate higher-quality 3D assets compared to the above methods. In rows 1, 4, and 7, DreamFusion and Magic3D have arisen with the Janus problem, which means the Gundam has multiple arms, the horse has wrong legs, and the elephant has numerous heads, while our method can alleviate this problem due to the three-stage refine.

Also, compared to the two SDS-based optimization generation methods, the total generation time we need is reduced. Our method only requires 5000 iterations in total. As a comparison, the DreamFusion needs 15000 iterations to get a 3D asset, while the Magic3D needs 5000 iterations to train a coarse 3D asset and has a 3000 iteration refinement process. The speed comparison results are shown in Table 1. From the result, we can observe that apart from our high-quality generation result, BoostDream can rapidly finish optimization.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (V100-sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoostDream-NeRF</td>
<td>2038</td>
</tr>
<tr>
<td>BoostDream-DMTet</td>
<td>2128</td>
</tr>
<tr>
<td>BoostDream-3D Gaussian Splatting</td>
<td>1977</td>
</tr>
<tr>
<td>DreamFusion</td>
<td>3519</td>
</tr>
<tr>
<td>Magic3D</td>
<td>2355</td>
</tr>
</tbody>
</table>

Table 1: Speed comparison. Each time is the average training time of ten different text prompts. Note that DreamFusion and Magic3D are tested on threestudio implementation.

4.4 Ablation Study

In the proposed pipeline, we delineate three distinct stages: the initialization stage, the boost stage under the guidance of a coarse 3D asset, and the self-boost stage under the guidance of its intrinsic multiview. To elucidate the impact of each stage, we undertake an ablation study where we systematically eliminate each stage in isolation to assess their contributions. The results are shown in Figure 5, which illustrates that instead of using the initialization stage, the model may not converge quickly and have wrong guidance from the 3D assets we want when randomly initialized. Without coarse 3D asset guidance, the model demonstrates a propensity to evolve towards unforeseen outcomes. This phenomenon can be attributed to the premise that post-initialization errors perpetuate a cycle of learning based on these inaccuracies. Finally, without its self-guidance, the model is constrained by the original coarse 3D asset, thus failing to achieve a level of detail and quality that is otherwise attainable.

We also conducted two other ablation studies. One uses the coarse 3D asset generated by the feed-forward method to initialize the traditional SDS-based method. Another is using different control conditions to replace normal maps, such as canny edge and depth maps. See more ablation studies in the Appendix [Yonghao et al., 2024].

4.5 User Study

Due to the lack of existing evaluation metrics for 3D generation, we conduct a user study to assess model performance. We create an evaluation set generated through 30 prompts using four methods. For a specific 3D asset, a rendered video with an input text prompt was shown to participants. 20 participants are invited to individually score each item in the set, focusing on prompt relevance and the quality of generated details, with scores ranging from 1 to 5. As shown in Figure 6, the results demonstrate that our method is markedly superior.

5 Conclusion

In summary, our research introduces BoostDream, a novel method that seamlessly combines differentiable rendering and text-to-image advancements to generate high-quality 3D assets efficiently. Central to our approach is the multi-view SDS, which effectively overcomes the Janus problem in generation processes. BoostDream can be applied to various differentiable 3D representations, generally improving the quality and reducing the time consumption of existing 3D generation methods. This work not only advances the quality and diversity of 3D assets but also sets a precedent for future innovations in 3D modeling, with wide-reaching implications for virtual reality and gaming industries.
Ethical Statement

The BoostDream proposed in this paper uses the 2D diffusion model as the prior to generating high-quality 3D assets with a multi-view strategy. We aim to reduce the cost of creating highly detailed 3D assets. However, we do note that the method could be applied to generate disinformation or violent and sexual content. With 2D diffusion as prior, it also inherits similar ethical and legal considerations to problematic biases and limitations that 2D diffusion models may have.

References


