I-SKED: Improved Sketch-guided Text-based 3D Modeling

Maggie Wu
Department of Computer Science
Stanford University
mwu9@stanford.edu

Zhongchun Yu
Department of Mechanical Engineering
Stanford University
zcyu@stanford.edu

Haoming Zou
Department of Electrical Engineering
Stanford University
zhm2023@stanford.edu

Github repo: https://github.com/Haoming2000/I-SKED2

Abstract

In this work, we introduce I-SKED, an improved version of the Sketch-guided Text-based 3D Editing (SKED) framework for locally modifying existing 3D models using sketches and text prompts. Our pipeline enables users to do intuitive, CAD-like modeling without requiring extensive technical expertise. The improvements introduced in I-SKED address key limitations of the original framework such as the inability to perform deletions and handle non-overlapping edits. By integrating a debiasing strategy with Neural Radiance Fields (NeRF), our method supports detailed edits and resolves multi-face issues inherent in imperfect sketches, thereby enhancing the consistency and realism of the edited 3D objects. This work enhances the functionality and usability of the SKED pipeline, enabling more intuitive and effective user interactions.

1. Introduction

Editing and modifying 3D computer-aided design (CAD) models can be time-consuming and requires technical skills and domain knowledge that may not be accessible to all people who want to do so. Additionally, being able to quickly iterate through edits to a 3D model is often desirable for visualizing changes. In order to provide a usable pipeline for people who want to modify or edit their existing 3D models, we propose I-SKED, a pipeline where users are able to perform a variety of operations on 3D models through intuitive sketch and text prompt inputs.

This project aims to create a pipeline that transforms initial concepts conveyed through sketches and text into a modified 3D model. A user will be able to refine intermediate 3D models using sketch modifications with text instructions. Common and important aspects of modeling include the ability to add, delete, and replace parts of 3D models in order to iterate on creating a final model. Current methods for using Stable Diffusion [10] to edit models such as SKED [5] are fairly successful at producing novel 3D models using additive sketches, where various features are added on top of an existing model. However, we have found that the model fails when asked to delete or shrink existing parts of the model, and is incredibly limited in even its additive edits.

In order to create a pipeline that users can use in real-life scenarios for prototyping models, we aim to first replicate the results of the SKED paper, then to improve upon them to allow deletions and replacements as modifications to models in the prototyping process, in addition to improving the quality of their existing results. We create our own data to use for various use cases of the SKED pipeline, and add a new loss function to allow deletions of models. We also add score debiasing [2] to our score distillation sampling loss, which resolves various quality issues with the original SKED model.

To evaluate the results of our model, we generate results using the original baseline SKED model, our model, and an alternative Gaussian-splatting model [11]. We survey over 30 people on which model they think produced the best results corresponding to the text prompts used to generate each result. Through our improvements to the model, we show that we enable deletions, replacements, overlapping edits, and higher-quality results for additive edits to 3D models, and we also greatly alleviate the Janus (multi-face) problem as described in [2]. Our model enables iterative edits that produce high-quality results at each step, resulting in a more complete pipeline that makes 3D modeling more accessible and helps creators of 3D models iterate through ideas quickly but with enough user guidance to produce useful models.

2. Related Work

In recent years, the success of image diffusion models such as Stable Diffusion [10] has enabled text-to-3D methods to generate 3D models from scratch. DreamFusion [8] uses a 2D text-to-image diffusion model as a prior for optimization in order to refine a 3D NeRF [6] model. DreamFusion introduces the score distillation sampling loss, which uses score functions from the trained diffusion model to generate 3D models that look good when rendered from random angles. This allows a NeRF to be generated from just a short text prompt.

DreamGaussian [11], similarly to DreamFusion, uses an image diffusion prior to generate a 3D model, using Gaussian splatting [3] rather than training a NeRF. DreamGaussian allows for any combination of text and image input to create and refine a 3D model, and additionally designs a mesh-extraction algorithm for texture refinement. While both DreamFusion and DreamGaussian are flexible methods for generating 3D models from simple user input, they are not methods designed for iterating upon existing 3D models.

Our main reference for this project is SKED: Sketch-guided Text-based 3D Editing [5]. SKED takes a pre-trained Neural Radiance Field (NeRF) [6] as input, lets users provide sketches on top of the NeRF from multiple views along with a text prompt, and then uses diffusion to fill in the added space to generate a modified NeRF. This pipeline thus allows for iterative edits to a 3D model. However, in our experiments with the SKED model, we find that many simple edits are not possible. Our project improves upon the SKED model by adding a loss function that enables deletion from 3D models, along with score debiasing, which greatly improves the quality of generated results.

Debiased score distillation sampling [2] is introduced as a solution to the Janus (multi-face) problem, where an object's most canonical view appears from multiple views due to the inherent bias of 2D diffusion models. Score debiasing uses a coarse-to-fine method where gradients of the score distillation sampling loss are clipped dynamically to prevent the score from being too biased towards specific directions or gradients from the user prompt. We incorporate the methods for score debiasing in our project in order to resolve various generation quality issues, including the Janus problem.

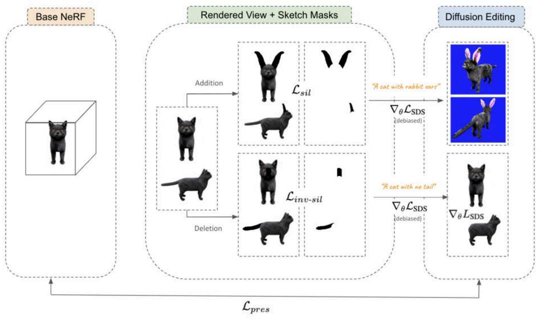
Base NeRF
Rendered View + Sketch Masks
Diffusion Editing
Addition
Deletion
$\mathcal{L}_{sil}$
$\mathcal{L}_{inv\text{-}sil}$
"A cat with rabbit ears"
$\nabla_\theta \mathcal{L}_{\text{SDS}}$
(debiased)
"A cat with no tail"
$\nabla_\theta \mathcal{L}_{\text{SDS}}$
(debiased)
$\nabla_\theta L_{\text{SDS}}$
$\mathcal{L}_{pres}$

Figure 1. Overview of I-SKED pipeline. We start with a NeRF model, and using 2 rendered views of the model, draw the corresponding masks for either addition or deletion. We then use diffusion editing to modify the NeRF model based on the provided sketch and text prompt to obtain a final model.

3. Technical Approach

3.1. Framework

Our method (see Figure 1) builds on the foundational principles of the Sketch-based Editing via Diffusion (SKED) framework [5], combining Neural Radiance Fields (NeRF) [6] and diffusion-based techniques to enable precise local editing while preserving the integrity of unedited scene parts. By utilizing NeRF for renderings and diffusion processes for guided edits, our approach is iteratively refined through user-driven sketches and textual prompts. This method enables rapid prototyping and continuous iterative modifications, providing an easy method for 3D modeling.

1. **Base NeRF Model Initialization:** We start with base NeRF model F_o, which can be prepared using techniques such as multiview reconstruction [7], text-to-3D pipelines [8], or direct modeling from a customized dataset. In our experiments, we use BlenderNeRF [9] to render views (along with their corresponding camera poses) of meshes in order to create training data for a base NeRF mdoel. The renderings of the base NeRF model from N different views, denoted as $\{C_i\}_{i=1}^N$, serve as canvases for users to draw sketches on to indicate desired modifications.

2. **Sketches for Local Edit Mapping:** Users sketch directly on the selected rendered views, indicating desired changes. These sketches are then preprocessed to generate binary masks, $\{M_i\}_{i=1}^N$, which isolate the edited regions on the renderings.

3. **Diffusion-Based Editing:** Employing a modified stable diffusion approach, we apply semantic and structural edits as guided by the sketches and a text prompt

y. This step refines F_e, an editable copy of F_o, adjusting the scene's 3D model to align with the sketches and text descriptions.

4. **Final Rendering:** The edited NeRF model F_e is used to render the final 3D object from random or specified views, verifying the edits conform to the intended design and text specifications.

3.2. Score Distillation Sampling

In diffusion-based editing, we utilize the Score Distillation Sampling (SDS) method, first introduced in DreamFusion [8], to ensure the outputs of a volumetric renderer align with high-fidelity distributions from advanced diffusion processes. This approach guarantees semantic coherence with text prompts. We employ Neural Radiance Fields (NeRF) as the renderer, where I_θ represents images rendered from complex 3D scenes, parameterized by θ, and adapted by learned parameters to varying conditions.

Optimization via Diffusion Loss Minimization

Within our framework, we incorporate a pre-trained denoising model, denoted as $\hat{\epsilon}_\phi$, from Stable-DreamFusion [4]. This integration aims to optimize the generated result by minimizing the diffusion loss $\mathcal{L}_{\text{diff}}$, thereby enhancing the editing capabilities of NeRF. The loss function is defined as:

$$\mathcal{L}_{\text{diff}}(\phi, I_\theta) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I),t}\left[\|\hat{\epsilon}_\phi(I_t, y, t) - \epsilon\|^2\right], \tag{1}$$

where y represents the text prompt guiding the generation, and $I_t = \sqrt{\alpha_t} I_\theta + \sqrt{1-\alpha_t}\epsilon$ is the t-step noised version of I_θ, rendered from a NeRF model parameterized by θ. This loss function quantifies the discrepancy between the predicted noise by the diffusion model and the actual noise introduced, which is critical for training a NeRF to generate images that closely match the distributions learned by the diffusion process.

Gradient of Score Distillation Sampling

In practical implementations, the gradient $\nabla_\theta \mathcal{L}_{\text{SDS}}$ is computed by omitting the Jacobian term associated with the denoising model from the gradient $\nabla_\theta \mathcal{L}_{\text{diff}}$. This simplification is intentional to avoid the computational complexity of backpropagating through the denoising model. The gradient of the original SDS loss is defined as follows:

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, I_\theta) \triangleq \mathbb{E}_{t,\epsilon}\left[(\hat{\epsilon}_\phi(I_t, y, t) - \epsilon)\frac{\partial I_\theta}{\partial \theta}\right] \tag{2}$$

The gradient provides the direction in which we need to change θ to make the generated images more like those from the distribution learned by the diffusion model.

Score Debiasing

However, using this SDS loss may sometimes result in the well-known "multiface issue", often referred to as the Janus problem, as discussed in [2]. This problem arises when the unconditional score, modeled by 2D diffusion models, is inherently biased towards certain viewing directions during the initial stages of optimizing a 3D volume. Such biases can adversely impact the consistency and realism of the 3D objects generated (see Sec. 4), particularly if specific views are disproportionately represented in the 2D data distribution for certain objects.

To solve this problem, we integrate a dynamic clipping gradient strategy inspired by [2] into our original gradient calculation ($\nabla_\theta \mathcal{L}_{\text{SDS}}$). This strategy dynamically constrains the score to decrease the effect of bias and artifacts in the predicted 2D scores:

$$\begin{gathered}\psi_{\text{dynamic}} := (1-\tau)\psi_{\text{start}} + \tau\psi_{\text{end}},\\ \nabla_\theta \mathcal{L}_{\text{SDS}}^{\text{clipped}}(\phi, I_\theta) = \text{Clip}(\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, I_\theta), \psi_{\text{dynamic}}),\end{gathered} \tag{3}$$

where $\tau = \frac{\text{step}}{\text{max step}}$.

3.3. Loss Functions

The user indicates whether their modification requires removal or addition, directing the system to apply the appropriate loss function (in addition to the SDS loss). For local additions and modifications, we adapt the preservation and silhouette loss from SKED [5]. For local removals, we introduce a new loss function to handle part exclusion.

3.3.1 Local Addition/Modification

1. **Preservation Loss, $\mathcal{L}_{\text{pres}}$:** $\mathcal{L}_{\text{pres}}$ constrains the NeRF model F_e to ensure that its density and color outputs match those of the base NeRF model F_o in areas outside the sketched regions. It is defined as:

$$\mathcal{L}_{\text{pres}} = \frac{1}{K}\sum_{i=1}^{K} w_i \left[\text{CE}(\bar{\alpha}_e, \bar{\alpha}_o) + \lambda_c \bar{\alpha}_o \|\bar{c}_e - \bar{c}_o\|^2\right], \tag{4}$$

where w_i adjusts the weight of each sample based on proximity to a masked region when a 3D point is projected to a sketch camera view, the Cross-Entropy term $\text{CE}(\bar{\alpha}_e, \bar{\alpha}_o)$ ensures prediction accuracy by comparing the predicted density distribution to the original density, and the regularization term $\lambda_c \bar{\alpha}_o \|\bar{c}_e - \bar{c}_o\|^2$ maintains color consistency, penalizing color deviations from the original model.

2. **Silhouette Loss, $\mathcal{L}_{\text{sil}}$:** Concurrently, $\mathcal{L}_{\text{sil}}$ ensures that the object mask occupies the regions delineated by the sketches by maximizing the opacity (alpha values) in

target mask regions. The loss function is:

$$\mathcal{L}_{\text{sil}} = \frac{1}{H \cdot W \cdot N} \sum_{j=1}^{N} \sum_{i=1}^{H \times W} -\mathbb{I}_{M_j}(x_i) C_j^{\alpha}(x_i) \quad (5)$$

3.3.2 Local Exclusion/Modification

1. **Inverted Silhouette Loss, $\mathcal{L}_{\text{inv-sil}}$:** To address the part exclusion issue in the original SKED model, we propose an Inverted Silhouette Loss, $\mathcal{L}_{\text{inv-sil}}$, which penalizes density in masked regions of the edited NeRF, with the opposite objective of the silhouette loss. The loss is computed as:

$$\mathcal{L}_{\text{inv-sil}} = \frac{1}{H.W.N} \sum_{j=1}^{N} \sum_{i=1}^{H \times W} \mathbb{I}_{M_j}(x_i) C_j^{\alpha}(x_i) \quad (6)$$

   where H and W are the dimensions of the rendered object masks, $\mathbb{I}_{M_j}$ is a binary pixel indicator for the exclusion zone (the part to remove), and C_j^{α} is the alpha mask rendered per sketch with $F_e(\mathbf{P}, r; \phi)$. This loss function minimizes alpha values in specified sketch regions to ensure they remain unrendered by the NeRF model.

4. Experiments and Results

4.1. Data

We work with the 3D model of a black cat as provided by the authors of SKED in order to generate comparisons between the baseline SKED model and our improved model. We create our own sketches and text prompts to use for experiments on both models. The model uses a zero-shot approach for learning, so our experiments do not require any additional training data.

In the following sections, we describe the various features enabled by our model and discuss the issues the original SKED model had with the same tasks. We will show the rendered results and discuss the qualitative differences in our results and the baseline's. We also create a separate pipeline using DreamGaussian [11] and Stable Diffusion [10] to generate separate comparisons from an alternative method. This alternative method uses the same inputs as our experiments (sketches on a rendered view of a 3D model), and uses Stable Diffusion to inpaint the sketch before using DreamGaussian to render a 3D model.

4.2. Additive Edits

We first experiment with adding parts/objects to the existing 3D model. All of the original experiments shown in the SKED paper [5] are additive edits, consisting of adding an accessory or overwriting part of the original 3D model.

4.2.1 "A cat with rabbit ears"

We use a prompt to add rabbit ears to the base NeRF model of a black cat. From the results in Figure 2, we can see that the baseline SKED result suffers from the multi-face issue, where it renders four ears (this shows from both the front and side views). With score debiasing, our model is able to avoid this and clearly produce two separate ears. However, we can see that only the DreamGaussian model was able to infer that the ear color should be the same as the rest of the cat, leading to a more realistic rendering.

4.2.2 "A cat with a cup on its back"

We attempt to add a completely new object to the scene with the cat by adding a cup on its back. From the corresponding column in Figure 2, we see that the baseline SKED model becomes confused and is unable to generate a new object and also generates artifacts outside the sketch region near the cat's head. We hypothesize that it struggles with objects that aren't related to the original model and don't overlap any part of it, leading to these results. With score debiasing however, we see a recognizable cup from our model, while DreamGaussian produces a cup overlapping the inside of the cat.

4.3. Deletions

A noticeable issue with the baseline SKED model was that it could not delete parts of the original 3D model as edits. To alleviate this issue, we introduced an inverted silhouette loss that penalized density inside masked regions.

4.3.1 "A cat with no tail"

To show the effects of our added loss function, we attempt to remove the tail from the cat model. We use a mask that covers the tail from two views, and from Figure 2, we can see that our model successfully removes the entire tail without any remaining artifacts, while both the original SKED and DreamGaussian models were unable to do so.

4.4. Replacements and Iterative Edits

Overlapping edits that would require deleting some part of the original model are also out of scope of the original SKED model's capabilities. Our model allows for this through iterative edits, illustrated in Figure 3.

4.4.1 "A cat with a rabbit tail"

An example of an "overlapping edit" is to replace the cat's tail with a rabbit tail. The baseline SKED model struggles with this as it cannot remove the original tail, and begins to generate some vague artifacts around the base of the tail, but fails to create anything substantial. By first removing the

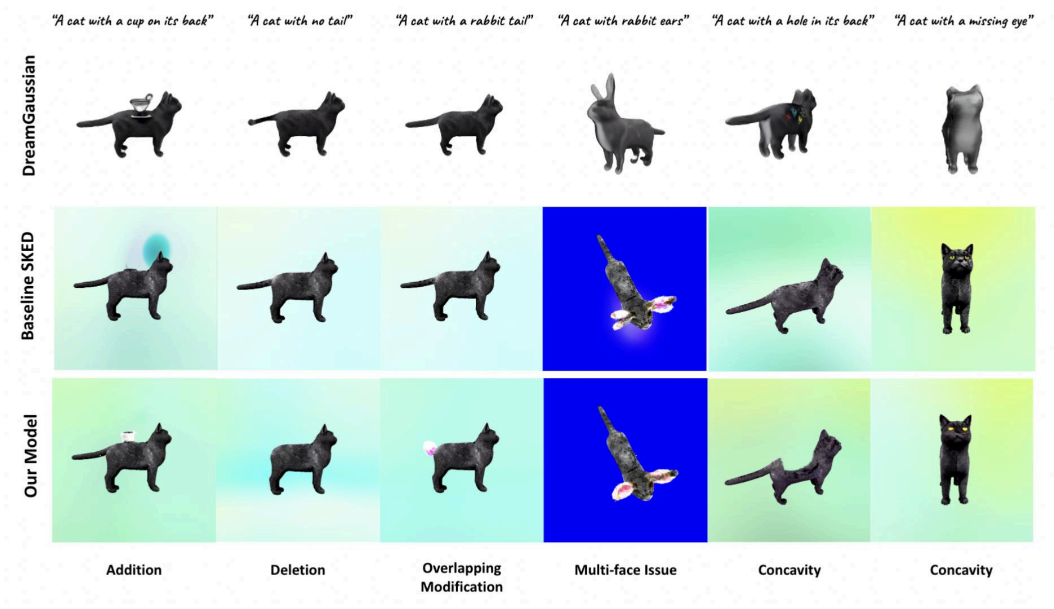

Figure 2. Comparison of results from DreamGaussian, SKED baseline and our model I-SKED for each experiment.

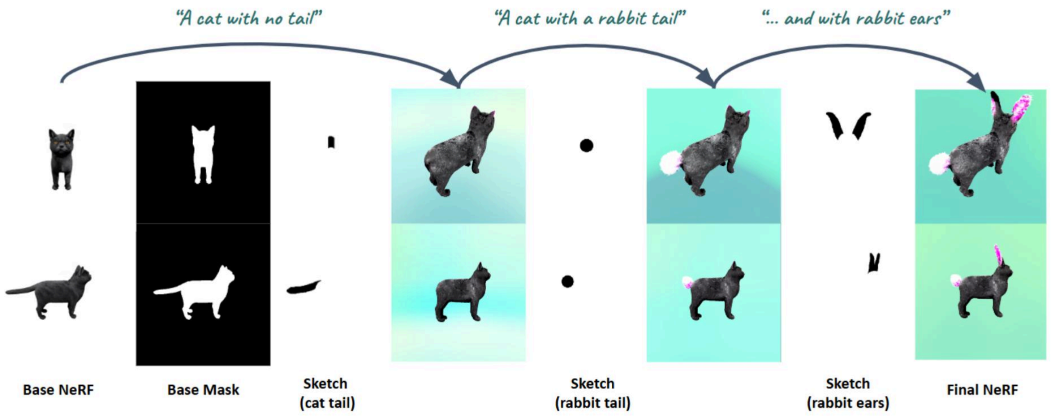

Figure 3. Our method allows for the replacement of the cat's tail through iterative edits and performs well with multiple rounds of edits.

cat's tail, then iterating on the result and adding a rabbit tail, our model is able to replace parts of the original model that the comparative methods could not. We also show in Figure 3 that we can continue to iterate and preserve changes from each iteration.

4.5. Concavities

A use case that our model struggles with is carving concavities into existing models. Concavities prove especially difficult to handle when they aren't visible as full deletions from any view. For the following examples, we can see that the experiments weren't fully successful for any method, making this an avenue to explore for future work.

4.5.1 "A cat with a hole in its back"

We try to put a bowl-shaped concavity in the cat's back by experimenting with various prompts. The most successful prompt was "A cat with a hole in its back," which completely deletes part of the cat's body. DreamGaussian renders a cat with an ambiguous texture on its midriff, and the

Table 1. Comparison of model performance on various editing tasks. Scores are displayed, with percentage of votes for 1st place (best result) displayed in parentheses.

Model	**No tail**	**Rabbit tail**	**Rabbit ears**	**Missing eye**	**Hole**	**Cup**
DreamGaussian	26.5 (5.6%)	16.9 (5.6%)	33.8 (40.0%)	**48.0 (65.7%)**	22.4 (11.1%)	39.5 (39.5%)
Baseline SKED	17.2 (11.1%)	22.9 (5.6%)	27.5 (20.0%)	18.6 (8.6%)	20.5 (5.6%)	13.3 (5.3%)
I-SKED	**56.3 (83.3%)**	**60.2 (88.9%)**	**38.7 (40.0%)**	33.3 (25.7%)	**57.1 (83.3%)**	**47.2 (55.3%)**

baseline SKED model fails to make any changes. We believe that since we use a side and front view for sketches, an accurate mask would be empty for both views since a concavity in the cat's back would not be visible. However, not providing a mask with any information for the model does not perform well, and we believe more investigation and future work is needed to address concavities that do not translate well to the masks used in our pipeline.

4.5.2 "A cat with a missing eye"

An additional experiment we run to test concavities is to remove an eye from the cat. While the baseline model makes no changes, our model slightly changes the side of the face of the cat, and more noticeably removes the pupils from the cat's eyes. The DreamGaussian model paints over the cat's eyes to essentially remove them, which could be seen as a more successful attempt than our model's. Similar to the previous example, we believe that creating a well-defined concavity even on this small scale is difficult without many more camera views.

4.5.3 Quantitative Results

Survey Method

We surveyed 32 people to evaluate the effectiveness of our improved Sketch-guided Text-based 3D Editing (I-SKED) model in comparison with the baseline SKED and Gaussian splatting methods. Participants were asked to rank the outputs of each model based on their alignment with the provided text prompts as shown in Figure 2. Scoring was implemented using a system where 10 points were awarded for a first-place ranking, 5 points for second place, and 1 point for third place. The scores were then aggregated to determine the overall preference and scaled to a max of 100 points. Additionally, the percentage of times each model was ranked first was calculated to provide a clear measure of its preference.

Survey Results

Table 1 summarizes the scores and percentages for each model's results being ranked first. The results show that the results from our I-SKED model were preferred by the majority of participants, achieving the highest scores and percentages across most editing tasks. This demonstrates a significant enhancement in user satisfaction and performance over the original SKED and Gaussian-splatting methods, more effectively meeting user expectations for 3D model editing.

Certain examples had competing results from the DreamGaussian model, which painted over the missing eye (a more convincing result than the missing pupils in our model's render) and generated rabbit ears that matched the color of the cat. These cases fall under future work and we believe that additional experimentation could help with enforcing concavities and additional debiasing could help with issues like color consistency.

5. Conclusions and Future Work

In conclusion, our enhanced I-SKED framework successfully addresses significant limitations of previous NeRF-based 3D model editing tools by enabling precise deletions and handling complex, non-overlapping edits. By incorporating score debiasing, we also improve the quality of generated 3D models and avoid issues such as the Janus problem. Our improvements result in a more complete pipeline for 3D model editing that lights up many more use cases, allowing users to intuitively make effective modifications to existing models.

Future work on this project will include continuing to improve the quality of generated models, as well as expanding the variety of edits allowed by the pipeline. We plan to improve the handling of concave modifications, enabling more complex transformations such as carving out internal structures in NeRF models. This will allow for more detailed and functional designs. We also aim to address the limitations posed by image priors in our current diffusion model. Integrating more advanced commercial diffusion models or modifying our loss functions to enforce color and textural consistency could help ensure that modifications fully align with the original model. We also plan to do an additional investigation into alternative/additional debiasing methods such as prompt debiasing [2] and the Perp-Neg [1] algorithm to further improve the quality of our generated models.

These developments are expected to significantly refine the tool's functionality and user experience in 3D content creation.

References

[1] Mohammadreza Armandpour, Ali Sadeghian, Huangjie Zheng, Amir Sadeghian, and Mingyuan Zhou. Re-imagine the Negative Prompt Algorithm: Transform 2D Diffusion into 3D, alleviate Janus problem and Beyond, Apr. 2023. arXiv:2304.04968 [cs]. 6

[2] Susung Hong, Donghoon Ahn, and Seungryong Kim. Debiasing Scores and Prompts of 2D Diffusion for View-consistent Text-to-3D Generation, Dec. 2023. arXiv:2303.15413 [cs]. 1, 2, 3, 6

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering, Aug. 2023. arXiv:2308.04079 [cs]. 2

[4] kiui. ashawkey/stable-dreamfusion, June 2024. original-date: 2022-10-06T06:18:39Z. 3

[5] Aryan Mikaeili, Or Perel, Mehdi Safaee, Daniel Cohen-Or, and Ali Mahdavi-Amiri. SKED: Sketch-guided Text-based 3D Editing, Aug. 2023. arXiv:2303.10735 [cs]. 1, 2, 3, 4

[6] 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, Mar. 2020. 2

[7] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, July 2022. 2

[8] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. 2022. 2, 3

[9] Maxime Raafat. BlenderNeRF, May 2023. 2

[10] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, Apr. 2022. arXiv:2112.10752 [cs]. 1, 2, 4

[11] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation, Mar. 2024. arXiv:2309.16653 [cs]. 1, 2, 4