# SKED: Sketch-guided Text-based 3D Editing: Supplementary material

## A. Background

In the following, we include an extended background chapter cut off from the main paper for brevity.

### A.1. Latent diffusion models (LDMs)

LDMs [?] are a class of diffusion models that operate on a latent space instead of directly sampling high-resolution color images. These models have two main components: a variational autoencoder consisting of an encoder $\mathcal{E}(x)$ and a decoder $\mathcal{D}(z)$, pretrained on the training data, and a denoising diffusion probabilistic model (DDPM) trained on the latent space of the autoencoder. Specifically, let $Z$ be the latent space learned by the autoencoder. The objective of the DDPM is to minimize the following expectation:

$$\mathbb{E}_{z_0 \sim Z, \epsilon \sim \mathcal{N}(0,I), t}[||\epsilon_\phi(z_t, t) - \epsilon||^2], \tag{1}$$

where $t$ is the time-step of the diffusion process, $z_t = \sqrt{\alpha_t} z_0 + \sqrt{1-\alpha_t}\epsilon$ is the input latent image with noise added to it, and $\epsilon_\phi$ is the denoising model, often constructed as a U-Net [?]. Once trained, it is possible to sample from the latent space $Z$ by starting from a random standard Gaussian noise and running the backward diffusion process as described by Ho et al. [?]. The sampled latent image then can be fed to $D(z)$ to get a final high-resolution image.

### A.2. Score distillation sampling (SDS)

First introduced by DreamFusion [?], SDS is a method of generating gradients from a pretrained diffusion model, by using its *Score Function* to push the outputs of a parameterized image model towards the mode of the diffusion model distribution. More formally, let $I_\theta$ be an image model with parameters $\theta$. In the case of our application, $I_\theta$ is a neural renderer such as NeRF [?] or Instant-NGP [?]. We can use a pretrained diffusion model with denoiser $\epsilon_\phi(z_t, t)$, to optimize the following:

$$\min_\theta \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I), t}[||\epsilon_\phi(\sqrt{\alpha_t} I_\theta + \sqrt{1-\alpha_t}\epsilon, t) - \epsilon||^2], \tag{2}$$

where $t$ is the time-step of the diffusion process, and $\alpha_t$ is a constant scheduling the diffusion forward and backward processes. The Jacobian of the denoiser can be omitted in the gradient of the above expression, to get:

$$\mathbb{E}_{\epsilon \sim \mathcal{N}(0,I), t}[(\epsilon_\phi(\sqrt{\alpha_t} I_\theta + \sqrt{1-\alpha_t}\epsilon, t) - \epsilon)\frac{\partial I_\theta}{\partial \theta}]. \tag{3}$$

The advantage of SDS is that one can apply constraints directly on the image model making this framework suitable for our application of sketch-guided 3D generation.

## B. Additional Evaluations

### B.1. Quantitative Comparisons

**Base Model Fidelity.** In Table 4, We include the SSIM metric to further quantify our method's capability to preserve the base model.

### B.2. Qualitative Comparisons

**Comparison to Latent-NeRF [?].** To the best of our knowledge, we are the first work to employ 2D sketch-based editing of NeRFs. Given that prior works are not directly comparable with our editing setting, we attempt to create a close comparison instead, faithful to the original compared method and fair to evaluate our editing setting. As baseline, we use the method from Latent-NeRF's [?] 3D sketch shape pipeline. We initialize a NeRF with the base object weights, and create a *3D sketch shape*, a mesh, by intersecting the bounding boxes of our 2D sketches in the 3D space. Note that we could also intersect the sketch masks, however, due to view inconsistencies, we found that the results are far inferior. After initializing the NeRF and creating the sketch shape, we proceed to use the sketch shape loss from the paper to preserve the geometry, while editing the NeRF according to the input text. In Fig. 11, we establish that while this baseline is able to perform meaningful edits, it suffers from two apparent issues: (i) the baseline severely changes the base NeRF, and (ii) the edited region is bound to the coarse geometry of the intersected bounding boxes. To alleviate the latter, one could resort to modeling 3D assets as a sketch shape. However, we show that by using simple multiview sketches, it is possible to perform local editing without going through the effort of modeling accurate 3D masks. Finally, we include a quantitative summary of the preservation ability and the performance of the two methods in Table 7.

Table 4: Fidelity of base field. We measure the **Structural Similarity (SSIM ↑)** of the method's output against renderings from the base model. SKED *(no-preserve)* refers to a variant of our method which doesn't apply $\mathcal{L}_{pres}$. Text-Only refers to a public re-implementation of Latent-NeRF [**?**]. Latent-NeRF uses the setting from Section B.2.

| Method | Cat | | Cupcake | | Horse | | Sundae | | Plant | | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *+chef hat* | | *+candle* | | *+horn* | | *+cherry* | | *+flower* | | |
| | A | B | A | B | A | B | A | B | A | B | |
| SKED | **0.978** | **0.990** | **0.964** | **0.973** | **0.990** | **0.986** | **0.963** | **0.962** | 0.927 | **0.938** | **0.967** |
| SKED *(no-preserve)* | 0.867 | 0.890 | 0.944 | 0.948 | 0.950 | 0.934 | 0.913 | 0.921 | 0.803 | 0.801 | 0.897 |
| Text-Only [**?**] | 0.875 | 0.918 | 0.937 | 0.943 | 0.933 | 0.908 | 0.947 | 0.951 | 0.891 | 0.883 | 0.919 |
| Latent-NeRF [**?**] | 0.915 | 0.948 | 0.950 | 0.956 | 0.947 | 0.927 | 0.904 | 0.906 | **0.930** | 0.925 | 0.930 |

Table 5: Fidelity of base field. We measure the **Perceptual Image Patch Similarity (LPIPS ↓)** of the method's output against renderings from the base model. We use VGG [**?**] as the learned perceptual encoder. SKED *(no-preserv)* refers to a variant of our method which doesn't apply $\mathcal{L}_{pres}$. Text-Only refers to a public re-implementation of DreamFusion [**?**]. Latent-NeRF uses the setting from Section B.2.

| Method | Cat | | Cupcake | | Horse | | Sundae | | Plant | | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *+chef hat* | | *+candle* | | *+horn* | | *+cherry* | | *+flower* | | |
| | A | B | A | B | A | B | A | B | A | B | |
| SKED | **0.070** | **0.069** | 0.069 | **0.061** | **0.028** | **0.032** | 0.086 | 0.094 | 0.158 | 0.128 | **0.079** |
| SKED *(no-preserv)* | 0.290 | 0.250 | 0.091 | 0.093 | 0.089 | 0.098 | 0.169 | 0.154 | 0.291 | 0.309 | 0.183 |
| Text-Only [**?**] | 0.150 | 0.137 | 0.076 | 0.076 | 0.115 | 0.134 | **0.081** | **0.079** | 0.170 | 0.180 | 0.120 |
| Latent-NeRF [**?**] | 0.102 | 0.101 | **0.066** | 0.065 | 0.081 | 0.100 | 0.139 | 0.141 | **0.108** | **0.113** | 0.101 |

Table 6: Fidelity of base field. Following the experiments in section **??**, we measure the PSNR of the base objects on additional examples provided in Fig. **??** and Fig. **??**.

| | Tree to Cactus | | Anime+Skirt | | Pancake+Cream | | Gift on Table | | Mean |
|---|---|---|---|---|---|---|---|---|---|
| Method | view 1 | view 2 | view 1 | view 2 | view 1 | view 2 | view 1 | view 2 | |
| SKED | **29.15** | **27.47** | **39.67** | **37.40** | **27.48** | **26.64** | **34.16** | **31.52** | **31.68** |
| Text-Only | 23.12 | 24.40 | 22.61 | 21.95 | 16.97 | 15.35 | 19.05 | 20.70 | 20.51 |

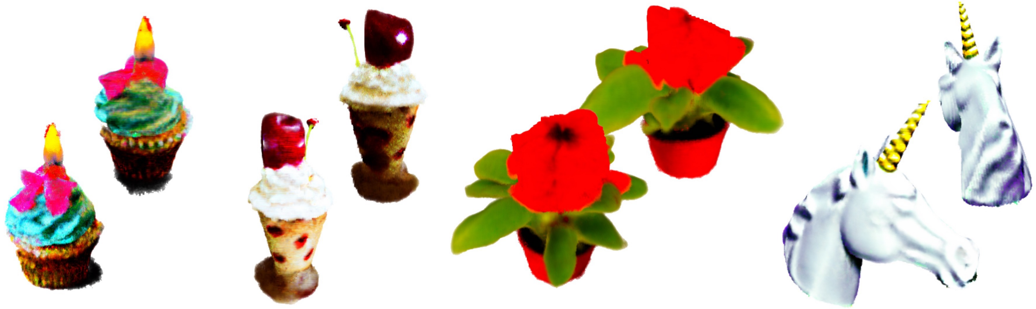
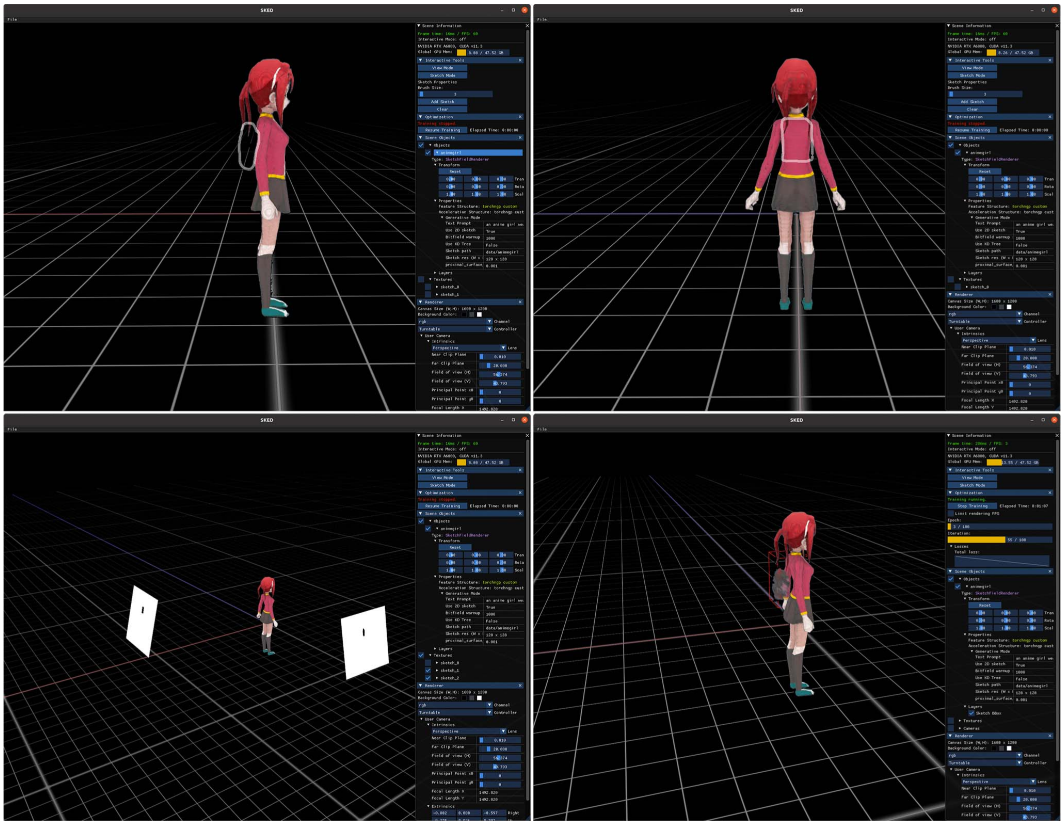


Figure 11: Examples from the modified version of the sketch shape pipeline of Latent-NeRF [**?**]

Table 7: To compare our method's ability to preserve the base with the baseline derived from Latent-NeRF [**?**], we measure the PSNR of both method's outputs against renderings from the base model. Additionally, we report the average runtime of our method compared to the baseline.

| Method | Cat<br>+*chef hat*<br>A | <br><br>B | Cupcake<br>+*candle*<br>A | <br><br>B | Horse<br>+*horn*<br>A | <br><br>B | Sundae<br>+*cherry*<br>A | <br><br>B | Plant<br>+*flower*<br>A | <br><br>B | PSNR<br>Mean | Runtime (minutes)<br>Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SKED | **31.05** | **34.13** | **23.73** | **25.98** | **32.45** | **31.46** | **26.47** | **25.99** | **21.71** | **22.31** | **27.53** | **38** |
| Latent-NeRF [?] | 21.15 | 22.62 | 21.99 | 21.20 | 17.00 | 15.97 | 16.07 | 15.47 | 17.66 | 16.78 | 18.59 | 64 |



Frame time: 16ms / FPS: 60
Interactive Mode: off
NVIDIA RTX A6000, CUDA v11.3
Global GPU Mem: 8.00 / 47.52 GB
Sketch Properties
Brush Size: 3
Training stopped.
Elapsed Time: 0:00:00
animegirl
Type: SketchFieldRenderer
Transform
Tran 0.00 0.00 0.00
Rota 0.00 0.00 0.00
Scal 1.00 1.00 1.00
Properties
Feature Structure: torchngp custom
Acceleration Structure: torchngp cust
Generative Mode
Text Prompt: an anime girl we
Use 2D sketch: True
Bitfield warmup: 1000
Use KD Tree: False
Sketch path: data/animegirl
Sketch res (W x H): 128 x 128
proximal_surface: 0.001
Canvas Size (W,H): 1600 x 1200

Figure 12: The interactive UI allows users to sketch over a pretrained NeRF. **Top row**: The user draws scribbles from two different views using "Sketch Mode". **Bottom left**: After pressing "Add sketch", the scribbles are filled to generate masks, ready to be used with our pipeline. **Bottom right**: The bounding box marks the sketches intersection region, where the edit takes place.

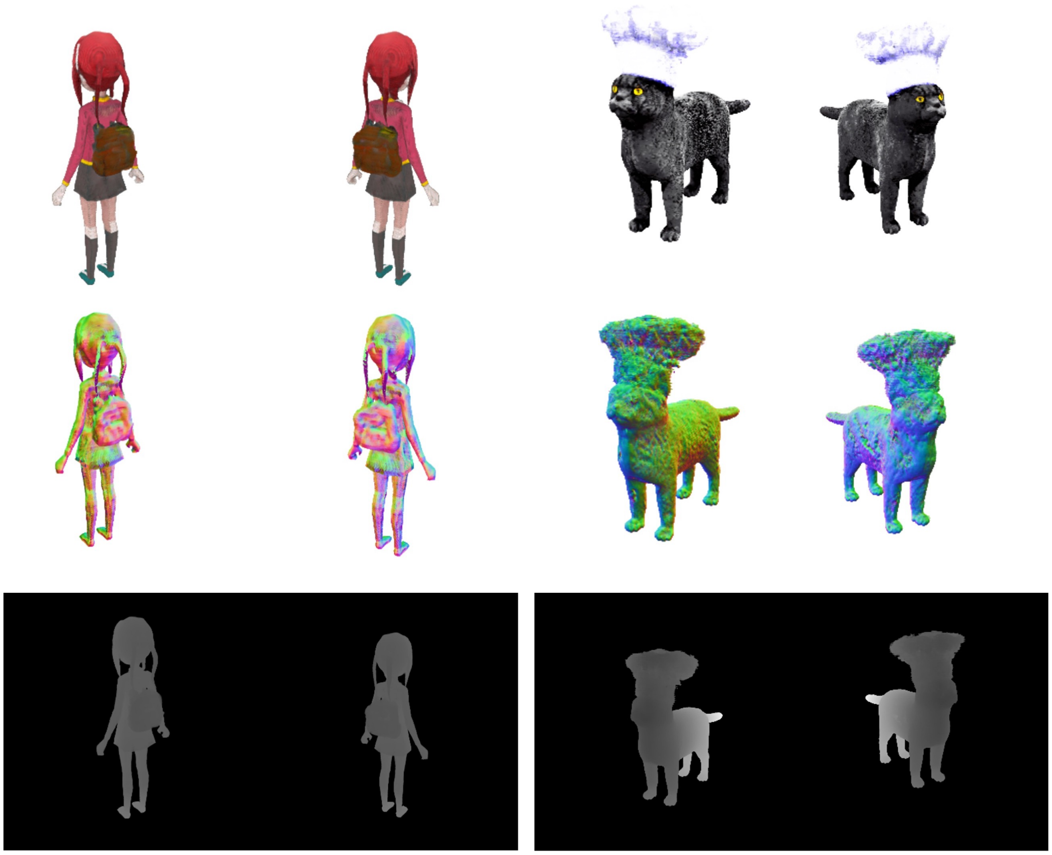


Figure 13: From top to bottom: color, normal and depth maps of outputs generated by our method.

# C. Implementation Details

This section contains additional implementation details omitted from the manuscript.

## C.1. Text Prompts

In the following, we include the full list of prompts that were used to generate the examples within the paper.

- ”A cat wearing a chef hat”
- ”A cherry on top of a sundae”
- ”A red flower stem rising from a potted plant”
- ”A teddy bear wearing sunglasses”
- ”A candle on top of a cupcake”
- ”An anime girl wearing a brown bag”
- ”An apple on a plate”
- ”A Nutella jar on a plate”
- ”A globe on a plate”
- ”A tennis ball on a plate”
- ”A cat wearing a red tie”
- ”A cat wearing red tie wearing a chef hat”
- ”A 3D model of a unicorn head”

Additionally, similar to DreamFusion[**?**] we use directional prompts, where based on the rendering view, we modify prompt **T** as follows:

- ”**T**, overhead view”
- ”**T**, side view”
- ”**T**, back view”
- ”**T**, bottom view”
- ”**T**, front view”

## C.2. Interactive UI

Since our method requires user interaction, we include an interactive user interface with our implementation (Fig. 12). The user interface allows users to optimize newly reconstructed base NeRF models, or load pretrained ones. To perform edits, users can position the camera on the desired sketch view, and draw scribbles to guide SKED. By pressing ”Add Sketch”, scribbles are filled and converted to masked sketch inputs, ready to be used with our method.

## C.3. Quality Notes

Our implementation uses an early version of Stable-DreamFusion [**?**] which does not include the optimizations very recently suggested by Magic3D [**?**]. In contrast to DreamFusion [**?**] and Magic3D [**?**], which use commercial diffusion models with larger language models [**?**, **?**], we rely on Stable Diffusion [**?**], which is less sensitive to directional prompts. Our results are therefore not comparable in visual quality to these previous works.

# D. Additional Assets

## D.1. Geometry and Depth

In addition to RGB images, we share examples highlighting the geometry of our method’s outputs. In Fig. 13 we include the normal maps and depth maps of two output samples.