Posterior Distillation Sampling

Juil Koo[1] Chanho Park[2] Minhyuk Sung[1]
[1]KAIST [2]Dankook University

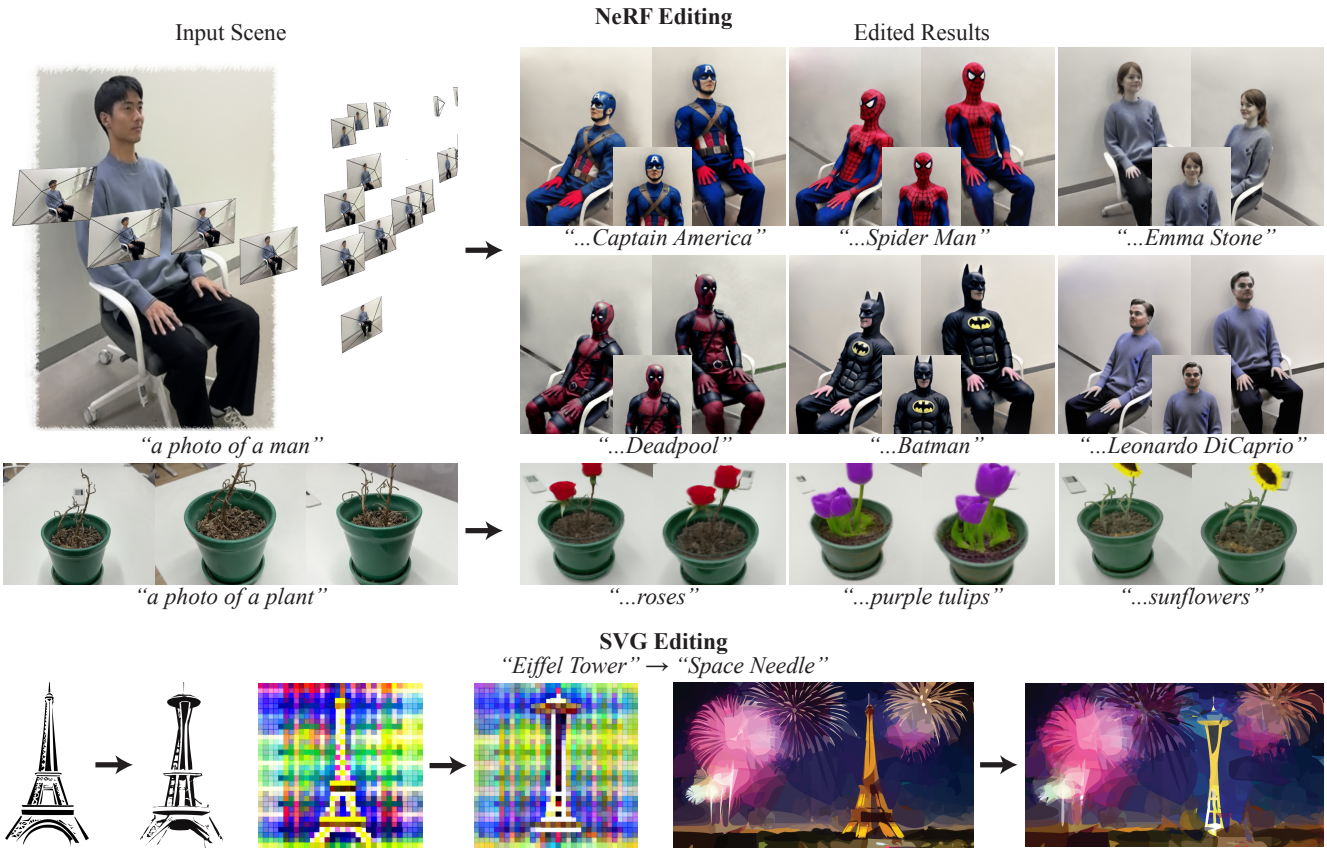


Figure 1. **Parametric image editing results obtained by Posterior Distillation Sampling (PDS).** PDS is an optimization tailored for editing across diverse parameter spaces. It preserves the original details of the source content while aligning them with the input texts.

Abstract

We introduce Posterior Distillation Sampling (PDS), a novel optimization method for parametric image editing based on diffusion models. Existing optimization-based methods, which leverage the powerful 2D prior of diffusion models to handle various parametric images, have mainly focused on generation. Unlike generation, editing requires a balance between conforming to the target attribute and preserving the identity of the source content. Recent 2D image editing methods have achieved this balance by leveraging the stochastic latent encoded in the generative process of diffusion models. To extend the editing capabilities of diffusion models shown in pixel space to parameter space, we reformulate the 2D image editing method into an optimization form named PDS. PDS matches the stochastic latents of the source and the target, enabling the sampling of targets in diverse parameter spaces that align with a desired attribute while maintaining the source's identity. We demonstrate that this optimization resembles running a generative process with the target attribute, but aligning this process with the trajectory of the source's generative process. Extensive editing results in Neural Radiance Fields and Scalable Vector Graphics representations demonstrate that PDS is capable of sampling targets to fulfill the aforementioned balance across various parameter spaces. See our project page posterior-distillation-sampling.github.io for more results.

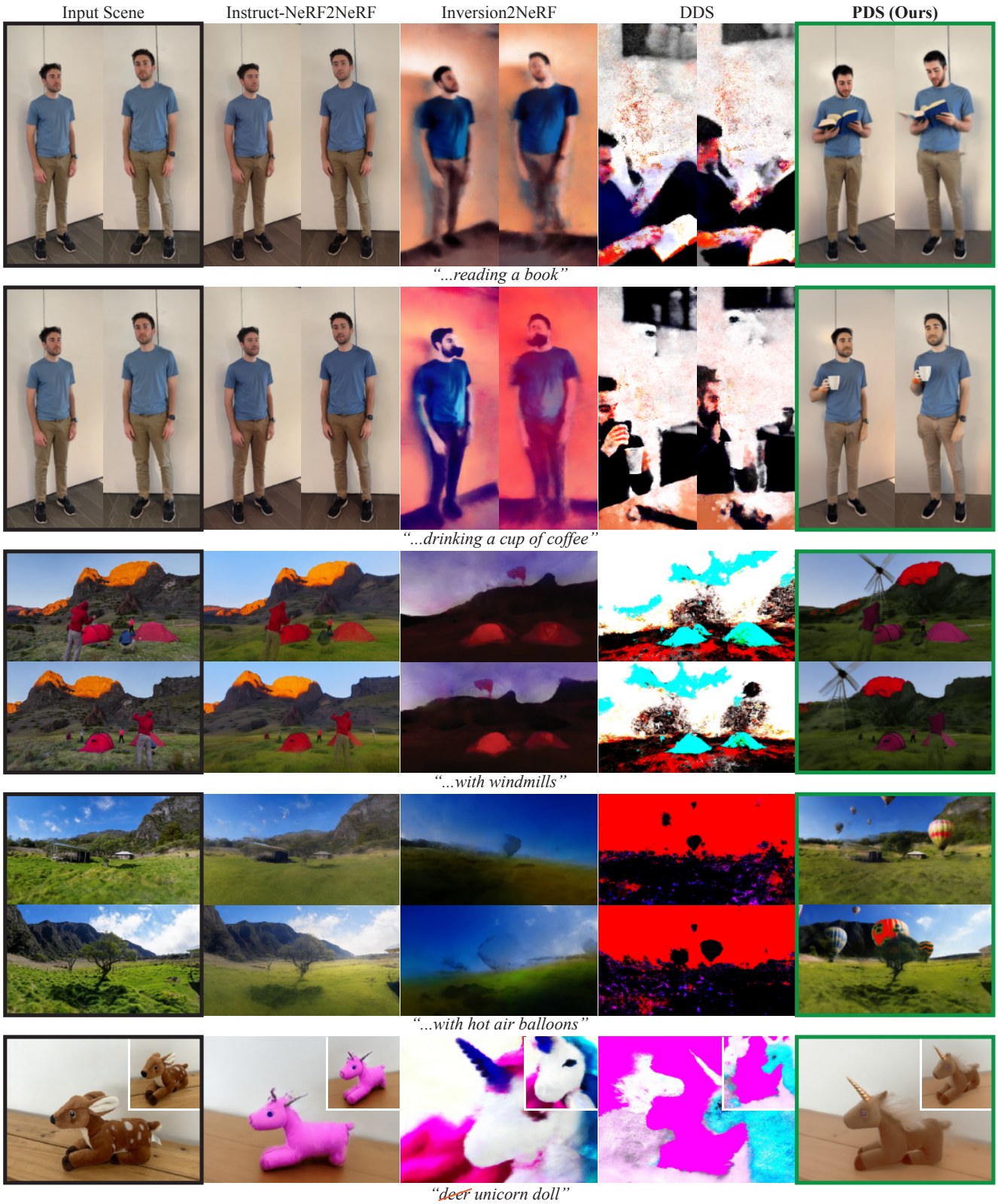


Figure 2. **A comparison of 3D scene editing between PDS and other baselines.** Given input 3D scenes on the left, PDS, marked by green boxes on the rightmost side, successfully performs complex editing, such as geometric changes and adding objects, according to the input texts. On the other hand, the baselines either fail to change the input 3D scenes or produce results that greatly deviate from the input scenes, losing their identity.

1. Introduction

Diffusion models [13, 45–48] have recently led to rapid development in text-conditioned generation and editing across diverse domains, including 2D images [11, 15, 20, 49, 52], 3D objects [18, 19, 21, 32], and audio [7, 14, 55]. Among them, in particular, 2D image diffusion models [5, 26, 37, 39, 41] have demonstrated their powerful generative prior aided by Internet-scale image and text datasets [3, 42, 43]. Nonetheless, this rich 2D generative prior has been confined to pixel space, limiting their broader applicability. A pioneer work overcoming this limitation, DreamFusion [34], has introduced Score Distillation Sampling (SDS). It leverages the generative prior of text-to-image diffusion models to synthesize 3D scenes represented by Neural Radiance Fields (NeRFs) [28] from texts. Beyond NeRF representations [4, 23, 36, 44, 50, 51, 57], SDS has been widely applied to various parameter spaces, where images are not represented by pixels but specific parameterizations, such as texture [1, 25], material [54] and Scalable Vector Graphics (SVGs) [16, 17, 53].

While SDS [34] has achieved great advances in generating parametric images, editing is also an essential element for full freedom in handling visual content. Editing differs from generation in that it requires considerations of both the target text and the original source content, thereby emphasizing two key aspects: (1) alignment with the target text prompt and (2) preservation of the source content's identity. To extend SDS, which lacks the latter aspect, Hertz *et al.* [10] propose Delta Denoising Score (DDS). DDS reduces the noisy gradients inherent in SDS, leading to better-maintaining background details and sharper editing outputs. However, the optimization function of DDS still lacks an explicit term for identity preservation.

To address the absence of preserving the source's identity in SDS [34] and DDS [10], we turn our attention to a recent 2D image editing method [15, 52] based on diffusion models, known as stochastic diffusion inversion. Their primary objective is to compute the stochastic latent of an input image within the generative process of diffusion models. Once the stochastic latent of a source image is computed, the source image can be edited by running a generative process with new conditions, such as new target text prompts, while feeding the source's stochastic latent into the process. Feeding the source's stochastic latent into the target image's generative process ensures that the target image maintains the structural details of the source while moving towards the direction of the target text. Thus, this editing process reflects the aforementioned two key aspects of editing.

To extend the editing capabilities of the stochastic diffusion inversion method from pixel space to parameter space, we reformulate this method into an optimization form named Posterior Distillation Sampling (PDS). Unlike SDS [34] and DDS [10], which match two noise variables, PDS aims to match the stochastic latents of the source and the optimized target. We demonstrate that our optimization process resembles aligning forward process posteriors of the source and the target, ensuring that the target's generative process trajectory does not significantly deviate from that of the source.

When parametric images come from NeRF [28], Haque *et al.* [9] have recently introduced a promising text-driven NeRF editing method called Iterative Dataset Update (Iterative DU). To edit 3D scenes, it performs an editing process in 2D space bypassing direct edit in 3D space. Thus, when a text prompt induces large variations in 2D space across different views, it has difficulty producing the right edit in 3D space. On the other hand, our method directly updates NeRF in 3D space, thus gradually transforming a 3D scene into its edited version in a view-consistent manner even in the case where text prompts induce large variations, such as large geometric changes or the addition of objects to unspecified regions.

Our extensive editing experiment results, including NeRF editing (Section 6.1) and SVG editing (Section 6.2), demonstrate the versatility of our method for parametric image editing. In NeRF editing, we are the first to produce large geometric changes or to add objects to arbitrary regions without specifying local regions to be edited. Figure 2 shows these examples. Qualitative and quantitative comparisons of SVG editing with other optimization methods, namely SDS [34] and DDS [10], have demonstrated that PDS produces only the necessary changes to source SVGs, effectively aligning them with the target prompts.

2. Related Work

2.1. Score Distillation Sampling

Following the remarkable success of diffusion models in text-to-image generation, there have been attempts to leverage the 2D prior of diffusion models for various other types of generative tasks. In these tasks, images are represented through rendering processes with specific parameters, including Neural Radiance Fields [17, 34, 50], texture [1, 25], material [54] and Scalable Vector Graphics (SVGs) [16, 17, 53]. The primary method employed in these tasks is Score Distillation Sampling (SDS). SDS is an optimization approach that updates the rendering parameter towards the image distribution of diffusion models by enforcing the noise prediction on noisy rendered images to match sampled noise. Concurrently, Wang *et al.* [50] also have introduced Score Jacobian Chaining which converges toward a similar algorithm as SDS but from a different mathematical derivation. Wang *et al.* [51] have proposed Variational Score Distillation (VSD) to address oversaturation, over-smoothing, and low-diversity problems in SDS [34]. Instead of updating a single data point, VSD

updates multiple data points to align an optimized distribution with the diffusion model's image distribution. Zhu and Zhuang [57] use more accurate predictions of diffusion models via iterative denoising at every SDS update step.

When it comes to editing, Hertz *et al.* [10] propose Delta Denoising Score (DDS), an adaptation of SDS for editing tasks. It reduces the noisy gradient directions in SDS to better maintain the input image details. Nonetheless, its optimization function lacks an explicit term to preserve the identity of the input image, thus often producing outputs that significantly deviate from the input images. To alleviate this issue, we propose Posterior Distillation Sampling, a novel optimization approach that incorporates a term dedicated to preserving the identity of the source in its optimization function.

2.2. Text-Driven NeRF Editing

Haque *et al.* [9] have proposed a text-driven NeRF editing method, known as Iterative Dataset Update (Iterative DU). It iteratively replaces reference images, initially used for NeRF [28] reconstruction, with edited images using Instruct-Pix2Pix [2]. By applying a reconstruction loss with these iteratively updated images to an input NeRF [28] scene, the scene is gradually transformed to its edited counterpart. Mirzae *et al.* [29] improve Instruct-NeRF2NeRF [9] by computing local regions to be edited. However, this iterative image replacement method suffers from edits that involve large variations across different views, such as complex geometric changes or adding objects to unspecified regions. Thus, they have mainly focused on appearance changes.

Instead of the Iterative DU method, several recent works [22, 33, 58] directly apply SDS [34] or DDS [10] to NeRF editing. However, these optimizations do not fully consider the preservation of the source's identity and are thus prone to producing outputs that substantially diverge from the input scenes. In contrast, our novel optimization inherently guarantees the preservation of the source's identity, facilitating involved NeRF editing while maintaining the identity of the original scene.

2.3. Diffusion Inversion

Diffusion inversion computes the latent representation of an input image encoded in diffusion models. This allows for real image editing by finding the corresponding latent that can fairly reconstruct the given image. The computed latent is then decoded into a new image through a generative process. Using the deterministic generative process of Denoising Diffusion Implicit Models (DDIM) [46], one can approximately run the ODE of the generative process in reverse [6, 46], referred to as DDIM inversion. Several recent works have improved DDIM inversion by adjusting text features [8, 30, 31], introducing new cross-attention maps during a generative process [11] or alternatively coupling intermediate latents from two inversion trajectories [49]. Meanwhile, an alternative approach, known as DDPM inversion [15, 52], employs the stochastic generative process of Denoising Diffusion Probabilistic Models (DDPM) [13]. They focus on capturing the structural details of an input image encoded in its stochastic latent. We extend the editing capabilities of this DDPM inversion method to parameter space by reformulating the method into an optimization form.

3. Preliminaries

We first discuss existing optimization-based approaches to handle parametric images, then introduce our novel parametric image editing method in Section 4.

3.1. Score Distillation Sampling (SDS) [34]

Score Distillation Sampling (SDS) [34] is proposed to generate parametric images by leveraging the 2D prior of pretrained text-to-image diffusion models. Given an input data $\mathbf{x}_0$ and a text prompt y, the training objective function of diffusion models is to predict injected noise $\boldsymbol{\epsilon}$ using a noise predictor $\boldsymbol{\epsilon}_\phi$:

$$\mathcal{L}(\mathbf{x}_0) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\boldsymbol{\epsilon}}\left[w(t)\|\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, y, t) - \boldsymbol{\epsilon}\|_2^2\right], \quad (1)$$

where $w(t)$ is a weighting function and $\mathbf{x}_t$ results from the forward process of diffusion models:

$$\mathbf{x}_t := \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}) \quad (2)$$

with variance schedule variables $\bar{\alpha}_t := \prod_{s=1}^{t}\alpha_s$. When the input data $\mathbf{x}_0$ is generated by a differentiable image generator $\mathbf{x}_0 = g(\theta)$, parameterized by θ, SDS updates θ by back-propagating the gradient of Equation 1 while omitting the U-Net jacobian term $\frac{\partial\boldsymbol{\epsilon}_\phi}{\partial\mathbf{x}_t}$ for computation efficiency:

$$\nabla_\theta\mathcal{L}_{\text{SDS}}(\mathbf{x}_0 = g(\theta)) = \mathbb{E}_{t,\boldsymbol{\epsilon}}\left[w(t)(\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, y, t) - \boldsymbol{\epsilon})\frac{\partial\mathbf{x}_0}{\partial\theta}\right], \quad (3)$$

where we denote a noise prediction of diffusion models with classifier-free guidance [12] by $\boldsymbol{\epsilon}_\phi$ for simplicity. Through this optimization process, SDS is capable of generating a parametric image which conforms to the input text prompt y.

3.2. Delta Denoising Score (DDS) [10]

Even though SDS has been widely used for various parametric images, its optimization is designed for generation, thus it does not reflect one of the key aspects of editing: preserving the source identity.

To extend SDS to editing, Hertz *et al.* [10] have proposed Delta Denoising Score (DDS). Given source data $\mathbf{x}^{\text{src}}$

and its corresponding text prompt y^{src}, the goal of DDS is to synthesize new target data $\mathbf{x}^{\text{tgt}}$ that is aligned with a target text prompt y^{tgt}. In the SDS formula 3, DDS replaces randomly sampled noise $\boldsymbol{\epsilon}$ with a noise prediction given a source data-text pair $\boldsymbol{\epsilon}_\phi(\mathbf{x}_t^{\text{src}}, y^{\text{src}}, t)$:

$$\nabla_\theta \mathcal{L}_{\text{DDS}} = \mathbb{E}_{t,\boldsymbol{\epsilon}}\left[w(t)\left(\boldsymbol{\epsilon}_\phi(\mathbf{x}_t^{\text{tgt}}, y^{\text{tgt}}, t) - \boldsymbol{\epsilon}_\phi(\mathbf{x}_t^{\text{src}}, y^{\text{src}}, t)\right)\frac{\partial \mathbf{x}_0^{\text{tgt}}}{\partial \theta}\right], \quad (4)$$

where the same noise $\boldsymbol{\epsilon}$ is shared for $\mathbf{x}_t^{\text{src}}$ and $\mathbf{x}_t^{\text{tgt}}$:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ \mathbf{x}_t^{\text{src}} &= \sqrt{\bar{\alpha}_t}\mathbf{x}_0^{\text{src}} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \\ \mathbf{x}_t^{\text{tgt}} &= \sqrt{\bar{\alpha}_t}\mathbf{x}_0^{\text{tgt}} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}. \end{aligned} \quad (5)$$

While DDS extends SDS for editing tasks, it lacks an explicit term in its optimization to preserve the identity of the source. As a result, DDS is still prone to produce editing results that significantly deviate from the source.

3.3. Stochastic Latent in Generative Process

To achieve both conformity to the text and preservation of the source's identity, we turn our attention to the rich information encoded in the stochastic generative process of DDPM [13]. When $\beta_t := 1 - \alpha_t$ are small, it is well-known that the posterior of the forward process also follows a Gaussian distribution according to a property of Gaussians. The forward process posteriors are represented as:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}_t, \mathbf{x}_0), \sigma_t \mathbf{I}), \quad (6)$$

where $\sigma_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ and the posterior mean $\boldsymbol{\mu}$ is a linear combination of $\mathbf{x}_0$ and $\mathbf{x}_t$: $\boldsymbol{\mu}(\mathbf{x}_t, \mathbf{x}_0) := \gamma_t \mathbf{x}_0 + \delta_t \mathbf{x}_t$ with $\gamma_t := \frac{\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)}{1-\bar{\alpha}_t}$ and $\delta_t := \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$.

Since $\mathbf{x}_0$ is unknown during a generative process, we approximate $\mathbf{x}_0$ with a one-step denoised estimate as follows:

$$\tilde{\mathbf{x}}_0(\mathbf{x}_t, y; \boldsymbol{\epsilon}_\phi) := \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, y, t)). \quad (7)$$

Consequently, one step of the generative process is represented as follows:

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_\phi(\mathbf{x}_t, y; \boldsymbol{\epsilon}_\phi) + \sigma_t \mathbf{z}_t, \quad \mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad (8)$$

where $\boldsymbol{\mu}_\phi(\mathbf{x}_t, y; \boldsymbol{\epsilon}_\phi) = \gamma_t \tilde{\mathbf{x}}_0(\mathbf{x}_t, y; \boldsymbol{\epsilon}_\phi) + \delta_t \mathbf{x}_t$.

Using Equation 8, one can compute stochastic latent $\tilde{\mathbf{z}}_t$ that captures the structural details of $\mathbf{x}_0$. This involves computing $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ via the forward process and then rearranging Equation 8 as follows:

$$\tilde{\mathbf{z}}_t(\mathbf{x}_0, y; \boldsymbol{\epsilon}_\phi) = \frac{\mathbf{x}_{t-1} - \boldsymbol{\mu}_\phi(\mathbf{x}_t, y; \boldsymbol{\epsilon}_\phi)}{\sigma_t}. \quad (9)$$

Several recent works [15, 52], known as DDPM inversion, have utilized the stochastic latent for image editing tasks. To edit an image using $\tilde{\mathbf{z}}_t$, they first pre-compute $\tilde{\mathbf{z}}_t$ of the source image across all t in the generative process. They then run a new generative process with a new target prompt while incorporating the pre-computed $\tilde{\mathbf{z}}_t$ of the source into the process instead of randomly sampled noise $\mathbf{z}_t$.

Although these works [15, 52] have utilized the rich information encoded in $\tilde{\mathbf{z}}_t$ for an editing purpose, their applications have been limited within 2D-pixel space due to reliance on the generative process. In our work, we broaden the application of the stochastic latent to parameter space by reformulating the method as an optimization form, enabling *parametric image editing*.

4. Posterior Distillation Sampling

Here, we introduce Posterior Distillation Sampling (PDS), a novel optimization function designed for parametric image editing.

Our objective is to synthesize $\mathbf{x}_0^{\text{tgt}}$ that is aligned with y^{tgt} while it retains the identity of $\mathbf{x}_0^{\text{src}}$. To achieve this, we employ the stochastic latent $\tilde{\mathbf{z}}_t$ in our optimization. For simplicity, we denote the stochastic latents of the source and the target as follows:

$$\tilde{\mathbf{z}}_t^{\text{src}} := \tilde{\mathbf{z}}_t(\mathbf{x}_0^{\text{src}}, y^{\text{src}}; \boldsymbol{\epsilon}_\phi) \quad (10)$$

$$\tilde{\mathbf{z}}_t^{\text{tgt}} := \tilde{\mathbf{z}}_t(\mathbf{x}_0^{\text{tgt}}, y^{\text{tgt}}; \boldsymbol{\epsilon}_\phi). \quad (11)$$

Using the stochastic latents, we define a novel objective function as follows:

$$\mathcal{L}_{\tilde{\mathbf{z}}_t}(\mathbf{x}_0^{\text{tgt}} = g(\theta)) := \mathbb{E}_{t,\boldsymbol{\epsilon}_{t-1},\boldsymbol{\epsilon}_t}\left[\|\tilde{\mathbf{z}}_t^{\text{tgt}} - \tilde{\mathbf{z}}_t^{\text{src}}\|_2^2\right], \quad (12)$$

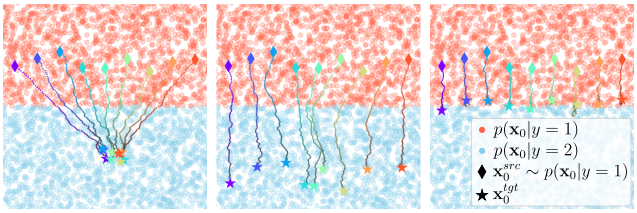
where, similar to Equation 5, $\tilde{\mathbf{z}}_t^{\text{src}}$ and $\tilde{\mathbf{z}}_t^{\text{tgt}}$ share the same noises, denoted by $\boldsymbol{\epsilon}_{t-1}$ and $\boldsymbol{\epsilon}_t$, when computing their respective $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$.

Rather than matching noise variables as in SDS [34] and DDS [10], we match the stochastic latents of the source and the target via the optimization. By taking the gradient of $\mathcal{L}_{\tilde{\mathbf{z}}_t}$ with respect to θ and ignoring the U-Net jacobian term as previous works [10, 34, 50], one can obtain PDS as follows:

$$\nabla_\theta \mathcal{L}_{\text{PDS}} := \mathbb{E}_{t,\boldsymbol{\epsilon}_t,\boldsymbol{\epsilon}_{t-1}}\left[w(t)(\tilde{\mathbf{z}}_t^{\text{tgt}} - \tilde{\mathbf{z}}_t^{\text{src}})\frac{\partial \mathbf{x}_0^{\text{tgt}}}{\partial \theta}\right]. \quad (13)$$

Expanding Equation 13, the following detailed formulation is derived:

$$\nabla_\theta \mathcal{L}_{\text{PDS}} := \mathbb{E}_{t,\boldsymbol{\epsilon}_t,\boldsymbol{\epsilon}_{t-1}}\left[(\psi(t)(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}) + \chi(t)(\hat{\boldsymbol{\epsilon}}_t^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_t^{\text{src}}))\frac{\partial \mathbf{x}_0^{\text{tgt}}}{\partial \theta}\right], \quad (14)$$



SDS DDS **PDS (Ours)**

Figure 3. **A visual comparison of the editing process through SDS [34], DDS [10] and PDS.** The figure illustrates the trajectories of samples drawn from $p(\mathbf{x}_0|y=1)$ as they are shifted towards $p(\mathbf{x}_0|y=2)$. PDS notably moves the samples near the boundary of the two marginals—the optimal endpoint in that it balances the necessary change with the original identity.

where $\hat{\boldsymbol{\epsilon}}_t^{\text{src}} := \boldsymbol{\epsilon}_\phi(\mathbf{x}_t^{\text{src}}, y^{\text{src}}, t)$ and $\hat{\boldsymbol{\epsilon}}_t^{\text{tgt}} := \boldsymbol{\epsilon}_\phi(\mathbf{x}_t^{\text{tgt}}, y^{\text{tgt}}, t)$. We leave a more detailed derivation to the **supplementary material**.

Matching $\mathbf{z}_t^{\text{tgt}}$ with $\mathbf{z}_t^{\text{src}}$ ensures that the posteriors of $\mathbf{x}_0^{\text{tgt}}$ and $\mathbf{x}_0^{\text{src}}$ do not significantly diverge, despite being steered by different prompts, y^{tgt} and y^{src}. This approach is akin to running a generative process with y^{tgt} while remaining near the trajectory made by the posteriors of $\mathbf{x}_0^{\text{src}}$. Consequently, PDS enables the sampling of $\mathbf{x}_0^{\text{tgt}}$ that aligns with y^{tgt}, while also retaining the identity of $\mathbf{x}_0^{\text{src}}$. This is achieved through the distillation of the posteriors of $\mathbf{x}_0^{\text{src}}$ into the target sampling process.

4.1. Comparison with SDS [34] and DDS [10]

In Figure 3, we visually illustrate the difference among the three optimization methods: SDS [34], DDS [10] and PDS. Here, we model a 2D distribution $\mathbf{x}_0 \sim p(\mathbf{x}_0) \in \mathbb{R}^2$ that is separated by two marginals, $p(\mathbf{x}_0|y=1)$ and $p(\mathbf{x}_0|y=2)$ which are colored by red and blue, respectively. Then, we train a diffusion model conditioned on the class labels y. Using the pre-trained conditional diffusion model, we aim to transition $\mathbf{x}_0^{\text{tgt}}$ starting from $\mathbf{x}_0^{\text{src}} \sim p(\mathbf{x}_0|y=1)$ towards the other marginal $p(\mathbf{x}_0|y=2)$. The trajectories of three optimization methods are plotted in Figure 3 with their endpoints denoted by stars. As illustrated, SDS and DDS significantly displace the data from the initial position, whereas our method is terminated near the boundary of the two marginals. This is the optimal endpoint for an editing purpose as it indicates proximity to both the starting points and $p(\mathbf{x}_0|y=2)$, thereby achieving a balance between the necessary change and the original identity.

4.2. Comparison with Iterative DU

When a parameterization of images is given as NeRF [28], recent works [9, 29] have shown promising NeRF editing results based on a method known as Iterative Dataset Update (Iterative DU). This method bypasses 3D editing by performing the editing process within 2D space. Given an

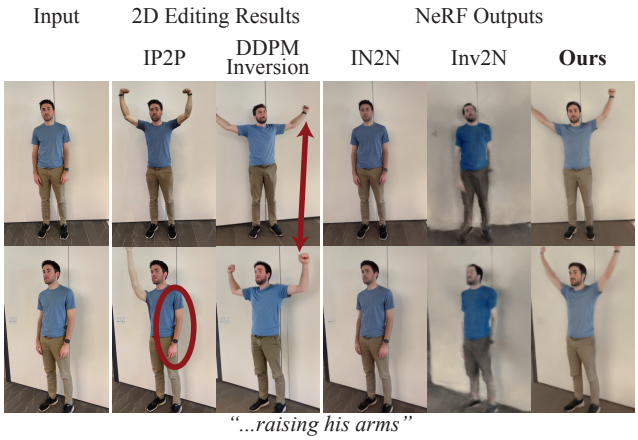


Figure 4. **An example of editing inducing large variations across different views.** The figure shows NeRF editing results of ours and Iterative DU methods, IN2N [9] and Inv2N, with their corresponding 2D editing results obtained by IP2P [2] and DDPM Inversion [15], respectively. When 2D editing leads to large variations, the Iterative DU methods fail to produce accurate edits in 3D space.

image dataset $\{I_v^{\text{src}}\}_{v=1}^N$ used for NeRF [28] reconstruction with viewpoints v, they randomly replace I_v^{src} with its 2D edited version using Instruct-Pix2Pix (IP2P) [2]. By iteratively updating the input images, they progressively transform the input NeRF scene into an edited version of it.

In contrast to Iterative DU which performs editing in 2D space, our approach directly edits NeRFs [28] in 3D space. To visually demonstrate this difference, Figure 4 presents a qualitative comparison of ours and various methods based on Iterative DU. Specifically, we compare ours with Instruct-NeRF2NeRF (IN2N) [9] which uses IP2P [2] for 2D editing. Additionally, we include another Iterative-DU-based method, Inversion2NeRF (Inv2N), which employs DDPM inversion [15] for its 2D editing process. Given the prompt *"raising his arms"*, the figure illustrates significant variations in 2D edited images across different views: the man raises either only one arm or both arms, as marked by the red circle. Furthermore, the red arrow highlights the inconsistency in the poses of raising arms across different views. Such notable discrepancies in 2D editing hinder the Iterative DU methods from transferring these edits into 3D space. Particularly noteworthy is the comparison of our method with Inv2N, both of which leverage the stochastic latent for editing. However, while Inv2N confines its editing within 2D space, ours directly updates NeRF parameters in 3D space by reformulating the 2D image editing method [15] into an optimization form. Consequently, as shown in Figure 4 and Figure 2, ours is the only one to facilitate complex geometric changes and the addition of objects in 3D scenes. It demonstrates the strength of our method lies in the novel optimization design, which allows

for direct 3D editing, not just relying on the editing capabilities of DDPM inversion [15].

5. NeRF Editing with PDS

As one of the applications of PDS, we present a detailed pipeline for NeRF [28] editing. NeRF can be seen as a parameterized rendering function. The rendering process is expressed as $I_v = g(v;\theta)$, where the function takes a specific viewpoint v to render the image I_v at that viewpoint with the rendering parameter θ. Using the publicly available Stable Diffusion [39] as our diffusion prior model, we encode the current rendering at viewpoint v to obtain the target latent $\mathbf{x}^{\text{tgt}}_{0,v}$: $\mathbf{x}^{\text{tgt}}_{0,v} := \mathcal{E}(g(v;\theta))$, where $\mathcal{E}$ is a pre-trained encoder. Similarly, given the original source images $\{I^{\text{src}}_v\}$ used for NeRF [28] reconstruction, the source latent $\mathbf{x}^{\text{src}}_{0,v}$ is also computed by encoding the source image at viewpoint v: $\mathbf{x}^{\text{src}}_{0,v} := \mathcal{E}(I^{\text{src}}_v)$.

For real scenes, there are no given source prompts. Thus, we manually create descriptions for the real scenes, such as *"a photo of a man"* in Figure 1. For target prompts y^{tgt}, we adjust y^{src} by appending a description of a desired attribute—e.g.,*"...raising his arms"* in Figure 4—or by substituting an existing word in y^{src} with a new one, such as changing *"deer doll"* to *"unicorn doll"* in the last row of Figure 2. Given a pre-fixed set of viewpoints $\{v\}$, we randomly select a viewpoint v to compute $\mathbf{x}^{\text{src}}_{0,v}$ and $\mathbf{x}^{\text{tgt}}_{0,v}$. The pairs of $(\mathbf{x}^{\text{src}}_{0,v}, y^{\text{src}})$ and $(\mathbf{x}^{\text{tgt}}_{0,v}, y^{\text{tgt}})$ are fed into the PDS optimization to update θ in a direction dictated by the target prompt. After the optimization, the updated NeRF parameter $\tilde{\theta}$ renders an edited 3D scene that is aligned with the target prompt: $\tilde{I}_v := g(v;\tilde{\theta})$.

To further improve the final output, we take a refinement stage inspired by DreamBooth3D [36]. During iterations of the refinement stage, we randomly select an edited rendering $\tilde{I}_v$ and refine it into a more realistic-looking image using SDEdit [24]. The edited NeRF scenes through PDS optimization are then further refined by a reconstruction loss with these repeatedly updated images.

In some cases of source prompts we create, we observe some gap between the ideal text prompt, which would ideally reconstruct the input image through the generative process, and the actual prompt we provide. To alleviate this discrepancy issue, we have found it effective to finetune the Stable Diffusion [39] with $\{I^{\text{src}}_v\}$ and y^{src} following the DreamBooth [40] setup.

6. Experiment Results

In this section, we conduct editing experiments across two types of parameterized images. Section 6.1 presents NeRF editing results, comparing our NeRF editing capabilities to the state-of-the-art NeRF editing methods. Furthermore, Section 6.2 shows SVG editing results to compare PDS against other optimization methods, namely SDS [34] and DDS [10].

6.1. NeRF Editing

Datasets. We use real scenes we capture as well as the scenes from IN2N [9] and LLFF [27]. The total number of scenes is 13, and the final number of pairs of source scenes and target text prompts is 37 with multiple target prompts for each scene.

Baselines. For extensive comparisons, we evaluate our method against three baselines: Instruct-NeRF2NeRF (IN2N) [9], DDS [10] and Inversion2NeRF (Inv2N). First, we compare ours with IN2N [9], which is a state-of-the-art NeRF editing method with its code publicly available. Additionally, as introduced in Section 4.2, we conduct a comparison with Inv2N, another method based on Iterative DU, which performs editing within 2D space rather directly in 3D space, but employs DDPM inversion [15] instead of IP2P [2] for 2D editing.

Results. Figure 2 presents the qualitative comparisons of NeRF editing. Notably, as depicted in rows 1 and 2, our method is the only one that makes large geometric changes in 3D scenes from the input text, folding the man's arms to create natural poses of him reading a book or drinking coffee. In contrast, Iterative-DU-based methods like IN2N [9] and Inv2N fail to produce the right edits in 3D space. DDS [10] produces the outputs that completely lose the identity of the input scenes, focusing solely on conforming to the input texts. Rows 3 and 4 of Figure 2 show the editing scenarios of adding objects in outdoor scenes without specifying local regions, which also leads to large variations. Here, our method successfully adds objects like windmills and hot air balloons in the input scenes, maintaining their background details. On the other hand, the baselines either fail to add the objects in 3D space or produce outputs that significantly deviate from the original scenes. When it comes to appearance change, which induces relatively little variations across different views, both our method and IN2N [9] effectively produce the desired appearance change in 3D scenes, as shown in the last row of Figure 2. However, ours most preserves the original identity of the input scene, such as the object's color, while making appropriate changes. Additional qualitative results are presented through videos on our project page[1].

To further assess the perceptual quality of the editing results, we conduct a user study compared to the baselines. Following Ritchie [38], participants were shown input NeRF scene videos, editing prompts, and edited NeRF scene videos produced by ours and the baselines. They were

[1] https://posterior-distillation-sampling.github.io

Table 1. **A quantitative comparison of NeRF editing between ours and other baselines.** Ours outperforms the baselines quantitatively. **Bold** indicates the best result for each column.

Methods	CLIP [35] Score ↑	User Preference Rate (%) ↑
IN2N [9]	0.2280	27.71
DDS [10]	0.2210	13.71
Inv2N	0.2232	9.24
PDS (Ours)	**0.2477**	**49.33**

Table 2. **A quantitative comparison of SVG editing between SDS [34], DDS [10] and PDS.** Ours outperforms the others in LPIPS [56] while achieving a CLIP [35] score that is on par with the others. **Bold** indicates the best result for each column.

Methods	CLIP [35] Score ↑	LPIPS [56] ↓	User Preference Rate (%) ↑
SDS [34]	**0.2606**	0.4855	30.83
DDS [10]	0.2460	0.5982	20.24
PDS (Ours)	0.2504	**0.3121**	**48.94**

then asked to choose the most appropriate edited NeRF scene video. As illustrated in Table 1, our editing results are most preferred over the baselines in human evaluation by a large margin: 49.33% (Ours) vs. 27.71% (IN2N [9], the second best). See the **supplementary material** for a more detailed user study setup.

For a quantitative evaluation, we measure CLIP [35] Score that measures the similarity between edited 2D renderings and target text prompts in CLIP [35] space. As shown in Table 1, ours outperforms the baselines quantitatively. This is corroborated by the qualitative results illustrated in Figure 2, especially in scenarios of geometric changes or object addition, where the other baselines have difficulty in making the right edits.

6.2. SVG Editing

Experimental Setup. We use pairs of SVGs and their corresponding text prompts used in VectorFusion [17] as input. By manually creating target text prompts, we conduct experiments with a total of 48 pairs of input SVGs and target text prompts. For comparison, we evaluate our method against other optimization methods, SDS [34] and DDS [10]. To perform editing with SDS, we start with a source SVG as an initial updated SVG and then update it using a target prompt according to the SDS [34] optimization. Following DDS, we use CLIP [35] score and LPIPS [56] as quantitative metrics.

Results. Qualitative results of SVG editing are shown in Figure 5. It demonstrates that while all the methods effectively change input SVGs according to the target text prompts, ours best preserves the structural semantics of the input SVGs. This is particularly evident in row 3 of Figure 5, where ours maintains the overall color pattern of the input SVG.

Figure 5. **A qualitative comparison of SVG editing using three different optimization methods: SDS [34], DDS [10] and PDS.** PDS makes changes according to input text while most preserving the structural semantics of the input SVGs.

The trends from the qualitative results are mirrored in our quantitative results. As seen in Table 2, ours significantly surpasses the others in LPIPS [56] by a large margin, which measures the fidelity to the input SVG, while our CLIP score is on par with the others. This demonstrates that our method introduces only minimal necessary changes to meet the described attributes in the target text prompts.

We further provide a user study result of SVG editing in Table 2. We use the same user study setup used in NeRF editing (Section 6.1). Consistent with the qualitative and quantitative results, ours are most preferred in human evaluation.

7. Conclusion

We propose Posterior Distillation Sampling (PDS), an optimization method for parametric image editing. PDS matches the stochastic latents of the source and the target to fulfill both conformity to the target text and preservation of the source identity in parameter space. We demonstrate the versatility of PDS in parametric image editing through a comparative analysis between ours and other optimization methods and extensive experiments across various parameter spaces.

References

[1] Anonymous. Learning pseudo 3D guidance for view-consistent 3D texturing with 2D diffusion. In *Submitted to The Twelfth International Conference on Learning Representations*, 2023. under review. 3

[2] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *CVPR*, 2023. 4, 6, 7

[3] Minwoo Byeon, Beomhee Park, Haecheon Kim, Sungjun Lee, Woonhyuk Baek, and Saehoon Kim. Coyo-700m: Image-text pair dataset. https://github.com/kakaobrain/coyo-dataset, 2022. 3

[4] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3D: Disentangling geometry and appearance for high-quality text-to-3D content creation. In *ICCV*, 2023. 3

[5] DeepFloyd. Deepfloyd if. https://www.deepfloyd.ai/deepfloyd-if/. 3

[6] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 2021. 4

[7] Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Text-to-audio generation using instruction tuned llm and latent diffusion model. *arXiv preprint arXiv:2304.13731*, 2023. 3

[8] Ligong Han, Song Wen, Qi Chen, Zhixing Zhang, Kunpeng Song, Mengwei Ren, Ruijiang Gao, Yuxiao Chen, Di Liu, Qilong Zhangli, et al. Improving negative-prompt inversion via proximal guidance. *arXiv preprint arXiv:2306.05414*, 2023. 4

[9] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D scenes with instructions. In *ICCV*, 2023. 3, 4, 6, 7, 8

[10] Amir Hertz, Kfir Aberman, and Daniel Cohen-Or. Delta denoising score. In *ICCV*, 2023. 3, 4, 5, 6, 7, 8, 12

[11] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-or. Prompt-to-prompt image editing with cross-attention control. In *ICLR*, 2023. 3, 4

[12] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 4, 12

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 3, 4, 5

[14] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. *arXiv preprint arXiv:2301.12661*, 2023. 3

[15] Inbar Huberman-Spiegelglas, Vladimir Kulikov, and Tomer Michaeli. An edit friendly DDPM noise space: Inversion and manipulations. *arXiv preprint arXiv:2304.06140*, 2023. 3, 4, 5, 6, 7

[16] Shir Iluz, Yael Vinker, Amir Hertz, Daniel Berio, Daniel Cohen-Or, and Ariel Shamir. Word-as-image for semantic typography. *ACM TOG*, 2023. 3

[17] Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *CVPR*, 2023. 3, 8, 12

[18] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 3

[19] Juil Koo, Seungwoo Yoo, Minh Hieu Nguyen, and Minhyuk Sung. SALAD: Part-level latent diffusion for 3d shape generation and manipulation. In *ICCV*, 2023. 3

[20] Yuseung Lee, Kunho Kim, Hyunjin Kim, and Minhyuk Sung. Syncdiffusion: Coherent montage via synchronized joint diffusions. In *NeurIPS*, 2023. 3

[21] Muheng Li, Yueqi Duan, Jie Zhou, and Jiwen Lu. Diffusion-sdf: Text-to-shape via voxelized diffusion. In *CVPR*, 2023. 3

[22] Yuhan Li, Yishun Dou, Yue Shi, Yu Lei, Xuanhong Chen, Yi Zhang, Peng Zhou, and Bingbing Ni. Focaldreamer: Text-driven 3D editing via focal-fusion assembly. *arXiv preprint arXiv:2308.10608*, 2023. 4

[23] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-resolution text-to-3D content creation. In *CVPR*, 2023. 3

[24] 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. 7, 12

[25] 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. 3

[26] Midjourney. Midjourney. https://www.midjourney.com/. 3

[27] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM TOG*, 2019. 7

[28] 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. 3, 4, 6, 7

[29] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Marcus A Brubaker, Jonathan Kelly, Alex Levinshtein, Konstantinos G Derpanis, and Igor Gilitschenski. Watch your steps: Local image and scene editing by text instructions. *arXiv preprint arXiv:2308.08947*, 2023. 4, 6

[30] Daiki Miyake, Akihiro Iohara, Yu Saito, and Toshiyuki Tanaka. Negative-prompt inversion: Fast image inversion for editing with text-guided diffusion models. *arXiv preprint arXiv:2305.16807*, 2023. 4

[31] Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *CVPR*, 2023. 4

[32] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 3

[33] Jangho Park, Gihyun Kwon, and Jong Chul Ye. ED-NeRF: Efficient text-guided editing of 3D scene using latent space NeRF. *arXiv preprint arXiv:2310.02712*, 2023. 4

[34] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3D using 2D diffusion. In *ICLR*, 2023. 3, 4, 5, 6, 7, 8, 12

[35] 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*, 2021. 8

[36] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. Dreambooth3D: Subject-driven text-to-3D generation. In *ICCV*, 2023. 3, 7

[37] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 3

[38] Daniel Ritchie. Rudimentary framework for running two-alternative forced choice (2afc) perceptual studies on mechanical turk. 7

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 3, 7

[40] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, 2023. 7

[41] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS*, 2022. 3

[42] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021. 3

[43] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *NeurIPS*, 2022. 3

[44] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. MVDream: Multi-view diffusion for 3D generation. *arXiv preprint arXiv:2308.16512*, 2023. 3

[45] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 3

[46] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 4

[47] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *NeurIPS*, 2019.

[48] 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. 3

[49] Bram Wallace, Akash Gokul, and Nikhil Naik. EDICT: Exact diffusion inversion via coupled transformations. In *CVPR*, 2023. 3, 4

[50] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2D diffusion models for 3D generation. In *CVPR*, 2023. 3, 5

[51] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. In *NeurIPS*, 2023. 3

[52] Chen Henry Wu and Fernando De la Torre. A latent space of stochastic diffusion models for zero-shot image editing and guidance. In *ICCV*, 2023. 3, 4, 5

[53] Ximing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. Diffsketcher: Text guided vector sketch synthesis through latent diffusion models. In *NeurIPS*, 2023. 3

[54] Xudong Xu, Zhaoyang Lyu, Xingang Pan, and Bo Dai. Matlaber: Material-aware text-to-3D via latent BRDF auto-encoder. *arXiv preprint arXiv:2308.09278*, 2023. 3

[55] Dongchao Yang, Jianwei Yu, Helin Wang, Wen Wang, Chao Weng, Yuexian Zou, and Dong Yu. Diffsound: Discrete diffusion model for text-to-sound generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023. 3

[56] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 8

[57] Joseph Zhu and Peiye Zhuang. HiFA: High-fidelity text-to-3D with advanced diffusion guidance. *arXiv preprint arXiv:2305.18766*, 2023. 3, 4

[58] Jingyu Zhuang, Chen Wang, Lingjie Liu, Liang Lin, and Guanbin Li. Dreameditor: Text-driven 3D scene editing with neural fields. *arXiv preprint arXiv:2306.13455*, 2023. 4

Appendix

A.1. Derivation of Posterior Distillation Sampling

For a comprehensive derivation of Equation 14, we first remind that the objective function of PDS is expressed as:

$$\mathcal{L}_{\tilde{\mathbf{z}}_t}(\mathbf{x}_0^{\text{tgt}}) = \mathbb{E}\left[\|\tilde{\mathbf{z}}_t^{\text{tgt}} - \tilde{\mathbf{z}}_t^{\text{src}}\|_2^2\right] \tag{15}$$

$$= \mathbb{E}\left[\left\|\frac{\mathbf{x}_{t-1}^{\text{tgt}} - \boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{tgt}}, y^{\text{tgt}}; \boldsymbol{\epsilon}_\phi)}{\sigma_t} - \frac{\mathbf{x}_{t-1}^{\text{src}} - \boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{src}}, y^{\text{src}}; \boldsymbol{\epsilon}_\phi)}{\sigma_t}\right\|_2^2\right] \tag{16}$$

$$= \mathbb{E}\left[\frac{1}{\sigma_t^2}\left\|(\mathbf{x}_{t-1}^{\text{tgt}} - \mathbf{x}_{t-1}^{\text{src}}) - \left(\boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{tgt}}, y^{\text{tgt}}; \boldsymbol{\epsilon}_\phi) - \boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{src}}, y^{\text{src}}; \boldsymbol{\epsilon}_\phi)\right)\right\|_2^2\right]. \tag{17}$$

Given that $\tilde{\mathbf{z}}_t^{\text{src}}$ and $\tilde{\mathbf{z}}_t^{\text{tgt}}$ share the same noises $\boldsymbol{\epsilon}_{t-1}$ and $\boldsymbol{\epsilon}_t$ for their respective $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, the difference between $\mathbf{x}_{t-1}^{\text{tgt}}$ and $\mathbf{x}_{t-1}^{\text{src}}$ results in a constant multiple of the difference between $\mathbf{x}_0^{\text{tgt}}$ and $\mathbf{x}_0^{\text{src}}$:

$$\mathbf{x}_{t-1}^{\text{tgt}} - \mathbf{x}_{t-1}^{\text{src}} = \sqrt{\bar{\alpha}_{t-1}}(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}). \tag{18}$$

Following our notation $\hat{\boldsymbol{\epsilon}}_t^{\text{src}} := \boldsymbol{\epsilon}_\phi(\mathbf{x}_t^{\text{src}}, y^{\text{src}}, t)$ and $\hat{\boldsymbol{\epsilon}}_t^{\text{tgt}} := \boldsymbol{\epsilon}_\phi(\mathbf{x}_t^{\text{tgt}}, y^{\text{tgt}}, t)$ introduced in Section 4, the difference between the approximated posterior means is also expressed as follows:

$$\boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{tgt}}, y^{\text{tgt}}; \boldsymbol{\epsilon}_\phi) - \boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{src}}, y^{\text{src}}, \boldsymbol{\epsilon}_\phi) = (\gamma_t + \delta_t\sqrt{\bar{\alpha}_t})(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}) - \gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}(\hat{\boldsymbol{\epsilon}}_t^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_t^{\text{src}}), \tag{19}$$

where $\boldsymbol{\mu}_\phi(\mathbf{x}_t, y; \boldsymbol{\epsilon}_\phi)$ can be expanded as shown in the following equation:

$$\boldsymbol{\mu}_\phi(\mathbf{x}_t, y; \boldsymbol{\epsilon}_\phi) = \gamma_t\tilde{\mathbf{x}}_0(\mathbf{x}_t, y; \boldsymbol{\epsilon}_\phi) + \delta_t\mathbf{x}_t \tag{20}$$

$$= \gamma_t\left(\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, y, t)\right) + \delta_t\mathbf{x}_t \tag{21}$$

$$= (\frac{\gamma_t}{\sqrt{\bar{\alpha}_t}} + \delta_t)\mathbf{x}_t - \gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, y, t) \tag{22}$$

$$= (\gamma_t + \delta_t\sqrt{\bar{\alpha}_t})\mathbf{x}_0 + \sqrt{\frac{1}{\bar{\alpha}_t} - 1}(\gamma_t + \delta_t\sqrt{\bar{\alpha}_t})\boldsymbol{\epsilon}_t - \gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, y, t). \tag{23}$$

Incorporating Equation 18 and Equation 19 into Equation 17, we can reformulate the objective function of PDS as follows:

$$\mathcal{L}_{\tilde{\mathbf{z}}_t}(\mathbf{x}_0^{\text{tgt}}) = \mathbb{E}\left[\frac{1}{\sigma_t^2}\left\|(\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}) + \gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}(\hat{\boldsymbol{\epsilon}}_t^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_t^{\text{src}})\right\|_2^2\right] \tag{24}$$

$$= \mathbb{E}\Bigg[\frac{1}{\sigma_t^2}\Bigg((\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})^2(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}})^2 \tag{25}$$

$$+ 2(\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})\gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}})(\hat{\boldsymbol{\epsilon}}_t^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_t^{\text{src}})$$

$$+ \gamma_t^2(\frac{1}{\bar{\alpha}_t} - 1)(\hat{\boldsymbol{\epsilon}}_t^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_t^{\text{src}})^2\Bigg)\Bigg].$$

By taking the gradient of $\mathcal{L}_{\tilde{\mathbf{z}}_t}$ with respect to θ while ignoring the U-Net jacobian term, $\frac{\partial\boldsymbol{\epsilon}_\phi^{\text{tgt}}}{\partial\mathbf{x}_0^{\text{tgt}}} = 0$, one can obtain PDS as follows:

$$\nabla_\theta\mathcal{L}_{\text{PDS}} = \frac{\partial\mathcal{L}_{\tilde{\mathbf{z}}_t}(\mathbf{x}_0^{\text{tgt}})}{\partial\mathbf{x}_0^{\text{tgt}}} \cdot \frac{\partial\mathbf{x}_0^{\text{tgt}}}{\partial\theta} \tag{26}$$

$$= \mathbb{E}\left[\frac{2}{\sigma_t^2}\left((\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})^2(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}) + (\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})\gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}(\hat{\boldsymbol{\epsilon}}_t^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_t^{\text{src}})\right)\frac{\partial\mathbf{x}_0^{\text{tgt}}}{\partial\theta}\right]. \tag{27}$$

Thus, the coefficients $\psi(t)$ and $\chi(t)$ in Equation 14 are as follows:

$$\psi(t) = \frac{2(\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})^2}{\sigma_t^2} \tag{28}$$

$$\chi(t) = \frac{2(\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})}{\sigma_t^2}\gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1} \tag{29}$$

In practice, we sample non-consecutive timesteps for $t-1$ and t since the coefficients become 0 when they are consecutive. Given a sequence of non-consecutive timesteps $[\tau_i]_{i=1}^S$, a more generalized form of PDS is represented as follows:

$$\nabla_\theta \mathcal{L}_{\text{PDS}} = \mathbb{E}_{i,\boldsymbol{\epsilon}_{\tau_i},\boldsymbol{\epsilon}_{\tau_{i-1}}}\left[\psi(i)(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}) + \chi(i)(\hat{\boldsymbol{\epsilon}}_{\tau_i}^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_{\tau_i}^{\text{src}})\frac{\partial \mathbf{x}_0^{\text{tgt}}}{\partial\theta}\right], \tag{30}$$

where

$$\psi(i) = \frac{2(\sqrt{\bar{\alpha}_{\tau_{i-1}}} - \gamma_{\tau_i} - \delta_{\tau_i}\sqrt{\bar{\alpha}_{\tau_i}})^2}{\sigma_{\tau_i}^2}, \tag{31}$$

$$\chi(i) = \frac{2(\sqrt{\bar{\alpha}_{\tau_{i-1}}} - \gamma_{\tau_i} - \delta_{\tau_i}\sqrt{\bar{\alpha}_{\tau_i}})}{\sigma_{\tau_i}^2}\gamma_{\tau_i}\sqrt{\frac{1}{\bar{\alpha}_{\tau_i}} - 1}. \tag{32}$$

For more details on timestep sampling, refer to the implementation details in the next section.

A.2. Implementation Details

In this section, we provide the implementation details of NeRF and SVG editing presented in Section 6.1 and Section 6.2, respectively.

NeRF Editing. We run the PDS optimization for 30,000 iterations with classifier-free guidance [12] weights within $[30, 100]$ depending on the complexity of editing. As detailed in Section A.1, we sample non-consecutive timesteps τ_{i-1} and τ_i since the coefficients $\psi(\cdot)$ and $\chi(\cdot)$ become zero when the sampled timesteps are consecutive. For this, we define non-consecutive timesteps $[\tau_i]_{i=1}^S$, which is a subset sequence of the total forward process timesteps of the diffusion model, $[1, ..., T]$. Specifically, we select these timesteps such that $\tau_i = \lfloor 2i \rfloor$, resulting in a subset sequence length of $S = 500$ out of the total $T = 1000$ timesteps. We then randomly sample the index i within a ratio range of $[0.02, 0.98]$, i.e., $i \sim \mathcal{U}(10, 490)$.

During the refinement stage, we randomly choose and replace $\tilde{I}_v$ every 10 iterations, over total 15,000 iterations. We denote a SDEdit [24] operator by $\mathcal{S}(\mathbf{x}_0; t_0, \boldsymbol{\epsilon}_\phi)$ which samples $\mathbf{x}_{t_0} \sim \mathcal{N}(\sqrt{\bar{\alpha}_{t_0}}\mathbf{x}_0, (1 - \bar{\alpha}_{t_0})\mathbf{I})$ then starts denoising it from t_0 using $\boldsymbol{\epsilon}_\phi$. For the denoising process, we randomly sample t_0 within a ratio range of $[0, 0.2]$ out of total denoising steps $N = 20$.

SVG Editing. Across all optimizations, SDS [34], DDS [10], and our proposed PDS, we apply the same classifier-free guidance weight of 100. For SDS [34], we sample t within a ratio range of $[0.05, 0.95]$ following VectorFusion [17]. For DDS [10], we follow its original setup, sampling t within $[0.02, 0.98]$. For PDS, we sample i out of a ratio range of $[0.1, 0.98]$.

A.3. Details of User Study

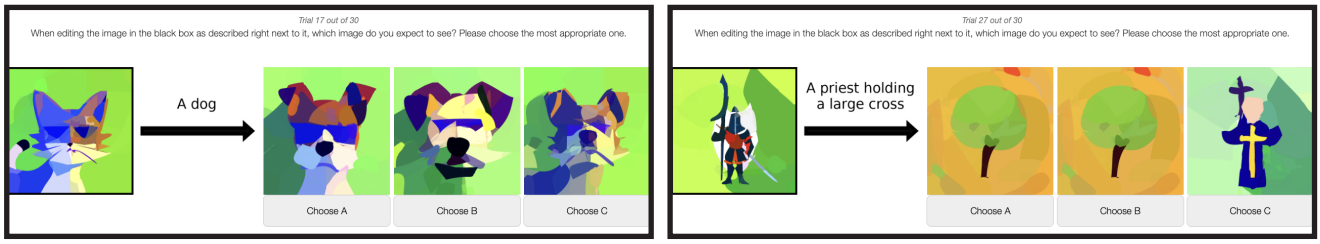
We conduct user studies for the human evaluation of NeRF and SVG editing through Amazon's Mechanical Turk. We collected survey responses only from those participants who passed our vigilance tasks. In the NeRF and SVG editing user studies, we received 42 and 17 valid responses, respectively. Screenshots of our NeRF and SVG editing user studies, including examples of vigilance tasks, are displayed in Figure A6 and Figure A7, respectively.

A.4. Effect of the Refinement Stage

Figure A8 illustrates the effect of our refinement stage. As shown in Figure A8, the desired complex edit — making the man raise his arms — is already produced through PDS optimization. However, the refinement stage further improves the fidelity of the output, leading to a more realistic appearance by effectively removing artifacts.

(a) Main problem (b) Vigilance task

Figure A6. **NeRF editing user study screenshots.** The participants are presented with NeRF scene videos and editing prompts, and are asked to answer the following question: `When editing the video in the black box as described right next to it, which video do you expect to see? Please choose the most appropriate one.`



(a) Main problem (b) Vigilance task

Figure A7. **SVG editing user study screenshots.** Given SVG images and editing prompts, the participants are asked to answer the following question: `When editing the image in the black box as described right next to it, which image do you expect to see? Please choose the most appropriate one.`

Figure A8. **The effect of the refinement stage.** While PDS optimization already produces desired edits in 3D scenes, the refinement stage further improves the fidelity of the outputs by removing artifacts.