Posterior Distillation Sampling — Supplementary Material

Juil Koo Chanho Park Minhyuk Sung
KAIST
{63days,charlieppark,mhsung}@kaist.ac.kr

In this supplementary material, we begin with related work and comparisons of PDS and previous work. Then, we present a detailed description of the framework of NeRF [16] editing with PDS in Section S.3. Following that, we present additional editing results with more diverse representations, including 3D Gaussian Splats (3DGS) [12] and 2D images, in Section S.4. Then, in Section S.5, we provide a more detailed derivation of Posterior Distillation Sampling introduced in Section 3 of the main paper. Following that, Section S.6 presents the implementation details of NeRF editing and SVG editing discussed in Sections 4.1 and 4.2 of the main paper, respectively. Subsequently, Section S.7 provides the details of our user study setups. Lastly, the effect of the refinement stage, discussed in Section S.3 of the main paper, is detailed in Section S.8.

S.1. Related Work

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 [10, 19, 24], texture [1, 15], material [28] and Scalable Vector Graphics (SVGs) [9, 10, 27]. 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. Wang *et al.* [25] have proposed Variational Score Distillation (VSD) to address over-saturation, over-smoothing, and low-diversity problems in SDS [19]. Zhu and Zhuang [29] use more accurate predictions of diffusion models via iterative denoising at every SDS update step. When it comes to editing, Hertz *et al.* [5] 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.

Text-Driven NeRF Editing. Haque *et al.* [4] have proposed a text-driven NeRF editing method, known as Iterative Dataset Update (Iterative DU). It iteratively replaces reference images, initially used for NeRF [16] reconstruction, with edited images using Instruct-Pix2Pix [2]. By applying a reconstruction loss with these iteratively updated images to an input NeRF [16] scene, the scene is gradually transformed to its edited counterpart. Mirzae *et al.* [17] improve Instruct-NeRF2NeRF [4] 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 [13, 18, 30] directly apply SDS [19] or DDS [5] 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.

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

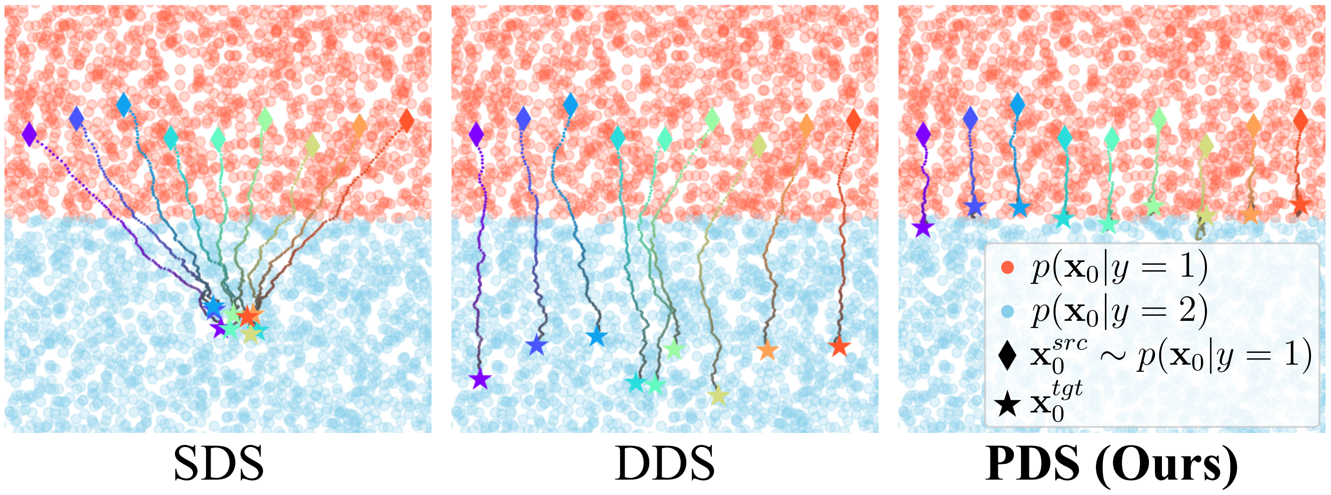


Figure S1. **A visual comparison of the editing process through SDS [19], DDS [5] and PDS.** The figure illustrates the trajectories of samples drawn from $p(\mathbf{x}_0|y=1)$ as they are shifted towards $p(\mathbf{x}_0|y=2)$. PDS notably moves the samples near the boundary of the two marginals—the optimal endpoint in that it balances the necessary change with the original identity.

of Denoising Diffusion Implicit Models (DDIM) [23], one can approximately run the ODE of the generative process in reverse [3, 23], referred to as DDIM inversion. Meanwhile, an alternative approach, known as DDPM inversion [8, 26], employs the stochastic generative process of Denoising Diffusion Probabilistic Models (DDPM) [7]. 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.

S.2. Comparison of PDS and Previous Work

Comparison with SDS [19] and DDS [5]. In Figure S1, we visually illustrate the difference among the three optimization methods: SDS [19], DDS [5] 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 S1 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.

Comparison with Iterative DU. When a parameterization of images is given as NeRF [16], recent works [4, 17] 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 image dataset $\{I_v^{\text{src}}\}_{v=1}^{N}$ used for NeRF [16] 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 [16] in 3D space. To visually demonstrate this difference, Figure S2 presents a qualitative comparison of ours and various methods based on Iterative DU. Specifically, we compare ours with Instruct-NeRF2NeRF (IN2N) [4] which uses IP2P [2] for 2D editing. Additionally, we include another Iterative-DU-based method, Inversion2NeRF (Inv2N), which employs DDPM inversion [8] 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 [8] into

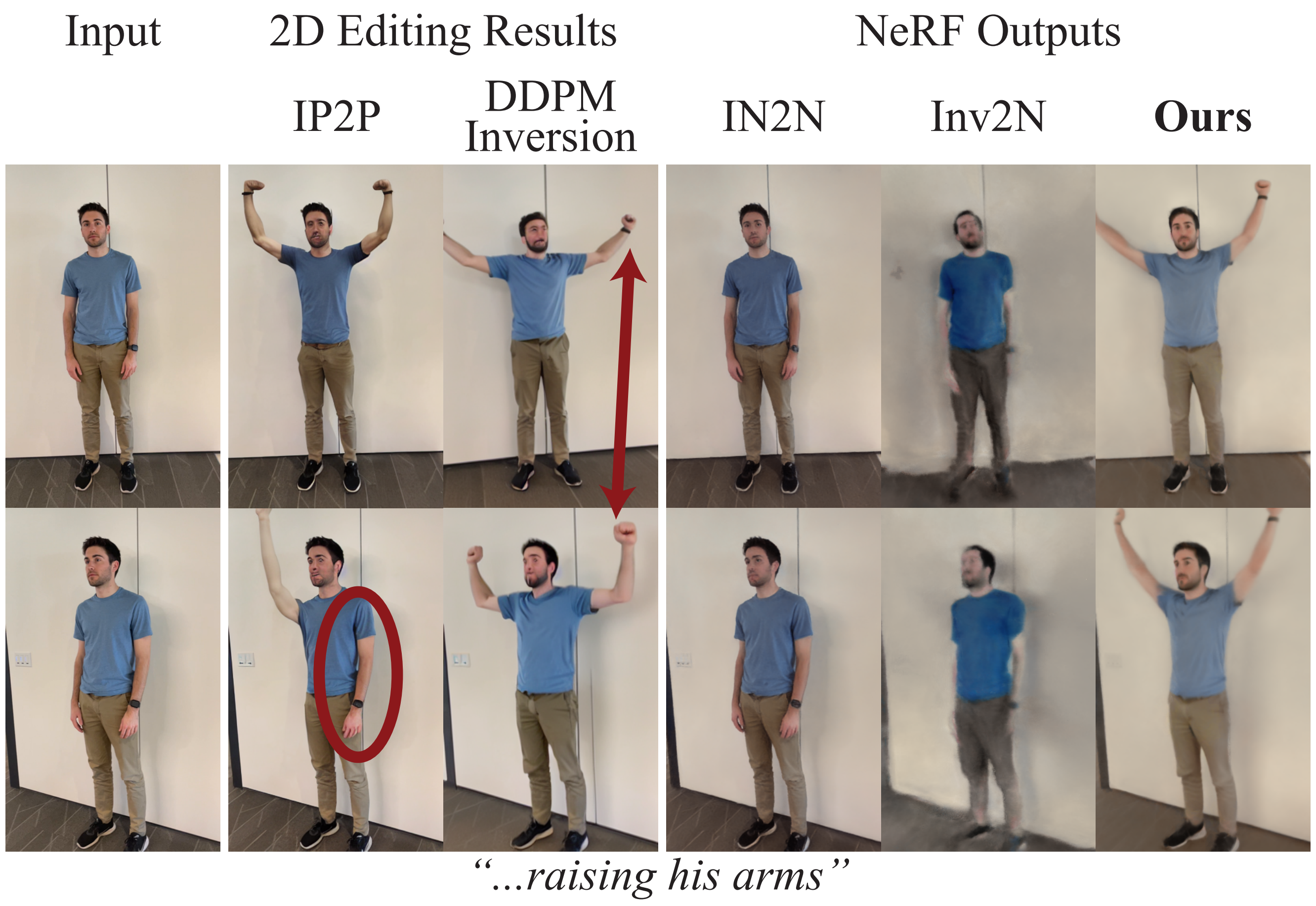


Figure S2. **An example of editing inducing large variations across different views.** The figure shows NeRF editing results of ours and Iterative DU methods, IN2N [4] and Inv2N, with their corresponding 2D editing results obtained by IP2P [2] and DDPM Inversion [8], respectively. When 2D editing leads to large variations, the Iterative DU methods fail to produce accurate edits in 3D space.

an optimization form. Consequently, as shown in Figure S2 and Figure 1, 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 [8].

S.3. NeRF Editing with PDS

As one of the applications of PDS, we present a detailed pipeline for NeRF [16] 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 [21] as our diffusion prior model, we encode the current rendering at viewpoint v to obtain the target latent $\mathbf{x}_{0,v}^{\text{tgt}}$: $\mathbf{x}_{0,v}^{\text{tgt}} := \mathcal{E}(g(v;\theta))$, where $\mathcal{E}$ is a pre-trained encoder. Similarly, given the original source images $\{I_v^{\text{src}}\}$ used for NeRF [16] reconstruction, the source latent $\mathbf{x}_{0,v}^{\text{src}}$ is also computed by encoding the source image at viewpoint v: $\mathbf{x}_{0,v}^{\text{src}} := \mathcal{E}(I_v^{\text{src}})$.

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.,*"...reading a book"* in Figure 1 of the main paper—or by substituting an existing word in y^{src} with a new one. Given a pre-fixed set of viewpoints $\{v\}$, we randomly select a viewpoint v to compute $\mathbf{x}_{0,v}^{\text{src}}$ and $\mathbf{x}_{0,v}^{\text{tgt}}$. The pairs of $(\mathbf{x}_{0,v}^{\text{src}}, y^{\text{src}})$ and $(\mathbf{x}_{0,v}^{\text{tgt}}, y^{\text{tgt}})$ 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 [20]. 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 [14]. 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 [21] with $\{I_v^{\text{src}}\}$ and y^{src} following the DreamBooth [22] setup.

S.4. Editing 3D Gaussian Splats [12] and 2D Images

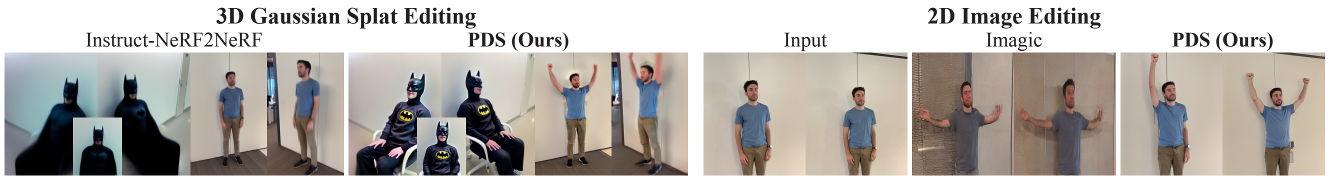


Figure S3. **Editing of more diverse representations, 3D Gaussian Splats [12] and 2D images.** PDS consistently outperforms the baselines. The target attributes are *"Batman"* and *"raising the arms."*

PDS encompasses various editing scenarios, not confined within a specific parameter space. To further assess the versatility and generalizability of PDS in editing tasks, we include both 3D Gaussian Splat (3DGS) [12] editing and 2D image editing. As NeRF editing, Figure S3 shows that PDS outperforms Instruct-NeRF2NeRF [4] in 3DGS representation while uniquely realizing geometric changes. In 2D image editing, PDS demonstrates superior performance compared to Imagic [11], which is introduced for 2D image editing using pre-trained 2D diffusion models. PDS edits the input image while preserving other details with high fidelity. On the other hand, Imagic [11] leaves artifacts, losing the identity of the source content.

S.5. Derivation of Posterior Distillation Sampling

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

$$\begin{aligned}\mathcal{L}_{\tilde{\mathbf{z}}_t}(\mathbf{x}_0^{\text{tgt}}) &= \mathbb{E}\left[\|\tilde{\mathbf{z}}_t^{\text{tgt}} - \tilde{\mathbf{z}}_t^{\text{src}}\|_2^2\right] && (1)\\ &= \mathbb{E}\left[\left\|\frac{\mathbf{x}_{t-1}^{\text{tgt}} - \boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{tgt}}, y^{\text{tgt}}; \boldsymbol{\epsilon}_\phi)}{\sigma_t} - \frac{\mathbf{x}_{t-1}^{\text{src}} - \boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{src}}, y^{\text{src}}; \boldsymbol{\epsilon}_\phi)}{\sigma_t}\right\|_2^2\right] && (2)\\ &= \mathbb{E}\left[\frac{1}{\sigma_t^2}\left\|(\mathbf{x}_{t-1}^{\text{tgt}} - \mathbf{x}_{t-1}^{\text{src}}) - \left(\boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{tgt}}, y^{\text{tgt}}; \boldsymbol{\epsilon}_\phi) - \boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{src}}, y^{\text{src}}; \boldsymbol{\epsilon}_\phi)\right)\right\|_2^2\right]. && (3)\end{aligned}$$

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 $\mathbf{x}_0^{\text{src}}$:

$$\mathbf{x}_{t-1}^{\text{tgt}} - \mathbf{x}_{t-1}^{\text{src}} = \sqrt{\bar{\alpha}_{t-1}}(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}). \tag{4}$$

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 3 of the main paper, the difference between the approximated posterior means is also expressed as follows:

$$\boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{tgt}}, y^{\text{tgt}}; \boldsymbol{\epsilon}_\phi) - \boldsymbol{\mu}_\phi(\mathbf{x}_t^{\text{src}}, y^{\text{src}}, \boldsymbol{\epsilon}_\phi) = (\gamma_t + \delta_t\sqrt{\bar{\alpha}_t})(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}) - \gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}(\hat{\boldsymbol{\epsilon}}_t^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_t^{\text{src}}), \tag{5}$$

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

$$\begin{aligned}\boldsymbol{\mu}_\phi(\mathbf{x}_t, y; \boldsymbol{\epsilon}_\phi) &= \gamma_t\tilde{\mathbf{x}}_0(\mathbf{x}_t, y; \boldsymbol{\epsilon}_\phi) + \delta_t\mathbf{x}_t && (6)\\ &= \gamma_t\left(\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, y, t)\right) + \delta_t\mathbf{x}_t && (7)\\ &= (\frac{\gamma_t}{\sqrt{\bar{\alpha}_t}} + \delta_t)\mathbf{x}_t - \gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, y, t) && (8)\\ &= (\gamma_t + \delta_t\sqrt{\bar{\alpha}_t})\mathbf{x}_0 + \sqrt{\frac{1}{\bar{\alpha}_t} - 1}(\gamma_t + \delta_t\sqrt{\bar{\alpha}_t})\boldsymbol{\epsilon}_t - \gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}\boldsymbol{\epsilon}_\phi(\mathbf{x}_t, y, t). && (9)\end{aligned}$$

Incorporating Equation 4 and Equation 5 into Equation 3, we can reformulate the objective function of PDS as follows:

$$\mathcal{L}_{\tilde{\mathbf{z}}_t}(\mathbf{x}_0^{\text{tgt}}) = \mathbb{E}\left[\frac{1}{\sigma_t^2}\left\|(\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}) + \gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}(\boldsymbol{\epsilon}_t^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_t^{\text{src}})\right\|_2^2\right] \tag{10}$$

$$= \mathbb{E}\left[\frac{1}{\sigma_t^2}\Bigg((\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})^2(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}})^2 \right. \tag{11}$$

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

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

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

$$\nabla_\theta \mathcal{L}_{\text{PDS}} = \frac{\partial \mathcal{L}_{\tilde{\mathbf{z}}_t}(\mathbf{x}_0^{\text{tgt}})}{\partial \mathbf{x}_0^{\text{tgt}}} \cdot \frac{\partial \mathbf{x}_0^{\text{tgt}}}{\partial \theta} \tag{12}$$

$$= \mathbb{E}\left[\frac{2}{\sigma_t^2}\left((\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})^2(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}) + (\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})\gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}(\hat{\boldsymbol{\epsilon}}_t^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_t^{\text{src}})\right)\frac{\partial \mathbf{x}_0^{\text{tgt}}}{\partial \theta}\right]. \tag{13}$$

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

$$\psi(t) = \frac{2(\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})^2}{\sigma_t^2}, \tag{14}$$

$$\chi(t) = \frac{2(\sqrt{\bar{\alpha}_{t-1}} - \gamma_t - \delta_t\sqrt{\bar{\alpha}_t})}{\sigma_t^2}\gamma_t\sqrt{\frac{1}{\bar{\alpha}_t} - 1}. \tag{15}$$

In practice, we sample non-consecutive timesteps for $t-1$ and t as in DDIM [23] 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:

$$\nabla_\theta \mathcal{L}_{\text{PDS}} = \mathbb{E}_{i,\boldsymbol{\epsilon}_{\tau_i},\boldsymbol{\epsilon}_{\tau_{i-1}}}\left[\psi(i)(\mathbf{x}_0^{\text{tgt}} - \mathbf{x}_0^{\text{src}}) + \chi(i)(\hat{\boldsymbol{\epsilon}}_{\tau_i}^{\text{tgt}} - \hat{\boldsymbol{\epsilon}}_{\tau_i}^{\text{src}})\frac{\partial \mathbf{x}_0^{\text{tgt}}}{\partial \theta}\right], \tag{16}$$

where

$$\psi(i) = \frac{2(\sqrt{\bar{\alpha}_{\tau_{i-1}}} - \gamma_{\tau_i} - \delta_{\tau_i}\sqrt{\bar{\alpha}_{\tau_i}})^2}{\sigma_{\tau_i}^2}, \tag{17}$$

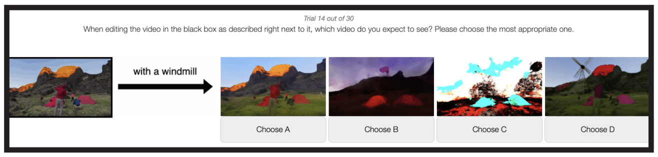
$$\chi(i) = \frac{2(\sqrt{\bar{\alpha}_{\tau_{i-1}}} - \gamma_{\tau_i} - \delta_{\tau_i}\sqrt{\bar{\alpha}_{\tau_i}})}{\sigma_{\tau_i}^2}\gamma_{\tau_i}\sqrt{\frac{1}{\bar{\alpha}_{\tau_i}} - 1}. \tag{18}$$

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

S.6. Implementation Details

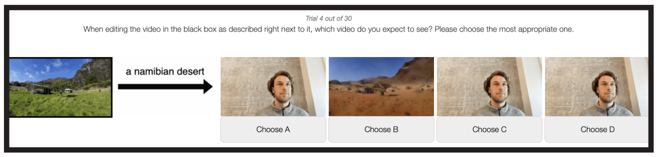
In this section, we provide the implementation details of NeRF and SVG editing presented in Section 4.1 and Section 4.2 of the main paper, respectively.

NeRF Editing. We run the PDS optimization for 30,000 iterations with classifier-free guidance [6] weights within $[30, 100]$ depending on the complexity of editing. As detailed in Section S.5, 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,


Trial 14 out of 30

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.

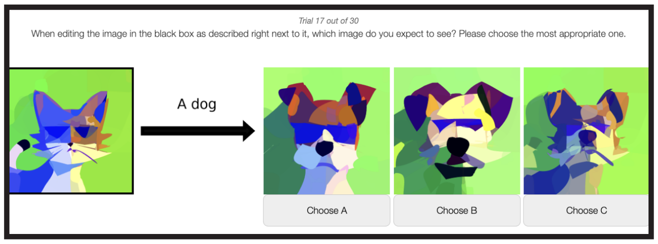
(a) Main problem


Trial 4 out of 30

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.

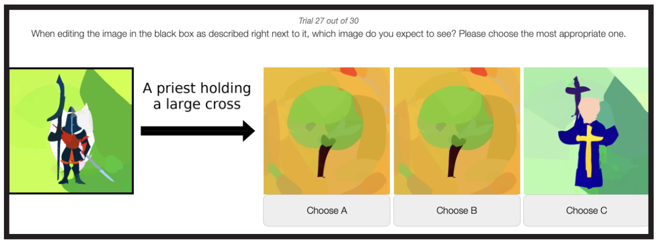
(b) Vigilance task

Figure S4. **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.`


Trial 17 out of 30

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.

(a) Main problem


Trial 27 out of 30

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.

(b) Vigilance task

Figure S5. **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.`

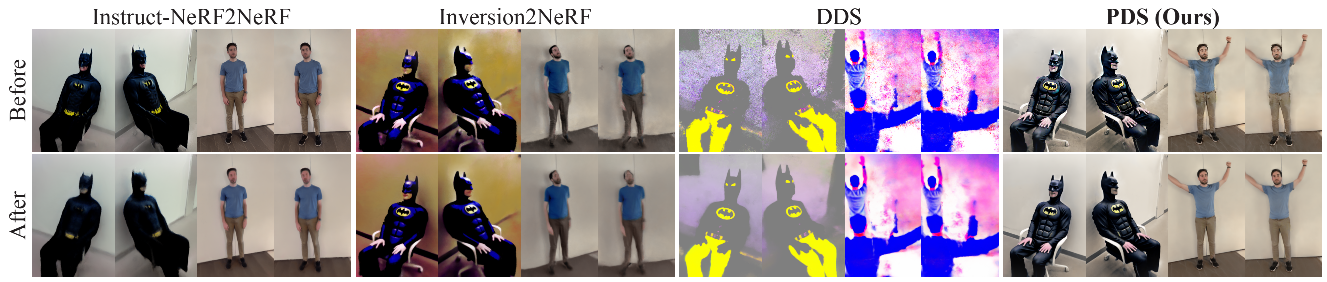


Figure S6. **The effect of the refinement stage.** The overall editing outcomes are determined before the refinement stage, whereas the refinement stage plays the role of removing artifacts. The target attributes are *"Batman"* and *"raising the arms."*

$[1, ..., T]$. 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 [14] 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 [19], DDS [5], and our proposed PDS, we apply the same classifier-free guidance weight of 100. For SDS [19], we sample t within a ratio range of $[0.05, 0.95]$ following VectorFusion [10]. For DDS [5], we follow its original setup, sampling t within $[0.02, 0.98]$. For PDS, we sample i out of a ratio range of $[0.1, 0.98]$.

S.7. Details of User Studies

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. To design our vigilance tasks, we create examples where, except for the correct answer choice, all other choices are replaced with ones from different scenes or unrelated SVG examples. Screenshots of our NeRF and SVG editing user studies, including examples of vigilance tasks, are displayed in Figure S4 and Figure S5, respectively. In the NeRF and SVG editing user studies, we received 42 and 17 valid responses, respectively.

S.8. Effect of the Refinement Stage

Figure S6 illustrates an ablation study of the refinement stage across various editing methods. As depicted, the desired complex edits — making the man raise his arms — are achieved solely through the optimization of PDS. The overall editing outcomes are realized before the refinement stage, and the refinement stage further enhances the fidelity of the outputs.

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. 1

[2] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning to follow image editing instructions. In *CVPR*, 2023. 1, 2, 3

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 2021. 2

[4] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D scenes with instructions. In *ICCV*, 2023. 1, 2, 3, 4

[5] Amir Hertz, Kfir Aberman, and Daniel Cohen-Or. Delta denoising score. In *ICCV*, 2023. 1, 2, 6

[6] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 5

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 2

[8] Inbar Huberman-Spiegelglas, Vladimir Kulikov, and Tomer Michaeli. An edit friendly DDPM noise space: Inversion and manipulations. *arXiv preprint arXiv:2304.06140*, 2023. 2, 3

[9] Shir Iluz, Yael Vinker, Amir Hertz, Daniel Berio, Daniel Cohen-Or, and Ariel Shamir. Word-as-image for semantic typography. *ACM TOG*, 2023. 1

[10] Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *CVPR*, 2023. 1, 6

[11] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *CVPR*, 2023. 4

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023. 1, 4

[13] 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. 1

[14] 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. 4, 6

[15] 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. 1

[16] 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. 1, 2, 3

[17] 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. 1, 2

[18] 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. 1

[19] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3D using 2D diffusion. In *ICLR*, 2023. 1, 2, 6

[20] 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

[21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 3, 4

[22] 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. 4

[23] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 2, 5

[24] 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. 1

[25] 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. 1

[26] 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. 2

[27] 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. 1

[28] 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. 1

[29] Joseph Zhu and Peiye Zhuang. HiFA: High-fidelity text-to-3D with advanced diffusion guidance. *arXiv preprint arXiv:2305.18766*, 2023. 1

[30] 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. 1