NeuSEditor: From Multi-View Images to Text-Guided Neural Surface Edits

Nail Ibrahimli, Julian F. P. Kooij, Liangliang Nan
Delft University of Technology
Julianalaan 134, Delft 2628BL, The Netherlands
{n.ibrahimli, j.f.p.kooij, liangliang.nan}@tudelft.nl

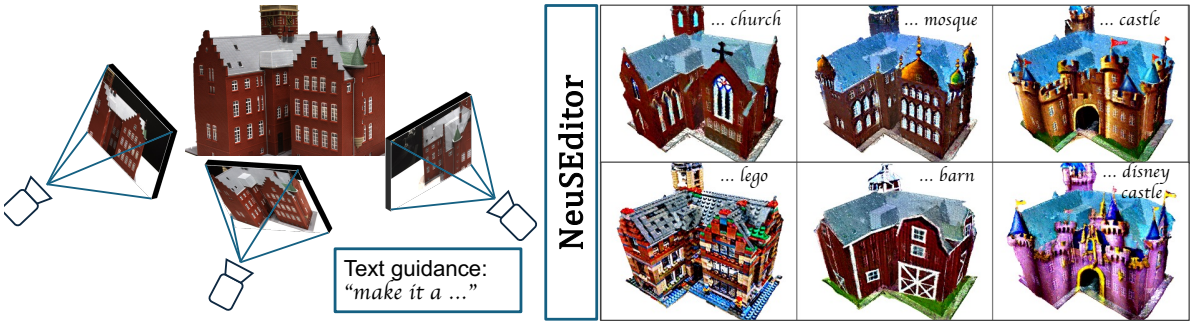

Figure 1. NeuSEditor is a novel method for text-guided neural surface editing from multi-view calibrated images. It generates an edited scene represented as a neural implicit field. Here, the sample inputs on the left are used to generate various 3D textured meshes.

Abstract

Implicit surface representations are valued for their compactness and continuity, but they pose significant challenges for editing. Despite recent advancements, existing methods often fail to preserve identity and maintain geometric consistency during editing. To address these challenges, we present NeuSEditor, a novel method for text-guided editing of neural implicit surfaces derived from multi-view images. NeuSEditor introduces an identity-preserving architecture that efficiently separates scenes into foreground and background, enabling precise modifications without altering the scene-specific elements. Our geometry-aware distillation loss significantly enhances rendering and geometric quality. Our method simplifies the editing workflow by eliminating the need for continuous dataset updates and source prompting. NeuSEditor outperforms recent state-of-the-art methods, delivering superior quantitative and qualitative results. For visual results, visit: neuseditor.github.io.

1. Introduction

Recent advancements in neural rendering techniques have significantly enabled realistic 3D scene capture from multi-view images [25, 35, 39, 64]. These approaches are gaining popularity because of their ease of use and superior output quality over traditional photogrammetry [14, 28, 29, 36, 57, 68]. These methods, typically reliant on calibrated images obtained through Structure from Motion (SfM) pipelines, offer a more photorealistic digital twin of scenes [48].

In parallel, diffusion-based generative models [18, 19, 53–55] have sparked rapid progress in text-conditioned generation and editing across various domains, spanning 2D images [2, 4, 33, 44–46], 3D objects [7–10, 23, 38, 52], and audio [12, 21, 63]. Driven by large datasets of images and textual descriptions, 2D diffusion models have demonstrated strong generative capabilities [27, 42, 49, 50]. As a result, leveraging 2D diffusion priors has become a common approach for generating and editing 3D content from text [6, 15, 16, 26, 30, 31, 34, 41, 43, 51, 60, 72].

However, editing implicit 3D scenes using text prompts presents several unresolved challenges. First, implicit geometry representations encapsulate the entire scene within neural network weights, often leading to inconsistent outputs that complicate precise local edits [11, 26, 30, 73] (e.g., bowtie addition in Fig. 6). Second, view-based optimizations distort the scene from certain perspectives, introducing artifacts like floaters and multi-view inconsistencies [24, 40, 51] (mitigated in our Supp. videos). Also, these issues, combined with the iterative nature of current meth-

Figure 2. Our network architecture retains information about the identity (learned source) of the original scene, including background and foreground details, while simultaneously learning the edited (target) render and geometry. Text prompt: *turn it to a polar bear*.

ods, can result in catastrophic forgetting, where new edits overwrite or degrade previously learned scene features.

In this paper, we propose a novel approach to address these challenges, thereby ensuring stable performance across diverse datasets. Unlike prior methods that depend on additional loss terms [17, 26] or require dataset updates [16, 58], our network architecture explicitly tackles the challenge of identity preservation (by maintaining the essential and recognizable characteristics of the input scene), while also incorporating "additive learning" techniques [20, 70] for incremental edits. Furthermore, we introduce geometry-aware distillation which is effective with a single edit prompt [22, 61]. As shown in Fig. 1, our proposed NeuSEditor enables Signed Distance Function (SDF)-based editing by utilizing multi-view images alongside simple editing text prompts. Our method is capable of generating accurate 3D textured meshes and renders across a variety of datasets, including DTU [1], Blender [35], and IN2N-data [16], which experimental results have validated.

The main contributions of this work are (1) a new implicit network architecture that preserves the identity of the input scene via separate source/target hash grids and conditioning (see Sec. 4); (2) a novel geometry-aware distillation loss for 3D editing (see Secs. 4 and 5.2); and (3) a method that targets foreground edits without introducing undesired changes to the background, offering precise control over the editing process (detailed in Sec. 7).

2. Related works

Differentiable rendering techniques have significantly advanced novel view synthesis and 3D reconstruction, starting with Neural Radiance Fields (NeRF). NeRF represents scenes as radiance fields for photorealistic rendering. However, NeRF is computationally intensive, requires long training times, and is unsuitable for precise geometry extraction due to its focus on volumetric color density rather than explicit surface modeling [32, 35, 39, 64]. To address the limitations in geometry representation, SDF-based volume rendering methods have been proposed [59, 64, 65], allowing for more accurate surface reconstruction.

To accelerate rendering and training, explicit representations such as voxel-based approaches and 3D point or Gaussian-based methods have been introduced [25, 56, 62, 66, 67]. Despite their performance advantages, these techniques require substantially more memory than implicit neural representations. To balance quality and computational efficiency, methods like Instant Neural Graphics Primitives (Instant NGP) [36] and Tensorial Radiance Fields (TensorRF) [5] have been introduced. Instant NGP accelerates the training of implicit neural representations using multiresolution hash encoding. TensorRF represents scenes with compact tensors, enabling efficient rendering and high-quality visual results. These approaches demonstrate that implicit representations can be fast and memory-efficient while maintaining high rendering quality.

We aim to retain the benefits of implicit representations while enabling effective editing without the storage overhead of explicit models. For example, in Fig. 1, learning the scene's identity requires around $28M$ parameters ($14M$ for the foreground and $14M$ for the background), with an additional $14M$ per edit. Thus, making N edits requires learning $28M + 14M \times N$ parameters.

Text-guided neural rendering editing gained popularity with the Instruct-Nerf2Nerf method [16]. This method relies on the convergence of iterative dataset updates using the instruct-pix2pix model [4]. Instruct-GS2GS [58] achieves similar performance by replacing the underlying NeRF representation with Gaussian splatting.

DreamFusion [41] introduced Score Distillation Sampling (SDS) for text-to-3D generation. DDS [17] extends SDS for editing by aligning noise predictions between source and target images, using both source and target prompts. Inspired by recent inversion methods [22, 61], PDS [26] adapts stochastic latent alignment for 3D neural editing, minimizing discrepancies between source and target scene posterior latents, also relying on both prompts.

Our method, unlike Instruct-Nerf2Nerf and Instruct-GS2GS, avoids iterative dataset updates. Unlike ours, both DDS and PDS require two text prompts to depict the source and target for noise and latent estimation. As shown in Fig. 2, unlike previous methods, our architecture explicitly disentangles the background from the foreground and sep-

arates identity (learned source) from edits (target). By retaining the scene identity within the network, our method not only alleviates catastrophic forgetting but also reduces the need for source prompting. In all experiments, we use default empty source prompts unless stated otherwise (e.g., Sec. 9, Fig. 10), highlighting our architecture's identity preservation.

3. Preliminaries

In this section, we review the key concepts and notations that underpin our approach. In particular, we outline neural implicit representations and signed distance functions (SDFs), and discuss the diffusion-based editing methods.

3.1. Neural implicit surfaces and rendering

Neural implicit representations commonly encode scene structure and appearance in neural network weights instead of explicit geometry [57, 68]. For better capturing implicit geometry, the *signed distance function* (SDF) serves as a key representation. For any point $\mathbf{x} \in \mathbb{R}^3$, the SDF, denoted by $\delta(\mathbf{x})$, returns the signed distance to the nearest surface:

$$\delta(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{S}} \|\mathbf{x} - \mathbf{y}\| - r, \tag{1}$$

where $\mathcal{S}$ is the set of surface points and r is a constant offset. The implicit surface is defined by the zero-level set $\{\mathbf{x} \mid \delta(\mathbf{x}) = 0\}$. This formulation allows for smooth and detailed surface extraction via *volume rendering* [35]. In particular, the color $\mathbf{c}$ along a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ (with origin $\mathbf{o}$ and direction $\mathbf{d}$) is given by:

$$\mathbf{c}(\mathbf{r}) = \int_{t_{near}}^{t_{far}} T(t)\, \sigma(\mathbf{r}(t))\, \mathbf{c}(\mathbf{r}(t), \mathbf{d})\, dt, \tag{2}$$

with the transmittance defined as

$$T(t) = \exp\left(- \int_{t_{near}}^{t} \sigma(\mathbf{r}(s))\, ds \right), \tag{3}$$

where $\sigma(\mathbf{r}(t))$ is the volume density (which can be derived from δ [59, 65]) and $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$ is the view-dependent color.

Optimizing these representations can be computationally intensive. To address this, *hash encoding* [29, 36] is employed to map 3D coordinates to a multiresolution hash grid storing feature vectors at grid vertices. These features are efficiently interpolated for any query point $\mathbf{x}$ and then fed into a lightweight MLP to predict both $\delta(\mathbf{x})$ and $\mathbf{c}(\mathbf{x}, \mathbf{d})$, significantly accelerating training and inference.

For *foreground geometry*, our approach is inspired by Neuralangelo [29]: we model the SDF using progressive hash encoding combined with NeuS's rendering equations. In contrast, for *background regions* where SDF cannot be computed, we adopt an Inverted Sphere Parameterization [69] along with hash encoding to effectively capture density fields in unbounded scenes.

3.2. Diffusion models for 3D editing

Diffusion models have recently achieved impressive results in text-to-image generation [19, 45]. In these models, an initial image x_0 is progressively perturbed by Gaussian noise over t timesteps, yielding a noisy image x_t. The forward process is defined as:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \tag{4}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ and $\boldsymbol{\epsilon}$ is standard Gaussian noise.

The reverse process, which recovers x_0 from x_t, is performed via iterative denoising. In DDPM, a single reverse step is given by:

$$\mu_\phi(x_t, y) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}}\, \epsilon_\phi(x_t, y) \right), \tag{5}$$

$$x_{t-1} = \mu_\phi(x_t, y) + \sigma_t\, \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I}), \tag{6}$$

where ϕ denotes the parameters of the diffusion network, $\epsilon_\phi(x_t, y)$ is the predicted noise, $\sigma_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}(1 - \alpha_t)$, y is text prompt, and $\mathbf{z}$ represents standard Gaussian noise.

Score distillation sampling (SDS). Building on diffusion models, *DreamFusion* [41] introduced *score distillation sampling* (SDS) to guide 3D optimization using 2D diffusion priors. Given a rendered image $x_0 = g(\theta)$ from 3D parameters θ, SDS minimizes:

$$\nabla_\theta \mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,\epsilon} \left[w(t)\, (\epsilon_\phi(x_t, y) - \epsilon)\, \frac{\partial x_0}{\partial \theta} \right], \tag{7}$$

where x_t is a noised version of x_0, y is a text prompt, and $w(t)$ is a time-dependent weighting function. While SDS enables text-to-3D generation, it struggles with *editing tasks*, often causing catastrophic forgetting of the original scene [17, 41].

Delta denoising score (DDS). To address SDS's limitations in editing, [17] proposed *delta denoising score* (DDS). Instead of comparing $\epsilon_\phi(x_t, y)$ to random noise ϵ, DDS computes the *difference* between noise predictions for source (x_0^{src}) and target (x_0^{tgt}) images:

$$\nabla_\theta \mathcal{L}_{\text{DDS}} = \mathbb{E}_{t,\epsilon_t} \left[w(t)\, \left(\epsilon_\phi(x_t^{\text{tgt}}, y^{\text{tgt}}) - \epsilon_\phi(x_t^{\text{src}}, y^{\text{src}}) \right) \frac{\partial x_0^{\text{tgt}}}{\partial \theta} \right]. \tag{8}$$

While DDS demonstrated potential in 2D image editing, its direct application to 3D scene editing often leads to failure to preserve structural coherence [26].

Posterior distillation sampling (PDS). The stochastic latent $\mathbf{z}$ in Eq. (6), often termed the *posterior stochastic latent*, encodes structural information critical for preserving scene coherence during editing [22, 61]. Rearranging Eq. (6), $\mathbf{z}$ is derived as:

$$\mathbf{z} = \frac{x_{t-1} - \mu_\phi(x_t, y)}{\sigma_t}. \tag{9}$$

Building on this insight, *posterior distillation sampling* (PDS) [26] adapts stochastic latent alignment to 3D neural editing by minimizing the discrepancy between source and target scene posterior latents:

$$\mathcal{L}_{\text{PDS}} = \mathbb{E}_{t,\epsilon_t,\epsilon_{t-1}\sim\mathcal{N}(0,\mathbf{I})} \left\| \mathbf{z}_t^{\text{tgt}} - \mathbf{z}_t^{\text{src}} \right\|_2^2, \tag{10}$$

where $\mathbf{z}_t^{\text{src}}$ and $\mathbf{z}_t^{\text{tgt}}$ denote the posterior latents of the source (unedited) and target (edited) scenes.

4. Method

Our method follows a two-stage approach. In the first stage, our network learns the identity of the source scene depicted in the input images. Similar to neural rendering techniques [35, 65], the network queries 3D points along with their corresponding view directions to generate a colored implicit field. In the second stage, after capturing the scene identity, the network shifts its focus to the editing process (target learning) while preserving the original identity. Fig. 3 provides an overview of our pipeline.

4.1. Identity (source) learning

Our approach captures the identity of a scene, its geometry and appearance, by combining progressive hash encodings [29] with SDF-based volume rendering [59]. At the core of this method is an MLP network that predicts both the signed distance (SDF) and color values for each 3D point. We train the MLP-based source renderer using photometric and eikonal losses [13]. Through this process, the scene's identity is fully encoded in the parameters of the hash grids and the weights of the MLP-based source renderer networks.

4.2. Editing process (target learning)

To edit the geometry, we introduce a second set of 3D hash features. These features are fed into an MLP-based target renderer that is conditioned on the source renderer, which encodes SDF geometry and geometric feature vector at each 3D point. This conditioning allows the target renderer to access richer features for effective scene editing.

We employ an additive learning strategy [20, 70] to enhance the editing process. Specifically, we initialize the target renderer networks so that at the start of the editing process, the target geometry and color match the identity. This initialization encourages target networks to focus on learning the essential modifications during the editing process. The target renderer networks are trained using a geometry-aware distillation process that leverages a diffusion model.

4.3. Geometry-aware distillation

We present a novel, geometry-aware method for stochastic latent distillation. Our approach begins by computing the stochastic latent variables as defined in Eq. (9):

$$\mathbf{z}_t^{\text{src}} = \mathbf{z}_t(x_0^{\text{src}}, y^{\text{src}}; \epsilon_\phi), \quad y^{\text{src}} \in \{\text{source prompt}, \emptyset\} \tag{11}$$

$$\mathbf{z}_t^{\text{tgt}} = \mathbf{z}_t(x_0^{\text{tgt}}, y^{\text{tgt}}; \epsilon_\phi) \tag{12}$$

$$\hat{\mathbf{z}}_t^{\text{tgt}} = \hat{\mathbf{z}}_t(\text{Phong}_0^{\text{tgt}}, y^{\text{tgt}}; \epsilon_\phi), \tag{13}$$

where $\mathbf{z}_t^{\text{src}}$ represents the stochastic latent of the source image at timestep t, with y^{src} denoting the source prompt. The variable y^{tgt} indicates the editing prompt, while $\mathbf{z}_t^{\text{tgt}}$ corresponds to the latent representation for the target image. Additionally, to enhance and promote geometric edits, we employ a Phong shading renderer. $\hat{\mathbf{z}}_t^{\text{tgt}}$ captures the stochastic latent associated with Phong shading of the target geometry. Computing Phong shading requires target SDF gradients. We employ *numerical* gradients for cleaner Phong shading compared to *analytical* gradients (details in Supp. Sec. 10). Since the Diffusion model requires a three-channel image, we convert the Phong shading to a three-channel image by simply repeating it across all three channels.

Other methods [17, 26, 37] typically involve two prompts to incorporate input (source) information. However, using two different prompts without explicit identity-preserving mechanism can easily steer the optimization in different directions. To alleviate this issue, PDS [26] employs two very similar prompts with keyword replacement or addition for editing purposes. In contrast, our method does not even require a source prompt, as we address identity preservation directly through the network architecture.

Our overall objective can be expressed as follows:

$$\mathcal{L}_{\text{PDS}} = \mathbb{E}_{t\sim U(0,1);\, \epsilon_t,\epsilon_{t-1}\sim N(0,I)} \left\| \mathbf{z}_t^{\text{tgt}} - \mathbf{z}_t^{\text{src}} \right\|_2^2 \tag{14}$$

$$\mathcal{L}_{\text{PE}} = \mathbb{E}_{t\sim U(0,1);\, \epsilon_t,\epsilon_{t-1}\sim N(0,I)} \left\| \hat{\mathbf{z}}_t^{\text{tgt}} - \mathbf{z}_t^{\text{src}} \right\|_2^2 \tag{15}$$

$$\mathcal{L}_{\text{PEPDS}} = \lambda_{\text{PDS}} \mathcal{L}_{\text{PDS}} + \lambda_{\text{PE}} \mathcal{L}_{\text{PE}}, \tag{16}$$

where $\mathcal{L}_{\text{PEPDS}}$ is the overall loss function, λ_{PDS} and λ_{PE} are weighting factors for the first and second terms, respectively. $\mathcal{L}_{\text{PDS}}$ matches the stochastic latents of the source and target images, while $\mathcal{L}_{\text{PE}}$ aligns the stochastic latents of the source and target Phong shading images. The addition of Phong shading-based distillation significantly reduces floaters in geometry (see also Sec. 5.2). To avoid the costly computation associated with the Diffusion UNet [45, 47], we directly compute the gradient of the loss function with respect to the (geometry and) renderer parameters θ, as suggested in SDS [41], while omitting the U-Net Jacobian:

$$\nabla_\theta \mathcal{L}_{\text{PDS}} = \mathbb{E}_{t,\epsilon_t,\epsilon_{t-1}} \left[w(t)(\mathbf{z}_t^{\text{tgt}} - \mathbf{z}_t^{\text{src}}) \frac{\partial x_0^{\text{tgt}}}{\partial \theta} \right] \tag{17}$$

$$\nabla_\theta \mathcal{L}_{\text{PE}} = \mathbb{E}_{t,\epsilon_t,\epsilon_{t-1}} \left[w(t)(\hat{\mathbf{z}}_t^{\text{tgt}} - \mathbf{z}_t^{\text{src}}) \frac{\partial x_0^{\text{tgt}}}{\partial \theta} \right] \tag{18}$$

$$\nabla_\theta \mathcal{L}_{\text{PEPDS}} = \lambda_{\text{PDS}} \cdot \nabla_\theta \mathcal{L}_{\text{PDS}} + \lambda_{\text{PE}} \cdot \nabla_\theta \mathcal{L}_{\text{PE}}, \tag{19}$$

where $w(t)$ is a weighting function depending on timestep [19] and x_0^{tgt} is the target image.

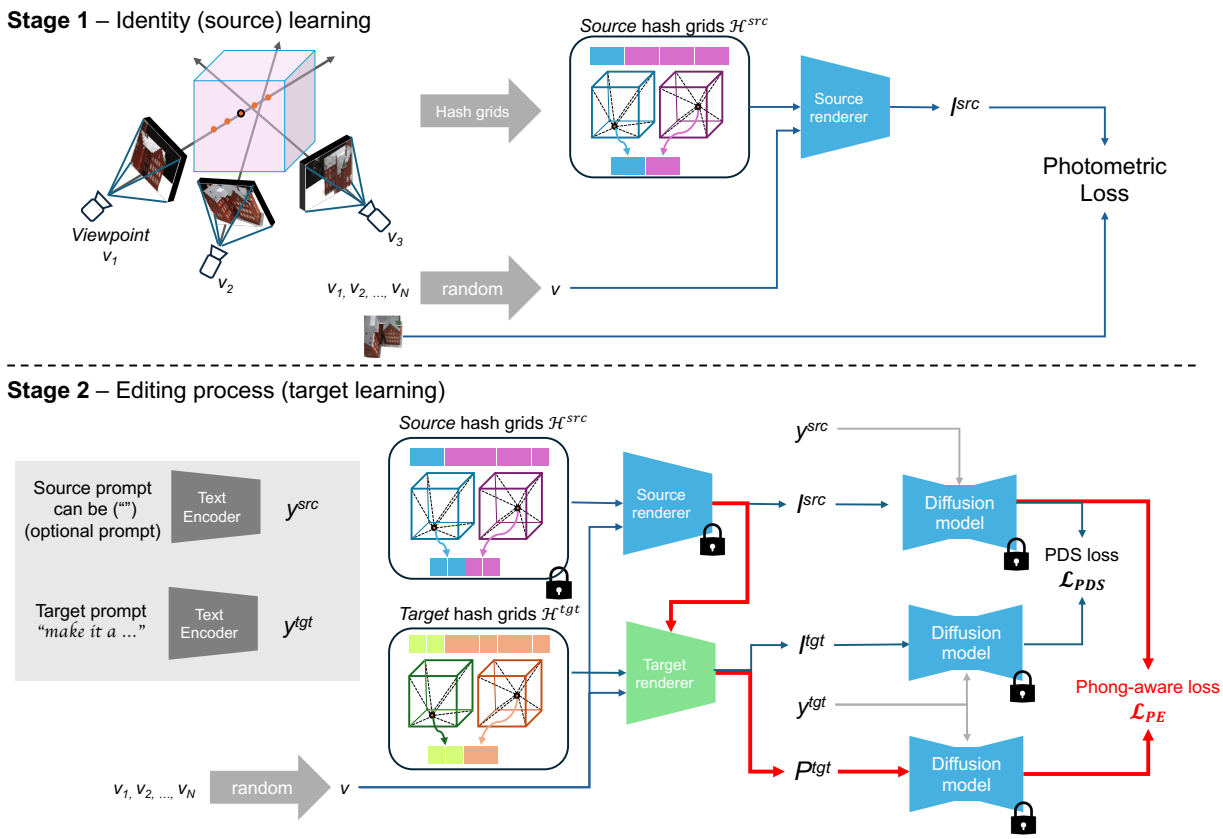

Figure 3. Overview of our two stage scene editing pipeline. In the first stage (source learning) the network captures the scene's identity, including its geometry and appearance, using SDF based volume rendering with progressive hash encodings. In the second stage (target learning) a target renderer, conditioned on the source renderer and guided by a geometry aware distillation loss, edits the scene. Here, I_{src} and I_{tgt} denote the source and target renderings respectively, while P_{tgt} represents Phong shading. Within the pipeline, the red arrows highlights the transfer of knowledge between source and target, the integration of Phong shading, and its role in the loss formulation.

5. Experiments

5.1. Experimental Details

Experiments ran on a single A100 GPU, with training times (1.5–6h) comparable to PDS_{NeRF}. Duration depends on (1) image resolution, (2) foreground pixel coverage, and (3) prompt complexity (e.g., Bear: 1.5h; DTU: 4–6h, including reconstruction + editing). One of the crucial hyperparameters is the guidance scale [18]. As illustrated in Fig. 4, a guidance scale of 350 generally yields superior results. It is noteworthy to observe how increasing the guidance scale turns the tie to the suit (top row) and gives the ketchup and mustard a fruity appearance (bottom row).

5.2. Ablation study

We conducted an ablation study to validate the contributions of our architecture and geometry-aware loss. In all experiments, we employed simple Hash Grids and analytical SDF gradient computations to exclusively assess the architecture and our loss. Fig. 5 summarizes the qualitative results.

Baseline. The top row shows the naive baseline, which uses Hash Grids for positional encoding [36], NeuS as the renderer [59], and PDS as the distillation loss [26]. The render view, depth map, and normal map reveal that this combination fails to preserve the original scene's identity and produces noisy geometry.

Architecture Enhancement. The middle row demonstrates that incorporating our architecture significantly improves geometry and rendering quality, although some floating artifacts [40] remain.

Loss Improvement. The bottom row highlights the impact of our loss objective: by employing Phong-aware distillation, the fidelity of fine details is enhanced and the number of floaters is drastically reduced.

We evaluated the ablations using both CLIP [42] and LPIPS [71] metrics (see Tab. 1). Our analysis of foreground renderings shows that our architecture and loss function improve the CLIP similarity score. To evaluate catastrophic forgetting, we trained a separate NeRF model to render videos from input images. Then, we computed the LPIPS

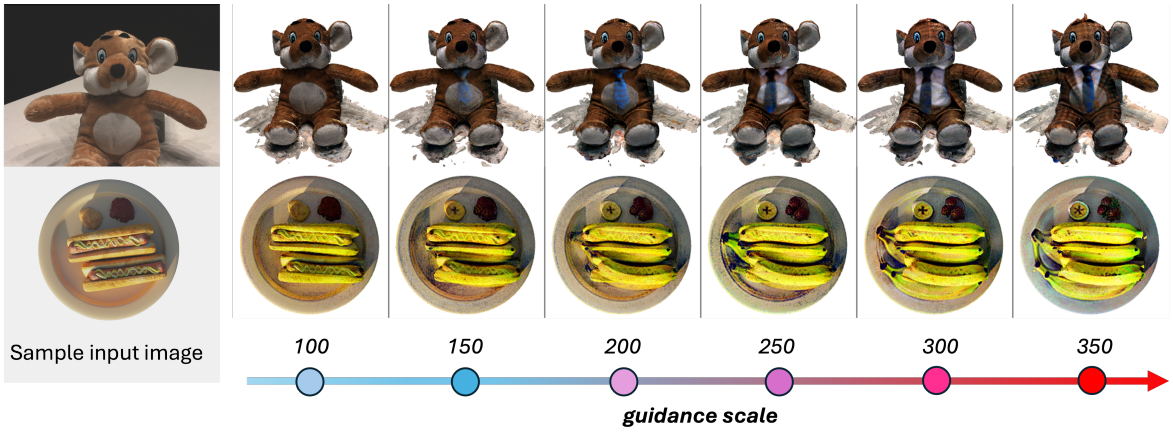

Figure 4. Influence of hyperparameter guidance scale. Left: two sample input images from the DTU dataset (scan105 scene) and the Blender dataset (hotdog scene), respectively. Right: the edited mesh renderings of the corresponding scenes at guidance scales from 100 to 350. The text prompt for the top row is '*add suit*', while the prompt for the bottom row is '*make it bananas*'.

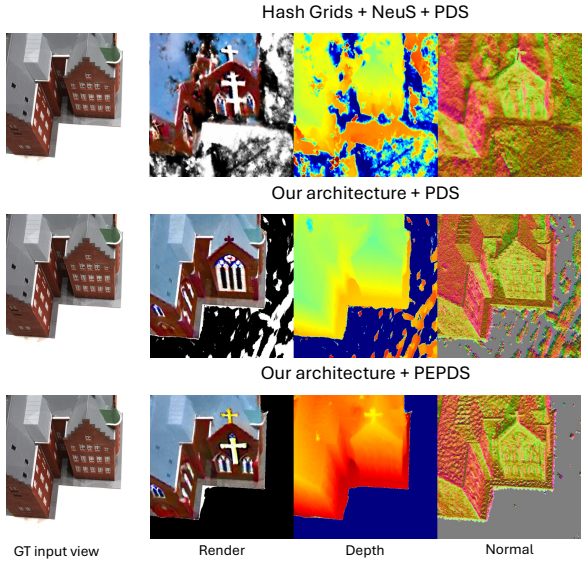

Figure 5. The left column displays an input view from the DTU dataset scan24, while the right column shows the foreground render, depth map, and normal map. The top row presents results from the naive Hash Grids + NeuS + PDS loss approach. The middle row shows our architecture with PDS loss, and the bottom row shows our architecture with Phong Enhanced PDS loss ($\lambda_{\text{PDS}} = 1.$ and $\lambda_{\text{PE}} = 0.2$). The text prompt is '*make it a church*'.

distance between the edited rendering and the reference rendering from the separate NeRF model ("input" scene). The results show a significant reduction in distance, indicating better identity preservation, which is due to our architecture. Additionally, we calculate the LPIPS distance between consecutive frames, expecting floaters and inconsistencies to increase the distance. Results show that both the proposed architecture and loss contribute to better consistency.

Metric	Backbone	Baseline	Arch + PDS	Ours
clip ↑	ViT/16	0.268	0.276	0.283
	ViT/32	0.261	0.270	0.276
lpips ↓	Alex	0.721	0.685	0.700
	VGG	0.720	0.676	0.683
lpips (MV) ↓	Alex	0.133	0.112	0.101
	VGG	0.231	0.187	0.177

Table 1. Quantitative ablation study results based on CLIP and LPIPS metrics. The final row reports the evaluation of multi-view consistency between consecutive frames using the LPIPS metric.

5.3. Dataset and benchmark

Our evaluation spans three datasets: DTU [1] (6 scenes, 11 experiments), Blender [35] (3 scenes, 7 experiments), and IN2N [16] (2 scenes, 6 experiments), totaling 11 scenes and 34 editing experiments. To ensure fair evaluation, we rendered videos using spherical interpolation of camera poses, setting the radius to the average distance from the scene origin. This approach was applied to all DTU scenes and the *bear* scene from the IN2N dataset (see Sec. 11). The *person* scene in the IN2N dataset was not suitable for this strategy, so we adopted a path similar to the default IN2N. For Blender, we followed the predefined test dataset trajectory.

Metric	Backbone	IN2N	PDS_{NERF}	PDS_{GS}	Ours
clip ↑	ViT/16	0.263	0.272	0.296	0.297
	ViT/32	0.264	0.272	0.293	0.294
lpips ↓	Alex	0.427	0.412	0.432	0.306
	VGG	0.443	0.415	0.438	0.344

Table 2. The top row shows text-image similarity, and the bottom shows perceptual similarity between the input and edited scenes.

5.3.1 Objective Metrics

To quantify editing performance, we computed text-image consistency and perceptual similarity-based metrics across all methods (Tab. 2). Specifically, we used CLIP scores with PDS text prompts over 34 editting tasks. Our method performs on par with the explicit approach (PDS_{Splat}) while outperforming implicit methods (IN2N, PDS_{NeRF}).

Additionally, similar to our ablation experiments, we trained a separate NeRF model to render videos from input images across all experiments. For each of the 34 editting tasks, we computed the LPIPS distance between the 3D-edited rendering and the reference (unedited) rendering from the separate NeRF model. High LPIPS distances indicate catastrophic forgetting in the edited scene. Our analysis reveals that our method preserves the original scene significantly better than competing approaches.

5.3.2 User study

We conducted a user study with 41 participants, each completing 34 experiments. In each experiment, participants viewed rendering videos from our model, PDS_{NeRF} [26, 57], PDS_{Splat} [26, 66], and IN2N [16, 57], alongside original scene images and editing text prompts. The video order was randomized to eliminate bias, and participants ranked each video on a scale from 0 (lowest quality) to 3 (highest quality). The evaluation prompt stated:

"Considering the original scene images and editing text prompts, assign an overall score to the following scenes in terms of geometry, editing quality, and clarity while preserving the features of the original scene."

Tabs. 3 and 4 summarize results across datasets. Fig. 6 shows survey samples with a reference image and two frames from four models: one with minimal noise and another highlighting floating artifacts affecting user choices. Users preferred NeuSEditor in 82% of the survey experiments (28 out of 34 edits; see Supp. Tab. 7).

Both quantitative and qualitative results show that our method and PDS_{Splat} outperform others on DTU, while IN2N struggles with degenerate cases and PDS_{NeRF} produces substantial floating artifacts. In Blender, our method and PDS_{Splat} again outperform on *hotdog* and *mic* scenes, while PDS_{NeRF} fails. IN2N surpasses PDS_{NeRF} due to fewer floating artifacts. In *ficus*, PDS_{Splat} encounters multi-view inconsistencies, IN2N struggles, and PDS_{NeRF} aligns better with text prompts. For the IN2N dataset, IN2N achieves the best performance. In *person*, all methods produce clean results, with users favoring PDS_{Splat} and IN2N for identity preservation. In *bear*, our method and IN2N yield comparable results, with users slightly preferring ours for mitigating the "Janus artifact." PDS_{Splat} struggles with multi-view consistency, and PDS_{NeRF} displays monotonic color artifacts. See supplementary and the survey page replica for details.

Dataset	IN2N	PDS_{NeRF}	PDS_{Splat}	Ours
DTU	0.52	1.07	1.87	2.54
Blender	1.08	0.71	1.53	2.69
IN2N-data	2.38	0.63	1.39	1.61
Avg. per dataset	1.33	0.80	1.60	2.28
Avg. per experiment	1.00	0.92	1.71	2.41

Table 3. Survey results across datasets. The last rows show averages per dataset and all 34 experiments. (Higher is better ↑)

Dataset - Scene	IN2N	PDS_{NeRF}	PDS_{Splat}	Ours
DTU - Scan24	0.18	1.26	1.97	2.59
DTU - Scan65	0.92	0.80	1.58	2.69
DTU - Scan83	0.07	1.95	1.76	2.22
DTU - Scan105	0.55	1.23	1.80	2.41
DTU - Scan106	0.52	1.18	1.67	2.62
DTU - Scan110	0.83	0.30	2.41	2.46
Blender - Hotdog	1.09	0.39	1.77	2.76
Blender - Mic	1.28	0.17	2.07	2.48
Blender - Ficus	0.93	1.28	1.00	2.78
IN2N - Person	2.37	0.65	2.20	0.79
IN2N - Bear	2.39	0.60	0.58	2.43
Avg. per scene	1.01	0.89	1.71	2.38

Table 4. Average user score per scene. (Higher is better ↑)

5.4. Limitations

NeuSEditor shares some common limitations with 3D generative AI pipelines. First, these pipelines generally require high CFG weights, which can steer optimization toward oversaturated colors [41]. Additionally, our method is constrained by the inherent biases of underlying diffusion models, making it unable to completely avoid Janus artifacts or content drift issues [24, 51].

6. Conclusion

In this paper, we presented NeuSEditor, a novel approach for text-guided editing of neural implicit surfaces from multi-view images. NeuSEditor introduces a novel identity-preserving architecture combined with a geometry-aware distillation loss, enabling high-quality 3D content editing that outperforms state-of-the-art methods in terms of rendering fidelity and geometric consistency. Our approach preserves scene identity while providing precise edit control. Our empirical results demonstrate that NeuSEditor effectively mitigates common issues such as catastrophic forgetting and degenerate outputs, such as those with monotonic colors, which are common pitfalls in other implicit surface-based methods. We believe that the novel architecture and Phong-based distillation could inspire new approaches for robust and compact 3D content editing, offering a foundation for future advancements in this area.

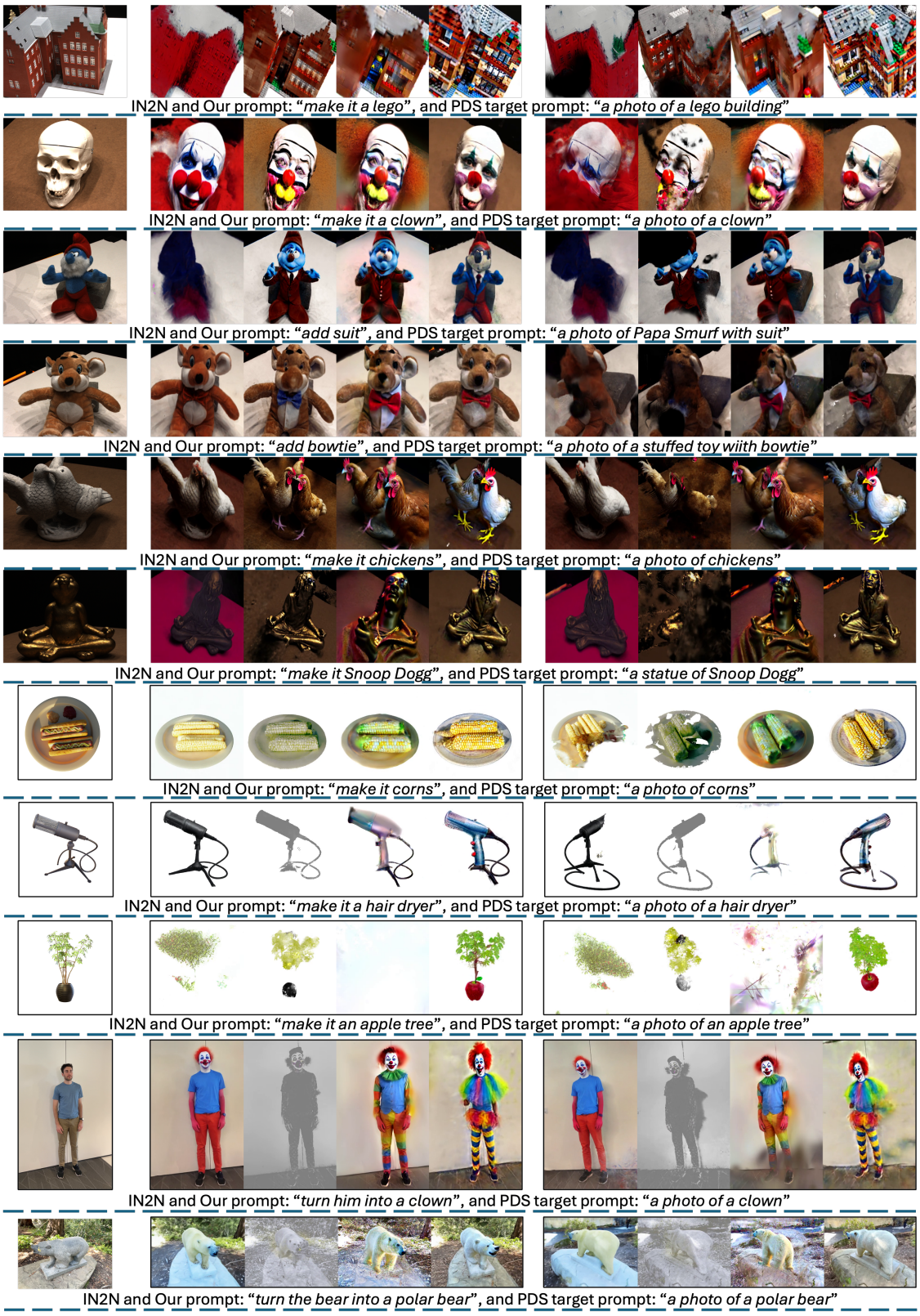

Figure 6. Comparison between IN2N, PDS_{NeRF}, $\text{PDS}_{\text{Splat}}$, and our method (in this order) across two frames. Edit prompts for ours and IN2N, and the target prompts for the PDS models are also provided below each row. We set $\lambda_{\text{PDS}} = 1$ and $\lambda_{\text{PE}} = 0.2$ in all experiments.

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *IJCV*, 2016. 2, 6

[2] Yuval Alaluf, Daniel Garibi, Or Patashnik, Hadar Averbuch-Elor, and Daniel Cohen-Or. Cross-image attention for zero-shot appearance transfer. In *ACM SIGGRAPH*, 2024. 1

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *CVPR*, 2023. 17

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

[5] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 2

[6] 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. 1

[7] Gene Chou, Ilya Chugunov, and Felix Heide. Gensdf: Two-stage learning of generalizable signed distance functions. In *NeurIPS*, 2022. 1

[8] Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-sdf: Conditional generative modeling of signed distance functions. 2023.

[9] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022.

[10] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. *arXiv preprint arXiv:2307.05663*, 2023. 1

[11] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *ICCV*, 2023. 1

[12] 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. 1

[13] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. 2020. 4

[14] Yuan-Chen Guo. Instant neural surface reconstruction, 2022. https://github.com/bennyguo/instant-nsr-pl. 1

[15] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3d content generation. https://github.com/threestudio-project/threestudio, 2023. 1

[16] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D scenes with instructions. In *ICCV*, 2023. 1, 2, 6, 7

[17] Amir Hertz, Kfir Aberman, and Daniel Cohen-Or. Delta denoising score. In *ICCV*, 2023. 2, 3, 4

[18] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 1, 5

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 1, 3, 4

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. 2, 4, 17

[21] 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. 1

[22] Inbar Huberman-Spiegelglas, Vladimir Kulikov, and Tomer Michaeli. An edit friendly DDPM noise space: Inversion and manipulations. In *CVPR*, 2024. 2, 3

[23] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 1

[24] Yash Kant, Aliaksandr Siarohin, Ziyi Wu, Michael Vasilkovsky, Guocheng Qian, Jian Ren, Riza Alp Guler, Bernard Ghanem, Sergey Tulyakov, and Igor Gilitschenski. Spad: Spatially aware multi-view diffusers. In *CVPR*, 2024. 1, 7

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023. 1, 2, 17

[26] Juil Koo, Chanho Park, and Minhyuk Sung. Posterior distillation sampling. In *CVPR*, 2024. 1, 2, 3, 4, 5, 7, 17

[27] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. 2022. 1

[28] Ruilong Li, Hang Gao, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: Efficient sampling accelerates nerfs. *arXiv preprint arXiv:2305.04966*, 2023. 1

[29] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *CVPR*, 2023. 1, 3, 4, 17

[30] 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. 1

[31] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *ICCV*, 2023. 1

[32] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: learning dynamic renderable volumes from images. *ACM TOG*, 2019. 2

[33] 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. 1

[34] 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

[35] 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, 4, 6

[36] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022. 1, 2, 3, 5, 17

[37] Hyelin Nam, Gihyun Kwon, Geon Yeong Park, and Jong Chul Ye. Contrastive denoising score for text-guided latent diffusion image editing. In *CVPR*, 2024. 4

[38] 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. 1

[39] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *CVPR*, 2020. 1, 2

[40] Julien Philip and Valentin Deschaintre. Floaters no more: Radiance field gradient scaling for improved near-camera training. *Eurographics Symposium on Rendering*, 2023. 1, 5

[41] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3D using 2D diffusion. In *ICLR*, 2023. 1, 2, 3, 4, 7

[42] 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. 2021. 1, 5

[43] 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. 1

[44] 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. 1

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 3, 4, 17

[46] 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. 1

[47] RunwayML. Stable diffusion v1.5. https://huggingface.co/stable-diffusion-v1-5/stable-diffusion-v1-5, 2022. 4

[48] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1, 17

[49] 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. 1

[50] 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. In *NeurIPS*, 2022. 1

[51] 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. 1, 7

[52] Jaehyeok Shim, Changwoo Kang, and Kyungdon Joo. Diffusion-based signed distance fields for 3d shape generation. In *CVPR*, 2023. 1

[53] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 1

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

[55] 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. 1

[56] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2

[57] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH*, 2023. 1, 3, 7, 17

[58] Cyrus Vachha and Ayaan Haque. Instruct-gs2gs: Editing 3d gaussian splats with instructions, 2024. 2

[59] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021. 2, 3, 4, 5, 12

[60] 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

[61] 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, 3

[62] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, 2022. 2

[63] 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. 1

[64] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. In *NeurIPS*, 2020. 1, 2

[65] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021. 2, 3, 4

[66] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for Gaussian splatting. *arXiv preprint arXiv:2409.06765*, 2024. 2, 7

[67] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2

[68] Zehao Yu, Anpei Chen, Bozidar Antic, Songyou Peng, Apratim Bhattacharyya, Michael Niemeyer, Siyu Tang, Torsten Sattler, and Andreas Geiger. Sdfstudio: A unified framework for surface reconstruction, 2022. 1, 3

[69] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 3, 12

[70] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. 2, 4, 17

[71] 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. 5

[72] Joseph Zhu and Peiye Zhuang. HiFA: High-fidelity text-to-3D with advanced diffusion guidance. In *ICLR*, 2024. 1

[73] Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In *SIGGRAPH Asia*, 2023. 1