ProEdit: Simple Progression is All You Need for High-Quality 3D Scene Editing

Jun-Kun Chen **Yu-Xiong Wang**
University of Illinois Urbana-Champaign
{junkun3, yxw}@illinois.edu
immortalco.github.io/ProEdit

Figure 1: By decomposing a difficult task into easy subtasks and then progressively performing them (upper part), our ProEdit achieves high-quality 3D editing results with bright colors and detailed textures along with introducing new controllability of the editing aggressivity (lower part). **More results are provided on our project page.**

Abstract

This paper proposes ProEdit – a simple yet effective framework for high-quality 3D scene editing guided by diffusion distillation in a novel *progressive* manner. Inspired by the crucial observation that multi-view inconsistency in scene editing is rooted in the diffusion model's large *feasible output space* (FOS), our framework controls the size of FOS and reduces inconsistency by decomposing the overall editing task into several subtasks, which are then executed progressively on the scene. Within this framework, we design a difficulty-aware subtask decomposition scheduler and an adaptive 3D Gaussian splatting (3DGS) training strategy, ensuring high quality and efficiency in performing each subtask. Extensive evaluation shows that our ProEdit achieves state-of-the-art results in various scenes and challenging editing tasks, *all* through a simple framework *without* any expensive or sophisticated add-ons like distillation losses, components, or training procedures. Notably, ProEdit also provides a new way to control, preview, and select the "aggressivity" of editing operation during the editing process.

1 Introduction

The emergence and advancement of modern scene representation models, exemplified by neural radiance fields (NeRFs) [1] and 3D Gaussian splatting (3DGS) [2], have significantly reduced the difficulty associated with high-quality reconstruction and rendering of large-scale scenes. In addition to reconstructing known scenes, there is growing interest in editing existing scenes to create new ones.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Among the various editing operations, the *instruction-guided scene editing* (IGSE) stands out as one of the most free-form tasks, supporting editing based on simple text descriptions. Due to the lack of 3D supervision data to train editing models in 3D, current state-of-the-art methods tackle IGSE using *2D diffusion distillation*, which involves distilling editing signals from a pre-trained 2D diffusion model [3, 4]. These methods leverage the 2D diffusion model to edit rendered images of scenes from multiple viewpoints, and then reconstruct the edited scene from these edited images using specific distillation losses.

However, a substantial challenge faced by such distillation-based approaches in achieving high-quality scene editing lies in ensuring that the scene representation converges on the edited multi-view images. Failure to achieve so results in gloomy colors, blurred textures, and noisy geometries (*e.g.*, the failure cases from [5]). We argue that **this challenge is rooted in the diffusion model's large *feasible output space* (FOS) for the same instruction** – since a text instruction can be interpreted in different yet plausible ways. For example, "make the person wear a hat" could be implemented with a hat of any style, shape, size, position, *etc*.

Therefore, large FOS is the underlying cause of *multi-view inconsistency* in 2D editing results, making the scene representation – originally designed for reconstructing from consistent images – hard to converge. Previous work, often unaware of this fundamental issue, deals with multi-view inconsistency by introducing inconsistency-robust distillation losses [6, 7] to tolerant inconsistency, or proposing additional components and training procedures [8, 9] to select consistent images from the FOS. While adding costs and complexities, these methods frequently fail to converge to a high-quality scene when the FOS is considerably large, especially for operations that change the scene's geometry.

In overcoming this challenge posed by the large FOS, *our key insight* is to control the FOS size through *editing task decomposition*, as illustrated in Fig. 1. Building on this insight, we propose *ProEdit*, a simple, novel framework to achieve high-quality IGSE, by decomposing the original, large-FOS task into multiple *subtasks* with significantly smaller FOS, and then *progressively* performing high-quality editing for each of these tasks. With each subtask's FOS effectively controlled, they can be solved under a simple solution *without* the need for additional distillation losses, components, or complex training procedures. Progressively solving all these subtasks naturally leads to a high-quality edited scene that meets the requirements of the original task.

To perform subtask decomposition, we introduce an intuitive formulation of "subtasks" with text encoding interpolation. Based on this formulation, we propose a *subtask scheduler* to determine the subtask decomposition and guide the editing process. This decomposition consists of a sequence of subtasks, where each subtask is applied to the edited scene from the previous one. We adaptively assign subtasks according to the estimated FOS size, so that each subtask has comparable FOS sizes and difficulty levels and can thus be solved relatively easily with high quality and efficiency.

Guided by the subtask scheduler, we progressively iterate on the subtasks to apply editing. Though their FOS size and difficulty are controlled, it still remains non-trivial to make the scene representation converge in precise geometry. Failing to achieve this will accumulate errors across subtasks, leading to unreasonable geometry in the final results. To this end, we choose 3D Gaussian splatting (3DGS) [2] as our scene representation for its high training efficiency. We design a novel *adaptive* Gaussian creation strategy in training to maintain and refine the geometric structure in each subtask, by controlling the size of the splitting and duplication operations. This strategy allows the geometry to be adjusted toward the goal of each subtask, while preventing and removing floc, floating noise, and multi-face structures.

With these key designs, our ProEdit achieves high-quality instruction-guided scene editing in various scenes and editing tasks with precise geometry and detailed textures, as shown in Fig. 1. Notably, ProEdit does not rely on complicated or expensive add-ons, such as specialized distillation losses, additional 3D attention or convolution components, or extended training procedures on the diffusion model. Moreover, as each subtask represents a partial completion of the overall task, our method enables users to *control, preview, and select* the intermediate stages of editing, which we refer to as "aggressivity" of editing operation during the editing process. This can be simply achieved by taking the edited scene from a subtask either during or after the editing process. Thus, in contrast to previous methods such as classifier-free guidance [10] and SDEdit [11], our ProEdit provides a novel way to monitor and manage the editing process. Users can *preview* different versions of editing with the intermediate outcomes, adjust the subtasks *on the fly* accordingly to achieve improved final results, and finally *select* the most satisfactory editing result from all the intermediate ones.

Our contributions are three-fold. (1) We offer a novel insight into subtask decomposition and progressive editing, tailored to address the core challenge of large feasible output space in 3D scene editing. (2) We propose a simple yet effective framework, ProEdit, that generates high-quality edited scenes by progressively solving each subtask, without requiring any complicated or expensive add-ons to the diffusion model, while also supporting control, training-time preview, and selection of editing task aggressivity. (3) We consistently achieve high-quality editing results in various scenes and challenging tasks, establishing state-of-the-art performance.

2 Related Work

Learning-Based 3D Scene Representation. Our framework necessitates a learnable 3D representation to depict the scene being edited. Traditional methods model the 3D geometric structure of a scene with implicit [12–14] or explicit [15–17] representations. However, these methods require more information or pre-processing beyond multi-view camera images. In 2020, the neural radiance field (NeRF) [1] emerges as the first neural network-based scene representation, enabling direct scene reconstruction from multi-view images captured at known camera locations, inspiring numerous follow-up work [18–24] that explores different aspects including quality, efficiency, and visual effects. Later, 3D Gaussian splatting (3DGS) [2] becomes a new trend, outperforming NeRF and its variants in rendering quality and efficiency. 3DGS also leads to several follow-up variants, aiming to improve geometry [25, 26] and visual effects [27], as well as extending to dynamic 3D scenes [28–30].

3D Scene Editing. Various scene editing tasks have been investigated, each aiming to achieve different editing objectives for a given scene across a range of scene representations. These tasks cover different aspects of a scene, including the location, shape, and color of objects [20, 31–33], physical effects [34], lighting conditions [27, 35, 36], and the overall appearance [5, 7, 9, 37, 38].

Instruction-Guided Scene Editing. Instruction-guided scene editing is a highly free-form yet challenging task, characterized by a straightforward task descriptor – either an editing operation (*e.g.*, "Give the person a hat") or a description of the desired scene (*e.g.*, "A person wearing a hat"). This task has attracted much attention in the computer vision community. Due to the lack of large-scale 3D datasets to train editing models directly in 3D, current state-of-the-art methods [5–7, 9, 38–41] achieve scene editing by distilling knowledge from a pre-trained 2D diffusion model [3, 5] using score distillation sampling (SDS) [42] and its variants. Instruct-NeRF2NeRF (IN2N) [5] and its variants [37, 39] apply SDS-equivalent iterative dataset updates to generate edited multi-view images and train the scene representation on them. One direction of follow-up work [6, 7] proposes novel distillation methods to better utilize the 2D editing capability, while another [9, 41] introduces additional components and training procedures to improve the consistency of generation. However, these approaches are unaware of the core challenge posed by large feasible output space (FOS), mitigating it with add-ons that may still fail when the FOS becomes considerably large. In contrast, our ProEdit is tailored for this challenge by proposing subtask decomposition to explicitly control the size of FOS, thereby extending the capability boundary of instruction-guided scene editing.

3 ProEdit: Methodology

The key insight of our ProEdit is to decompose a full editing task, described by a text instruction, into a sequence of simpler subtasks with smaller feasible output space (FOS), and apply each of them progressively on the scene. Our framework consists of three major components: (1) an interpolation-based subtask formulation that defines, obtains, and interprets each subtask; (2) a difficulty-aware subtask decomposition scheduler that breaks down the full editing task into several subtasks of comparable difficulty; and (3) an adaptive 3D Gaussian splatting (3DGS)-based [2] geometry-precise scene editing method that ensures high-quality editing for each subtask, ultimately leading to successful completion of the full task. Our framework is visualized in Fig. 2.

3.1 Interpolation-Based Subtask Formulation

In order to decompose a text-described task into subtasks, we first need to clearly define "task" and "subtasks." We define an editing task $T(s, e = E(p))$ as an operation that applies a prompt (instruction) p on the original scene s, where $e = E(p)$ denotes the text encoding of p calculated by a frozen text encoder $E(\cdot)$ as part of a 2D diffusion model. The notation $T(s, e)$ represents the edited

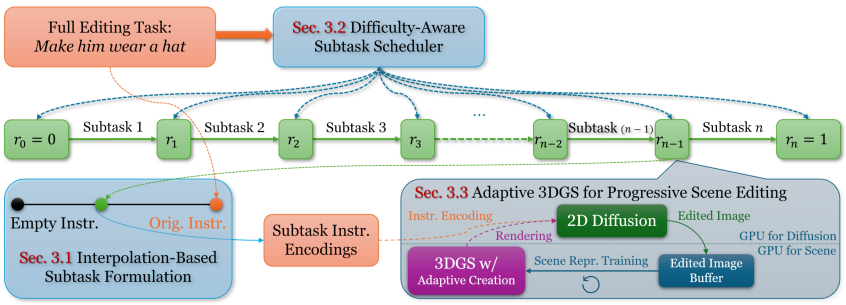

Figure 2: **Our ProEdit framework** features three major designs: an interpolation-based subtask formulation (Sec. 3.1), a difficulty-aware subtask scheduler for subtask decomposition (Sec. 3.2), and an adaptive 3DGS tailored for progressive scene editing through a dual-GPU pipeline (Sec. 3.3). For an editing task, we first decompose it into interpolation-based subtasks to schedule the editing process with the subtask scheduler, and then progressively perform the subtasks with adaptive 3DGS.

scene resulting from this task, and we also use $T(\cdot, e)$ to indicate the mapping from the original scene to the edited scene within this context. Additionally, we define $\varnothing$ as the empty prompt, indicating that the editing task with this prompt retains the original scene, or $T(s, E(\varnothing)) = s$.

Next, we define subtasks as $S(s, r) = T(s, e(r))$ with a ratio $r \in [0, 1]$, where $e(r) = r \cdot E(p) + (1 - r) \cdot E(\varnothing)$. This represents a task characterized by an instruction $p(r) = E^{-1}(r \cdot E(p) + (1-r) \cdot E(\varnothing))$, whose embedding is a ratio-r interpolation between $E(p)$ and $E(\varnothing)$. Assuming that the neural network $E(\cdot)$ is continuous, $S(s, r)$ will also be continuous w.r.t. r. Therefore, this formulation provides a continuous space of subtasks or intermediate tasks between the original task $T(\cdot, E(p))$ and the identity mapping $T(\cdot, E(\varnothing))$.

3.2 Difficulty-Aware Subtask Scheduler

Feasible Output Space (FOS) and Task Difficulty. Inspired by the derivation of SDS [42], we introduce the concept of *feasible output space* (FOS) for an editing task $T(s, E(p))$ as follows: the set of scenes s' such that, when s' is rendered from any view v, the resulting image resembles the edited image (based on instruction p) from the corresponding view v of the original scene s, *i.e.*, the set of all possible scenes that can be regarded as valid edited result for the given task. A larger FOS indicates greater diversity in how the editing task can be executed; however, this variability can cause multi-view inconsistency, if different views are edited differently. Therefore, an editing task with a larger FOS is inherently more difficult to accomplish.

Formulation of Subtask Decomposition. Our goal is to decompose the original editing task $T(\cdot, E(p))$ into a sequence of subtasks, such that applying each subtask progressively or iteratively on the current scene leads to the final editing result. Formally, the decomposition of a task $T(\cdot, E(p))$ is a monotonically increasing sequence $r_0, r_1, \cdots, r_n$, where $r_0 = 0, r_n = 1$. We then define s_i as the edited scene resulting from the i-th subtask. We have

$$s_i = \begin{cases} s, & \text{(original scene)}, \quad i = 0, \\ S(s_{i-1}, r_i), & \text{(apply subtask } r_i \text{ on previously edited scene)}, \quad i = 1, \cdots, n. \end{cases} \tag{1}$$

In other words, the i-th subtask is $S(s_{i-1}, r_i)$, which is applied on the edited scene s_{i-1} from the previous $(i-1)$-th subtask. The outcome of the i-th subtask is s_i.

Subtask Difficulty Measurement and Approximation. The difficulty of each subtask $S(s_{i-1}, r_i)$ is measured as being proportional to the size of FOS (a continuous space), which is difficult to compute or even rigorously define. Therefore, we approximate this difficulty by evaluating the difference between the original and edited images of the 2D diffusion model. Intuitively, an editing task that brings a significant change typically has more degrees of freedom, leading to a larger FOS. Additionally, each subtask r_i is applied on the scene s_{i-1}, which cannot be determined until all prior subtasks $r_1, r_2, \cdots, r_{i-1}$ are completed. So, we make another approximation based on the assumption that the image of a view in s_i will closely resemble the corresponding view of s edited by the 2D diffusion model following the instruction of the i-th subtask. In other words,

$$v_k(s_i) \approx T_{2D}(v_k(s), e(r_i)), \forall k \in V, \tag{2}$$

where $v_k(s)$ is the rendered image at the k-th view of scene s, and $T_{2D}(v, e)$ is the output of a 2D editing task applied on image v with instruction embedding e, generated by the 2D diffusion model. By applying such an approximation to both subtasks and using Learned Perceptual Image Patch Similarity (LPIPS) to measure the perceptual difference between images, we can then define the difficulty metric as

$$\mathrm{d}(r_i, r_j) \overset{\text{Def}}{=\!=} \sum_{k \in V} L_{\text{LPIPS}}(v_k(s_i), v_k(s_j)) \approx \sum_{k \in V} L_{\text{LPIPS}}(T_{2D}(v_k(s), e(r_i)), T_{2D}(v_k(s), e(r_j))). \tag{3}$$

Observing that $\mathrm{d}(r_i, r_j)$'s approximation is only related to the rendered image $v_k(s)$ of the original scene s and is independent of that of the edited scene (namely, $v_k(s_i)$), we can then allow $\mathrm{d}(r_a, r_b)$ to take any two arbitrary subtasks r_a and r_b. Our goal is to find the subtask decomposition $r_0, \cdots, r_n$ with similar $\{\mathrm{d}(r_{i-1}, r_i)\}$ for each i.

Difficulty-Aware Adaptive Subtask Decomposition. The approximation of $\mathrm{d}(r_i, r_j)$ disentangles its computation from the edited scene of task $T(\cdot, e(r_i))$, by substituting it with $T_{2D}(\cdot, e(r_i))$. This enables us to decompose the subtasks from a more *global* perspective. Therefore, we propose an adaptive method to obtain the set of subtask ratios $R = r_0, \cdots, r_n$. The algorithm operates recursively over an interval $[r_a, r_b]$ with a difficulty threshold $\mathrm{d}_{\text{threshold}}$, starting with the interval $[0, 1]$. In each recursion, the algorithm first includes both r_a and r_b in the set R, and stops the recursion if $\mathrm{d}(r_a, r_b) \leq \mathrm{d}_{\text{threshold}}$. Otherwise, it selects the middle point $r_m = (r_a + r_b)/2$, and recurses on the intervals $[r_a, r_m]$ and $[r_m, r_b]$. Once the recursion is complete, we obtain the sequence of subtasks $r_0, \cdots, r_n$ by sorting the set R, ensuring that $\mathrm{d}(r_{i-1}, r_i) \leq \mathrm{d}_{\text{threshold}}$ for all subtasks.

To simplify the subtask decomposition, we check if there exists a subtask r_i such that $\mathrm{d}(r_{i-1}, r_{i+1}) \leq \mathrm{d}_{\text{threshold}}$. If so, we could safely remove the subtask r_i while still maintaining $\mathrm{d}(r_{i-1}, r_{i+1}) \leq \mathrm{d}_{\text{threshold}}$. This iterative check continues until no further subtasks can be pruned.

Notably, an interpolated subtask can be regarded as a partial completion of the editing instruction. For example, the instruction "Make him smile" with an interpolation ratio of $r = 0.5$ can be interpreted as "Make him half-smile," indicating a lower *aggressivity* of the editing operation. In this context, high aggressivity indicates more significant changes towards the editing operation, whereas low aggressivity reflects greater similarity between the edited scene and the original one. Therefore, our subtask decomposition not only lays the foundation for our editing process but also categorizes task aggressivity, where each subtask corresponds to a specific level of aggressivity. Consequently, beyond performing editing, our ProEdit enables users to control, preview, and select the aggressivity of the editing operation *during or after the editing process*, by utilizing the edited scene of a subtask throughout the progressive editing workflow. Such a capability is absent in previous work.

Subtask Scheduling. The subtask scheduler also determines when the current subtask is complete, allowing us to proceed to the next one. Designing an image-based criterion to assess whether the images in the current subtask have been sufficiently edited is challenging. Therefore, we propose a criterion based on the scene representation training procedure. Specifically, when the running mean of the training loss no longer decreases over a specified number of iterations, we regard the scene representation to be converging to the edited scene, indicating that the current editing subtask is complete. Moreover, apart from the subtasks $r_0, r_1, \cdots, r_n$, we prepend an additional subtask r_0 to refine the initial scene representation using diffusion-reconstructed original images, and append another subtask r_n to consolidate the editing results, as detailed in Appendix B.

3.3 Adaptive 3DGS Tailored for Progression

We choose 3DGS [2] as our scene representation for its high efficiency and rendering quality. However, 3DGS is primarily designed for reconstruction from multi-view consistent images. Directly training on edited images with 3DGS results in a continuously increasing number of Gaussians that overfit the inconsistent views, ending up with an out-of-memory error. Therefore, we propose a novel Adaptive 3DGS specifically tailored for progressive scene editing.

Basic Workflow for Each Subtask. As each subtask has a reduced FOS and lower difficulty, we can use a straightforward approach to perform the subtask editing. Consistent with Instruct-NeRF2NeRF (IN2N) [5], we apply a simple iterative dataset update (Iterative DU) that iteratively generates edited views using the diffusion model and employs them to train the scene representation. Unlike NeRFs [1], our 3DGS-based scene representation accepts full images as supervision instead of rays, allowing

us to directly train on the edited images without the need to replace rays. This enables a simpler yet more effective workflow.

Adaptive Gaussian Creation Strategy. While the decomposition of subtasks controls the size of FOS and reduces potential inconsistencies, making 3DGS converge on the edited multi-view images remains challenging. Designed only for reconstruction from multi-view consistent images, 3DGS is not robust enough to deal with all inconsistencies. This can lead to overfitting on the inconsistent edited images with view-dependent colors, floating or floc noises, and multi-face structures.

Therefore, we propose an adaptive Gaussian creation strategy to refine the geometry of 3DGS, enabling it to converge on the edited images with reasonable and potentially high-quality geometric structures. As introduced in [2], the original 3DGS maintains Gaussian-represented geometry by periodically culling unnecessary Gaussians based on an opacity threshold, and by creating new Guassians (through splitting or duplicating) to expand model capability according to a training gradient threshold. Our strategy builds on this geometry maintenance schedule by *adaptively* controlling both thresholds. (1) At the beginning of each subtask, we set the opacity of all Gaussians to the threshold and perform several iterations of training without geometry maintenance. This training procedure implicitly identifies the Gaussians that correctly lie on the object surface by making them learn higher opacity, which allows them to be preserved in the scene representation. Conversely, Gaussians with incorrect geometry learn lower opacity and are subsequently culled during the next maintenance phase. (2) To prevent the training process from creating too many noisy Gaussians in a single iteration when operating with edited images, we also control the gradient threshold for Gaussian creation to achieve a smooth increase in the number of Gaussians. We schedule the number of created Gaussians based on the existing Gaussians in the scene and the number previously culled, selecting the threshold according to this scheduled number, as detailed in Appendix D. With these strategies, our 3DGS is able to converge to the edited scenes with clear texture and reasonable, even precise geometry.

Dual-GPU Training to Decouple Diffusion and 3DGS. Given the significant difference in iteration speeds – around 2 seconds per generation for the diffusion model inference and less than 0.02 seconds per iteration for the 3DGS training procedure – it is challenging to achieve an effective trade-off on a single GPU during Iterative DU. Inspired by [9, 43], we employ a dual-GPU training schedule to decouple them. The first GPU iteratively generates newly edited images using the diffusion model and stores them in a buffer as the updated dataset. Meanwhile, the second GPU iteratively trains 3DGS with the edited images in the buffer and raises a signal to indicate when the current subtask is complete. This approach enables a highly efficient training procedure within our ProEdit framework.

4 Experiments

4.1 Experimental Settings

Scene Representation and Diffusion Model. As mentioned in Sec. 3.3, our ProEdit leverages 3DGS-based scene representation for high quality and efficiency. We use the Splatfacto model from the NeRFStudio [44] library as our backbone. For the diffusion model, consistent with previous work [5, 6, 37, 45], we use a pre-trained Instruct-Pix2Pix (IP2P) [4] model from HuggingFace.

Scenes and Editing Instructions. According to Sec. 3.1, each editing task $T(s, E(p))$ is characterized by a scene s and an instruction p, and the desired output is the edited scene. We evaluate our ProEdit on the following scene datasets: (1) The IN2N dataset introduced by Instruct-NeRF2NeRF (IN2N) [5], which is available for free use and is the most widely used dataset in prior work. (2) The ScanNet++ dataset of indoor scenes, released under the ScanNet++ Terms of Use, which is introduced for instruction-guided scene editing in [9]. We use instructions either from previous methods for comparisons or from tasks that require highly noticeable geometric changes in the scene – one of the most challenging editing tasks that previous methods have struggled to perform well.

Subtask Scheduling. We determine the number of subtasks to balance editing quality, controllability, and efficiency. For texture-focused instructions (*e.g.*, style transfer), we decompose each task into approximately 4 subtasks using an appropriate threshold $\mathrm{d}_{\mathrm{threshold}}$; for geometry-related instructions with much higher FOS, we break each task down into around 8 subtasks with a proper $\mathrm{d}_{\mathrm{threshold}}$.

Baselines. We compare our ProEdit with recent state-of-the-art instruction-guided scene editing methods, including Instruct-NeRF2NeRF (IN2N) [5] (along with its 3DGS-based implementation

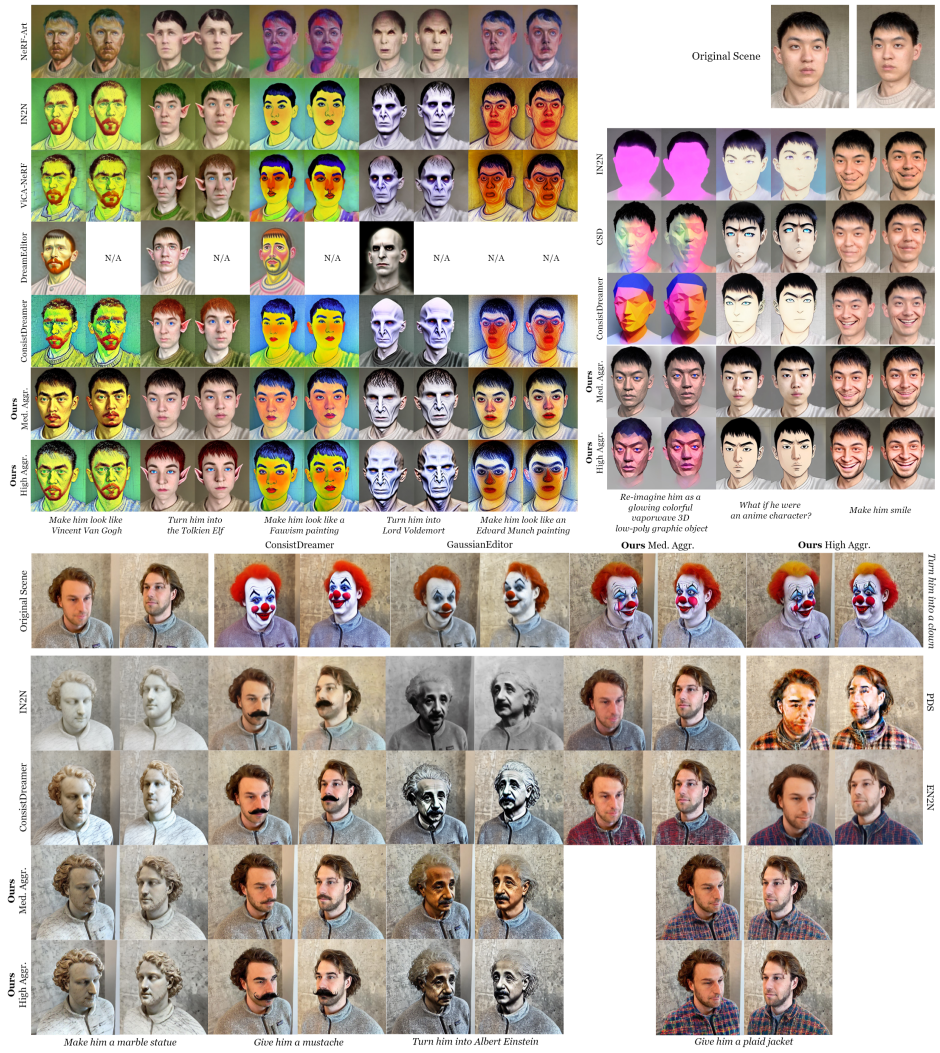

Figure 3: **In the comparative experiments on the Fangzhou and Face scenes**, our ProEdit achieves high-quality editing, with strong instruction fidelity, clear textures, and precise shapes across both levels of aggressivity controlled by subtask scheduling. The "medium aggressivity" editing results are obtained from an intermediate subtask. The editing results of the baselines are sourced from visualizations in their respective papers.

[45]), ViCA-NeRF [41], ConsistDreamer [9], CSD [6], PDS [7], Efficient-NeRF2NeRF (EN2N) [37], DreamEditor [46], *etc*. As different methods use different tasks for visualization in their papers, and some do not provide publicly available code or pre-trained models, our primary comparisons focus on common editing tasks, leveraging the visualizations presented in their papers. Also, we include comparisons for some additional tasks with results generated from available code or re-implementations. As our ProEdit specifically targets the instruction-guided scene editing task, we do not include comparisons with methods designed for other scene editing or generation tasks.

Implementation Details. We follow the default hyperparameter settings of the Splatfacto method, and set the classifier-free guidance (CFG) [10] as 7.5×1.5 for all instructions in the diffusion model. During the editing process for each subtask, consistent with IN2N [5], we use SDEdit's [11] method to control similarity with denoising timesteps between 450 and 850. We also apply HiFA's [47] annealing strategy to gradually decrease denoising timesteps in this process. Utilizing a dual-GPU training workflow (Sec. 3.3), the editing tasks are conducted on two NVIDIA A6000 or A100 GPUs, with each subtask taking 10 to 20 minutes to complete depending on its difficulty and convergence.

Metrics. We present the quantitative assessment under the following metrics: User Study of Overall Quality (USO), User Study of 3D Consistency (US3D), GPT Evaluation Score (GPT), CLIP [48]

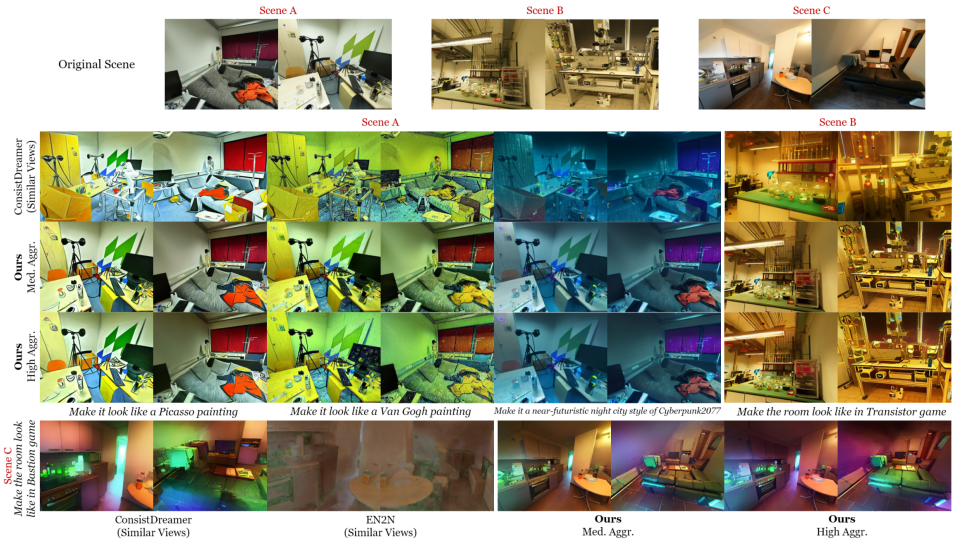

Figure 4: **In the comparative experiments on the ScanNet++ scenes**, our simple ProEdit also achieves high-quality editing that is comparable to, and in some cases even outperforms, the sophisticated baseline ConsistDreamer [9]. All visualizations are sourced from ConsistDreamer's paper.

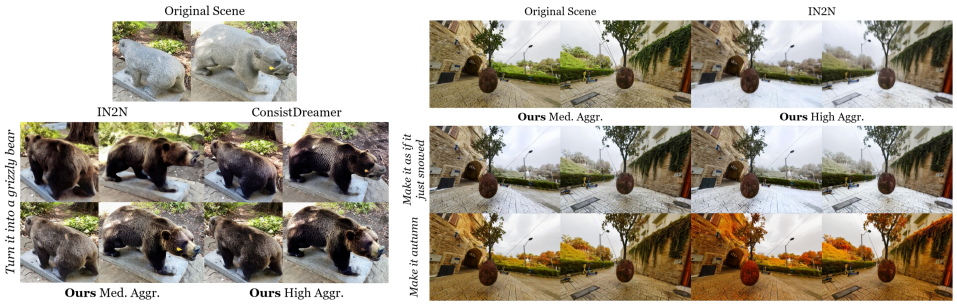

Figure 5: **In the comparative experiments across various outdoor scenes**, our ProEdit not only achieves high-quality editing that surpasses the baselines, but also enables aggressivity controls for a range of scenes and tasks.

Text-Image Direction Similarity (CTIDS), and CLIP Direction Consistency (CDC). The user study was conducted with 26 participants. The GPT score is detailed in Appendix E. The CLIP-based scores are consistent with those reported in IN2N [5].

4.2 Experimental Results and Analysis

Qualitative Results. Fig. 3 shows the comparisons in the Fangzhou scene and the IN2N's Face scene. Our ProEdit demonstrates results on two levels of editing aggressivity: high aggressivity results are obtained by executing all subtasks, while medium aggressivity results are derived from completing only the first 40% subtasks. Overall, our ProEdit produces high-quality editing results characterized by clear textures, bright colors, reasonable and precise geometry, and high instruction fidelity. Compared with the baselines, our ProEdit shows enhanced geometry editing capabilities, particularly in the "Tolkien Elf" editing which features a thinner face, and the "Lord Voldemort" editing which incorporates more wrinkles in the Fangzhou scene. By contrast, the baselines tend to maintain geometry more similar to the original scene. Notably, for the editing task "Give him a plaid jacket," our ProEdit generates much clearer and more noticeable plaid patterns than all baselines.

The experimental results on the ScanNet++ dataset are shown in Fig. 4. With subtask decomposition and progressive editing, our ProEdit achieves high-quality results that are comparable to and even outperform the baseline ConsistDreamer [9], which incorporates three complicated add-ons for ensuring 3D consistency. This shows that our simple progression is more effective in reducing inconsistency – through reducing the size of FOS – than explicit 3D consistency-enforcing components.

Method	USO↑	US3D↑	GPT↑	CTIDS↑	CDC↑	Running Time↓
IN2N [5]	51.35	65.45	45.32	0.0773	0.3260	**0.5-1h**
ConsistDreamer [9]	68.65	75.23	74.40	**0.0912**	**0.3912**	12-24h
ProEdit (Ours)	**87.96**	**80.23**	**81.00**	0.0844	0.3833	1-4h

Table 1: Our ProEdit significantly outperforms baselines in USO, US3D, and GPT metrics, and achieves comparable CLIP metrics to sophisticated ConsistDreamer with only 1/3 of its running time.

Method	USO↑	US3D↑	USP↑	GPT↑	CTIDS↑	CDC↑
ProEdit (ND)	68.46	61.72	60.73	72.87	0.0671	0.2902
ProEdit (Full)	**92.70**	**90.48**	**88.72**	**82.80**	**0.0844**	**0.3833**

Table 2: **Ablation study of our "no subtask decomposition (ND)" variant** shows that our full ProEdit significantly outperforms the "ND" variant across all metrics, validating that progression is crucial to achieving high-quality editing results.

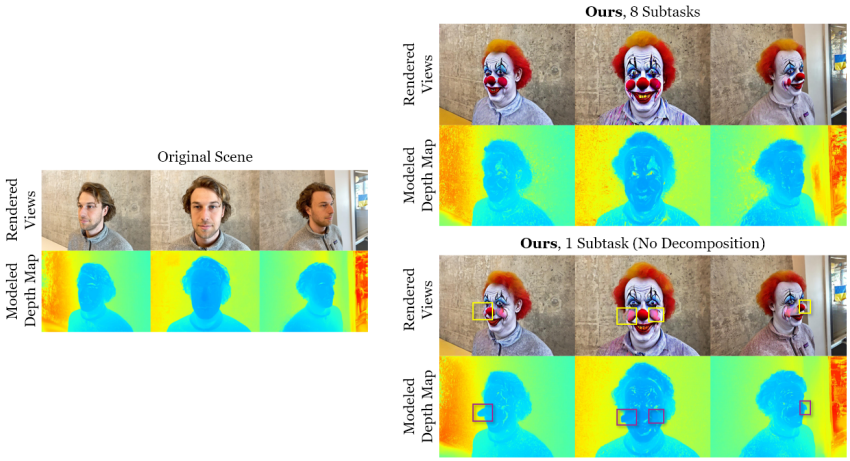

Figure 6: **Ablation study of our "no subtask decomposition" variant** shows that removing subtask decomposition results in unreasonable geometry, particularly near the cheek area (indicated by the bounding boxes). This validates the importance of subtask decomposition in achieving high-quality editing in our framework. "Modeled depth map" is the depths modeled by the scene representation.

We also conduct comparison experiments on two outdoor scenes: "Bear" from IN2N [5] and "Floating Tree" from NeRFStudio [44], as shown in Fig. 5. In the "grizzly bear" task, our ProEdit generates similar fur textures as ConsistDreamer, both of which are much clearer than IN2N, with the added advantage of aggressivity control in our model. Notably, our ProEdit achieves comparable editing quality at only 1/4 to 1/6 of ConsistDreamer's running time and with fewer GPUs. In the "snow" task, our ProEdit also delivers high-quality editing results, generating snow on the ground and making the sky whiter, while the baseline IN2N creates a blurred ground and leaves. In the "autumn" task, our ProEdit demonstrates its aggressivity control by adjusting the color intensity of the leaves. These results highlight the effectiveness of our approach for outdoor scenes as well.

In addition, our ProEdit shows the capability to control and categorize the aggressivity level of editing tasks. By selecting the edited scene from an intermediate subtask, we can obtain scenes with varying levels of aggressivity – namely, medium and high aggressivity, as shown in Figs. 3, 4, and 5 – with noticeable discrepancies. For example, in the medium-aggressivity version of the "Tolkien Elf" editing in Fig. 3, only the eye color and ear shape are modified, while in the high-aggressivity version, not only are the ears lengthened, but the hair is also colored red, and the face is thinned. These results underscore the unique strength of our ProEdit in controlling editing aggressivity.

Additional qualitative results are shown on our project page.

Quantitative Results. Table 1 presents quantitative comparisons. ProEdit consistently outperforms IN2N by a large margin. It also significantly surpasses the strong baseline ConsistDreamer in two overall quality metrics and the user study-based 3D consistency metric, while achieving comparable performance on CLIP-based metrics – all with only 1/3 of ConsistDreamer's running time.

Ablation Study. To validate the necessity of our subtask decomposition, we conduct experiments on a variant of ProEdit using only one subtask ($n = 1$, $r_0 = 0$, $r_1 = 1$), effectively disabling decomposition (referred to as "ND"). Qualitative results are shown in Fig. 6. Without subtask decomposition, the variant generates unrealistically long cheeks to accommodate inconsistencies in cheek decorations across views, resulting in blurred cheek textures in the rendered output due to the large FOS of the editing task. In contrast, our full ProEdit achieves bright, clear results with precise and realistic geometry. Quantitative results are shown in Table 2. For this comparison, we conducted a new user study involving 41 participants, including an additional User Study of Shape Plausibility ("USP") metric: we provide participants with the modeled depth maps, similar to those in Fig. 6, along with the rendered RGB images. We then ask them to evaluate whether the shapes are realistic and match the rendered images. The "ND" variant performs significantly worse than our full method on all user study metrics, further underscoring the effectiveness of our subtask decomposition. These results collectively demonstrate that reducing FOS through subtask decomposition is crucial to our high-quality results.

5 Discussion

3D Consistency Add-Ons. Different from our subtask decomposition strategy, 3D consistency add-ons, such as distillation losses, consistency-inducing components, and specific training procedures, offer an alternative way to control and reduce FOS. Although our framework achieves high-quality editing without them, combining it with these 3D consistency add-ons can leverage the strengths of both approaches, potentially reducing the number of required subtasks and enhancing editing quality.

Limitations. Our ProEdit is a distillation-guided framework from 2D diffusion, similar to all baselines. Therefore, its editing capability is constrained by the underlying diffusion model. If the diffusion model does not support applying a specific editing instruction on most views of a scene, our ProEdit will also be unable to do so. Additionally, ProEdit relies on 3DGS for efficient training, which NeRF-based representations do not support; consequently, it inherits certain limitations of 3DGS, including limited suitability for unbounded outdoor scenes. Finally, ProEdit may still encounter the multi-face or Janus problems, as the 2D diffusion model lacks 3D awareness.

Future Directions. There are many promising directions to explore in subtask decomposition beyond the interpolation-based strategy introduced in this paper. One potential way is to explicitly construct intermediate subtasks using semantic guidance. For example, applying "Turn him into a bald person" before "Make him wear a hat" could lead to a more free-form hat independent of the hair, with such intermediate instructions generated by large language models. Another alternative avenue involves leveraging video generation models to "animate" the transition from the original scene to the edited scene, treating this animation process as a series of subtasks. Doing so will enable ProEdit to function as a 3D scene animator, generating high-quality 4D (dynamic 3D) scenes. Additionally, the progressive framework of ProEdit can be potentially applied to scene generation.

Potential Societal Impacts. The positive societal impacts of our ProEdit include (1) the development of consumer-grade 3D scene editing products and applications, facilitated by advancements in 3D structured-light scanners for mobile phones and virtual reality (VR) and augmented reality (AR); and (2) the transformation of high-quality 3D and 4D (dynamic 3D) scene creation through the editing of existing high-resolution scenes. On the other hand, as our framework is based on generative models, it is crucial to address potential ethical and safety concerns, including risks of producing biased results and the possibility of misuse for illegal activities.

6 Conclusion

This paper proposes ProEdit, a novel 3D scene editing framework that decomposes the editing task into subtasks and performs them progressively. Our method targets the fundamental cause of inconsistency – the large feasible output space of the diffusion model with respect to an editing task. Extensive experiments show that our ProEdit produces high-quality editing results characterized by bright colors, sharp and detailed textures, and precise geometric structures across various scenes and editing tasks. Our method further enables a novel controllability over the aggressivity of the editing task, by allowing users to select which subtasks to execute. We hope that our ProEdit will inspire exciting applications and new research directions in 3D scene editing and generation.

Acknowledgments

This work was supported in part by NSF Grant 2106825, NIFA Award 2020-67021-32799, the IBM-Illinois Discovery Accelerator Institute, the Toyota Research Institute, and the Jump ARCHES endowment through the Health Care Engineering Systems Center at Illinois and the OSF Foundation. This work used computational resources on NCSA Delta through allocations CIS220014 and CIS230012 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, and on TACC Frontera through the National Artificial Intelligence Research Resource (NAIRR) Pilot.

References

[1] 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, 3, 5, 14

[2] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *TOG*, 42(4):139–1, 2023. 1, 2, 3, 5, 6, 14

[3] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 3

[4] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Learning to follow image editing instructions. In *CVPR*, 2023. 2, 6, 14, 16

[5] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D scenes with instructions. In *ICCV*, 2023. 2, 3, 5, 6, 7, 8, 9

[6] Subin Kim, Kyungmin Lee, June Suk Choi, Jongheon Jeong, Kihyuk Sohn2, and Jinwoo Shin1. Collaborative score distillation for consistent visual editing. In *NeurIPS*, 2023. 2, 3, 6, 7

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

[8] Jiayu Yang, Ziang Cheng, Yunfei Duan, Pan Ji, and Hongdong Li. ConsistNet: Enforcing 3D consistency for multi-view images diffusion. In *CVPR*, 2024. 2

[9] Jun-Kun Chen, Samuel Rota Bulò, Norman Müller, Lorenzo Porzi, Peter Kontschieder, and Yu-Xiong Wang. ConsistDreamer: 3D-consistent 2D diffusion for high-fidelity scene editing. In *CVPR*, 2024. 2, 3, 6, 7, 8, 9

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

[11] 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. 2, 7

[12] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, 2019. 3

[13] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, 1996.

[14] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 3

[15] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *ECCV*, 2018. 3

[16] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. SurfaceNet: An end-to-end 3D neural network for multiview stereopsis. In *ICCV*, 2017.

[17] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *TPAMI*, 38(10):2024–2039, 2015. 3

[18] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVS-NeRF: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021. 3

[19] Liwen Wu, Jae Yong Lee, Anand Bhattad, Yu-Xiong Wang, and David Forsyth. DIVeR: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In *CVPR*, 2022.

[20] Jun-Kun Chen, Jipeng Lyu, and Yu-Xiong Wang. NeuralEditor: Editing neural radiance fields via manipulating point clouds. In *CVPR*, 2023. 3

[21] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *NeurIPS*, 2020.

[22] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

[23] 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.

[24] Yuan-Chen Guo, Di Kang, Linchao Bao, Yu He, and Song-Hai Zhang. NeRFReN: Neural radiance fields with reflections. In *CVPR*, 2022. 3

[25] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. DN-Splatter: Depth and normal priors for Gaussian splatting and meshing, 2024. 3

[26] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH*, 2024. 3

[27] Xiaoyang Lyu, Yang-Tian Sun, Yi-Hua Huang, Xiuzhe Wu, Ziyi Yang, Yilun Chen, Jiangmiao Pang, and Xiaojuan Qi. 3DGSR: Implicit surface reconstruction with 3D Gaussian splatting, 2024. 3

[28] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian splatting for real-time dynamic scene rendering. In *CVPR*, 2024. 3

[29] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4D reconstruction with dynamic 3D Gaussian particle. In *CVPR*, 2024.

[30] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, 2024. 3

[31] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Junyan Zhu, and Bryan C. Russell. Editing conditional radiance fields. In *ICCV*, 2021. 3

[32] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *ICCV*, 2021.

[33] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing NeRF for editing via feature field distillation. In *NeurIPS*, 2022. 3

[34] Yi-Ling Qiao, Alexander Gao, and Ming C. Lin. NeuPhysics: Editable neural geometry and physics from monocular videos. In *NeurIPS*, 2022. 3

[35] Viktor Rudnev, Mohamed Elgharib, William Smith, Lingjie Liu, Vladislav Golyanik, and Christian Theobalt. NeRF for outdoor scene relighting. In *ECCV*, 2022. 3

[36] Yingyan Xu, Gaspard Zoss, Prashanth Chandran, Markus Gross, Derek Bradley, and Paulo Gotardo. ReNeRF: Relightable neural radiance fields with nearfield lighting. In *ICCV*, 2023. 3

[37] Liangchen Song, Liangliang Cao, Jiatao Gu, Yifan Jiang, Junsong Yuan, and Hao Tang. Efficient-NeRF2NeRF: Streamlining text-driven 3D editing with multiview correspondence-enhanced diffusion models, 2023. 3, 6, 7

[38] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. GaussianEditor: Swift and controllable 3D editing with Gaussian splatting. In *CVPR*, 2024. 3

[39] Hiromichi Kamata, Yuiko Sakuma, Akio Hayakawa, Masato Ishii, and Takuya Narihira. Instruct 3D-to-3D: Text instruction guided 3D-to-3D conversion. *arXiv preprint arXiv:2303.15780*, 2023. 3

[40] Lu Yu, Wei Xiang, and Kang Han. Edit-DiffNeRF: Editing 3D neural radiance fields using 2D diffusion model. *arXiv preprint arXiv:2306.09551*, 2023.

[41] Jiahua Dong and Yu-Xiong Wang. ViCA-NeRF: View-consistency-aware 3D editing of neural radiance fields. In *NeurIPS*, 2023. 3, 7

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

[43] Linzhan Mou, Jun-Kun Chen, and Yu-Xiong Wang. Instruct 4D-to-4D: Editing 4D scenes as pseudo-3D scenes using 2D diffusion. In *CVPR*, 2024. 6

[44] 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 *SIGGRAPH*, 2023. 6, 9

[45] Cyrus Vachha and Ayaan Haque. Instruct-GS2GS: Editing 3D Gaussian splats with instructions, 2024. 6, 7

[46] Jingyu Zhuang, Chen Wang, Lingjie Liu, Liang Lin, and Guanbin Li. DreamEditor: Text-driven 3D scene editing with neural fields. In *SIGGRAPH Asia*, 2023. 7

[47] Joseph Zhu and Peiye Zhuang. HiFA: High-fidelity text-to-3D with advanced diffusion guidance. In *ICLR*, 2024. 7

[48] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 7, 16

[49] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 15

[50] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *ECCV*, 2024. 16