3DEgo: 3D Editing on the Go!

Umar Khalid[1,*], Hasan Iqbal[2,*], Azib Farooq[3], Jing Hua[2], and Chen Chen[1]

[1] University of Central Florida, Orlando, FL, USA
[2] Department of Computer Science, Wayne State University, Detroit, MI, USA
[3] Department of Computer Science and Software Engineering, Miami University, Oxford, OH, USA

Abstract. We introduce **3DEgo** to address a novel problem of directly synthesizing photorealistic 3D scenes from monocular videos guided by textual prompts. Conventional methods construct a text-conditioned 3D scene through a three-stage process, involving pose estimation using Structure-from-Motion (SfM) libraries like COLMAP, initializing the 3D model with unedited images, and iteratively updating the dataset with edited images to achieve a 3D scene with text fidelity. Our framework streamlines the conventional multi-stage 3D editing process into a single-stage workflow by overcoming the reliance on COLMAP and eliminating the cost of model initialization. We apply a diffusion model to edit video frames prior to 3D scene creation by incorporating our designed *noise blender module* for enhancing multi-view editing consistency, a step that does not require additional training or fine-tuning of T2I diffusion models. **3DEgo** utilizes 3D Gaussian Splatting to create 3D scenes from the multi-view consistent edited frames, capitalizing on the inherent temporal continuity and explicit point cloud data. 3DEgo demonstrates remarkable editing precision, speed, and adaptability across a variety of video sources, as validated by extensive evaluations on six datasets, including our own prepared GS25 dataset. Project Page: https://3dego.github.io/

Keywords: Gaussian Splatting · 3D Edititng · Cross-View Consistency

1 Introduction

In the pursuit of constructing photo-realistic 3D scenes from monocular video sources, it is a common practice to use the Structure-from-Motion (SfM) library, COLMAP [40] for camera pose estimation. This step is critical for aligning frames extracted from the video, thereby facilitating the subsequent process of 3D scene reconstruction. To further edit these constructed 3D scenes, a meticulous process of frame-by-frame editing based on textual prompts is often employed [52]. Recent works, such as IN2N [11], estimate poses from frames using SfM [40] to initially train an unedited 3D scene. Upon initializing a 3D model, the training dataset is iteratively updated by adding edited images at a consistent rate

* Equal Contribution

of editing. This process of iterative dataset update demands significant computational resources and time. Due to challenges with initial edit consistency, IN2N [11] training necessitates the continuous addition of edited images to the dataset over a significantly large number of iterations. This issue stems from the inherent limitations present in Text-to-Image (T2I) diffusion models [4,37], where achieving prompt-consistent edits across multiple images—especially those capturing the same scene—proves to be a formidable task [7,19]. Such inconsistencies significantly undermine the effectiveness of 3D scene modifications, particularly when these altered frames are leveraged to generate unique views.

In this work, we address a *novel problem* of efficiently reconstructing 3D scenes directly from monocular videos without using COLMAP [40] aligned with the editing textual prompt. Specifically, we apply a diffusion model [4] to edit every frame of a given monocular video before creating a 3D scene. To address the challenge of consistent editing across all the frames, we introduce a novel *noise blender module*, which ensures each new edited view is conditioned upon its adjacent, previously edited views. This is achieved by calculating a weighted average of image-conditional noise estimations such that closer frames exert greater influence on the editing outcome. Our editing strategy utilizes the IP2P [4] 2D editing diffusion model, which effectively employs both conditional and unconditional noise prediction. Consequently, our method achieves multi-view consistency without the necessity for extra training or fine-tuning, unlike prior approaches [7,27,46]. For 3D scene synthesis based on the edited views, our framework utilizes the Gaussian Splatting (GS) [17] technique, capitalizing on the temporal continuity of video data and the explicit representation of point clouds. Originally designed to work with pre-computed camera poses, 3D Gaussian Splatting presents us with the possibility to synthesize views and construct edited 3D scenes from monocular videos without the need for SfM pre-processing, overcoming one of NeRF's significant limitations [25].

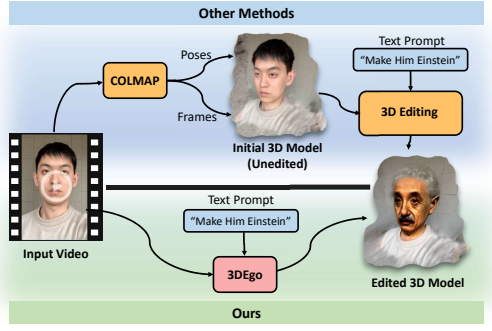

Fig. 1: Our method, **3DEgo**, streamlines the 3D editing process by merging a three-stage workflow into a singular, comprehensive framework. This efficiency is achieved by bypassing the need for COLMAP [40] for pose initialization and avoiding the initialization of the model with unedited images, unlike other existing approaches [7,11,19].

Our method grows the 3D Gaussians of the scene continuously, from the edited frames, as the camera moves, eliminating the need for pre-computed camera poses and 3D model initialization on original un-edited frames to identify

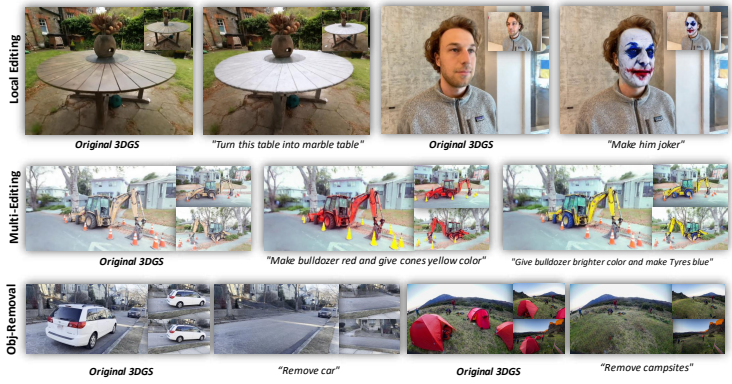

Fig. 2: 3DEgo offers rapid, accurate, and adaptable 3D editing, bypassing the need for original 3D scene initialization and COLMAP poses. This ensures compatibility with videos from any source, including casual smartphone captures like the **Van** 360-degree scene. The above results identify three cases challenging for IN2N [11], where our method can convert a monocular video into customized 3D scenes using a streamlined, single-stage reconstruction process.

an affine transformation that maps the 3D Gaussians from frame i to accurately render the pixels in frame $i + 1$. Hence, our method **3DEgo** condenses a three-stage 3D editing process into a single-stage, unified and efficient framework as shown in Figure 1. Our contributions are as follows:

- We tackle the novel challenge of directly transforming monocular videos into 3D scenes guided by editing text prompts, circumventing conventional 3D editing pipelines.
- We introduce a unique auto-regressive editing technique that enhances multi-view consistency across edited views, seamlessly integrating with pre-trained diffusion models without the need for additional fine-tuning.
- We propose a COLMAP-free method using 3D Gaussian splatting for reconstructing 3D scenes from casually captured videos. This technique leverages the video's continuous time sequence for pose estimation and scene development, bypassing traditional SfM dependencies.
- We present an advanced technique for converting 2D masks into 3D space, enhancing editing accuracy through Pyramidal Gaussian Scoring (PGS), ensuring more stable and detailed refinement.
- Through extensive evaluations on six datasets—including our custom **GS25** and others like IN2N, Mip-NeRF, NeRFstudio Dataset, Tanks & Temples, and CO3D-V2—we demonstrate our method's enhanced editing precision and efficiency, particularly with *360-degree* and *casually recorded* videos, as illustrated in Fig. 2.

2 Related Work

A growing body of research is exploring diffusion models for text-driven image editing, introducing techniques that allow for precise modifications based on user-provided instructions [30,35,37,39]. While some approaches require explicit before-and-after captions [12] or specialized training [38], making them less accessible to non-experts, IP2P [4] simplifies the process by enabling direct textual edits on images, making advanced editing tools more widely accessible.

Recently, diffusion models have also been employed for 3D editing, focusing on altering the geometry and appearance of 3D scenes [1, 4, 10, 13, 16, 18, 22–24, 26, 28, 31, 43, 44, 48, 49].

Traditional NeRF representations, however, pose significant challenges for precise editing due to their implicit nature, leading to difficulties in localizing edits within a scene. Earlier efforts have mainly achieved global transformations [6, 14, 29, 45, 47, 51], with object-centric editing remaining a challenge. IN2N [11] introduced user-friendly text-based editing, though it might affect the entire scene. Recent studies [7, 19, 52] have attempted to tackle local editing and multi-view consistency challenges within the IN2N framework [11]. Yet, no existing approaches in the literature offer pose-free capabilities, nor can they create a text-conditioned 3D scene from arbitrary video footage. Nevertheless, existing 3D editing methods [11, 52] universally necessitate Structure-from-Motion (SfM) preprocessing. Recent studies like Nope-NeRF [3], BARF [25], and SC-NeRF [15] have introduced methodologies for pose optimization and calibration concurrent with the training of *(unedited)* NeRF.

In this study, we present a novel method for constructing 3D scenes directly from textual prompts, utilizing monocular video frames without dependence on COLMAP poses [40], thus addressing unique challenges. Given the complexities NeRF's implicit nature introduces to simultaneous 3D reconstruction and camera registration, our approach leverages the advanced capabilities of 3D Gaussian Splatting (3DGS) [17] alongside a pre-trained 2D editing diffusion model for efficient 3D model creation.

3 Method

Given a sequence of unposed images alongside camera intrinsics, we aim to recover the camera poses in sync with the edited frames and reconstruct a photo-realistic 3D scene conditioned on the textual prompt.

3.1 Preliminaries

In the domain of 3D scene modeling, 3D Gaussian splatting [17] emerges as a notable method. The method's strength lies in its succinct Gaussian representation coupled with an effective differential rendering technique, facilitating real-time, high-fidelity visualization. This approach models a 3D environment

using a collection of point-based 3D Gaussians, denoted as $\mathcal{H}$ where each Gaussian $h = \{\mu, \Sigma, c, \alpha\}$. Here, $\mu \in \mathbb{R}^3$ specifies the Gaussian's center location, $\Sigma \in \mathbb{R}^{3\times3}$ is the covariance matrix capturing the Gaussian's shape, $c \in \mathbb{R}^3$ is the color vector in RGB format represented in the three degrees of spherical harmonics (SH) coefficients, and $\alpha \in \mathbb{R}$ denotes the Gaussian's opacity level. To optimize the parameters of 3D Gaussians to represent the scene, we need to render them into images in a differentiable manner. The rendering is achieved by approximating the projection of a 3D Gaussian along the depth dimension into pixel coordinates expressed as:

$$C = \sum_{p\in\mathcal{P}} c_p \tau_p \prod_{k=1}^{p-1} (1 - \alpha_k), \tag{1}$$

where $\mathcal{P}$ are ordered points overlapping the pixel, and $\tau_p = \alpha_p e^{-\frac{1}{2}(x_p)^T \Sigma^{-1}(x_p)}$ quantifies the Gaussian's contribution to a specific image pixel, with x_p measuring the distance from the pixel to the center of the p-th Gaussian. In the original 3DGS, initial Gaussian parameters are refined to fit the scene, guided by *ground truth poses* obtained using SfM. Through differential rendering, the Gaussians' parameters, including position μ, shape Σ, color c, and opacity α, are adjusted using a photometric loss function.

3.2 Multi-View Consistent 2D Editing

In the first step, we perform 2D editing with key editing areas (KEA) based on the user-provided video, V, and editing prompt, $\mathcal{T}$.

From the given video V, we extract frames $\{f_1, f_2, \ldots, f_N\}$. Analyzing the textual prompt $\mathcal{T}$ with a Large Language Model $\mathcal{L}$ identifies key editing attributes $\{A_1, A_2, \ldots, A_k\}$, essential for editing, expressed as $\mathcal{L}(\mathcal{T}) \rightarrow \{A_1, A_2, \ldots, A_k\}$. Utilizing these attributes, a segmentation model $\mathcal{S}$ delineates editing regions in each frame f_i by generating a mask M_i, with KEA marked as 1, and others as 0. The segmentation operation is defined as, $\mathcal{S}(f_i, \{A_1, A_2, \ldots, A_k\}) \rightarrow M_i, \quad \forall i \in \{1, \ldots, N\}$. Subsequently, a 2D diffusion model $\mathcal{E}$ selectively edits these regions in f_i, as defined by M_i, producing edited frames $\{E_1, E_2, \ldots, E_N\}$ under guidance from $\mathcal{T}$, such that $\mathcal{E}(f_i, M_i) \rightarrow E_i$.

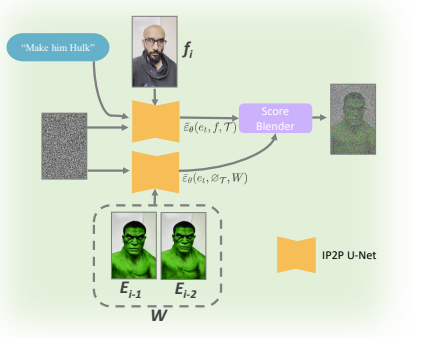

Fig. 3: Autoregressive Editing. At each denoising step, the model predicts $w + 1$ separate noises, which are then unified via weighted noise blender (Eq. 4) to predict $\varepsilon_\theta(e_t, f, \mathcal{T}, W)$.

Consistent Multi-View2D Editing. As discussed above, differing from IN2N [11] that incorporates edited images gradually over several training iterations, our approach involves editing the entire dataset at once before the training starts. We desire 1) each edited frame, E_i follows the editing prompt, $\mathcal{T}$, 2) retain the original images' semantic content, and 3) the edited images, $\{E_1, E_2, \ldots, E_N\}$ are consistent with each other.

***(i)* Multi-view Consistent Mask.** As $\mathcal{S}$ doesn't guarantee consistent masks across the views of a casually recorded monocular video, we utilize a zero-shot point tracker [34] to ensure uniform mask generation across the views. The procedure starts by identifying query points in the initial video frame using the ground truth mask. Query points are extracted from these ground truth masks employing the K-Medoids [32] sampling method. This method utilizes the cluster centers from K-Medoids clustering as query points. This approach guarantees comprehensive coverage of the object's various sections and enhances resilience to noise and outliers.

***(ii)*Autoregressive Editing.** To address the issue of preserving consistency across multiple views, we employ an autoregressive method that edits frames in sequence, with IP2P [4] editing restricted to the Key Editing Areas (KEA) as delineated by the relevant masks. Instead of editing each frame independently from just the input images - a process that can vary significantly between adjacent images - we integrate an *autoregressive* editing technique where the frame to be edited is conditioned on already edited adjacent frames.

As discussed above, we incorporate IP2P [4] as a 2D editing diffusion model. The standard noise prediction from IP2P's backbone that includes both conditional and unconditional editing is given as,

$$\tilde{\varepsilon}_\theta(e_t, f, \mathcal{T}) = \varepsilon_\theta(e_t, \varnothing_f, \varnothing_\mathcal{T}) + s_f\big(\varepsilon_\theta(e_t, f, \varnothing_\mathcal{T}) - \varepsilon_\theta(e_t, \varnothing_f, \varnothing_\mathcal{T})\big) + s_\mathcal{T}\big(\varepsilon_\theta(e_t, f, \mathcal{T}) - \varepsilon_\theta(e_t, f, \varnothing_\mathcal{T})\big) \quad (2)$$

where s_f and $s_\mathcal{T}$ are image and textual prompt guidance scale.We suggest enhancing the noise estimation process with our autoregressive training framework. Consider a set of w views, represented as $W = \{E_n\}_{n=1}^{w}$. Our goal is to model the distribution of the i-th view image by utilizing its w adjacent, already edited views. To achieve this, we calculate image-conditional noise estimation, $\varepsilon_\theta(e_t, E, \varnothing_\mathcal{T})$ across all frames in W. The equation to compute the weighted average $\bar{\varepsilon}_\theta$ of the noise estimates from all edited frames within W, employing β as the weight for each frame, is delineated as follows:

$$\bar{\varepsilon}_\theta(e_t, \varnothing_\mathcal{T}, W) = \sum_{n=1}^{w} \beta_n \varepsilon_\theta^n(e_t, E_n, \varnothing_\mathcal{T}) \quad (3)$$

Here, E_n represents the n-th edited frame within W, and β_n is the weight assigned to the n-th frame's noise estimate. The condition that the sum of all β values over w frames equals 1 is given by as, $\sum_{n=1}^{w} \beta_n = 1$. This ensures that the weighted average is normalized. As we perform 2D editing without any pose priors, our weight parameter β is independent of the angle offset between the frame to be edited, f_n and already edited frames in W. To assign weight parameters with exponential decay, ensuring the closest frame receives the highest

weight, we can use an exponential decay function for the weight β_n of the n-th frame in W. By employing a decay factor λ_d ($0 < \lambda_d < 1$), the weight of each frame decreases exponentially as its distance from the target frame increases. The weight β_n for the n-th frame is defined as, $\beta_n = \lambda_d^{w-n}$. This ensures the, E closest to the target, f ($n = 1$) receives the highest weight. To ensure the sum of the weights to 1, each weight is normalized by dividing by the sum of all weights, $\beta_n = \frac{\lambda^{w-n}}{\sum_{j=1}^{w} \lambda^{w-j}}$.This normalization guarantees the sum of β_n across all n equals 1, adhering to the constraint $\sum_{n=1}^{w} \beta_n = 1$.

Our editing path is determined by the sequence of frames from the captured video. Therefore, during the editing of frame f_n, we incorporate the previous w edited frames into the set W, assigning the highest weight β to E_{n-1}. Using Eq. 2 and Eq. 3, we define our score estimation function as following:

$$\varepsilon_\theta(e_t, f, \mathcal{T}, W) = \gamma_f \tilde{\varepsilon}_\theta(e_t, f, \mathcal{T}) + \gamma_E \bar{\varepsilon}_\theta(e_t, \varnothing_\mathcal{T}, W) \tag{4}$$

where γ_f is a hyperparameter that determines the influence of the original frame undergoing editing on the noise estimation, and γ_E represents the significance of the noise estimation from adjacent edited views.

3.3 3D Scene Reconstruction

After multi-view consistent 2D editing is achieved across all frames of the given video, V, we leverage the edited frames E_i and their corresponding masks M_i to construct a 3D scene without any SfM pose initialization. Due to the explicit nature of 3DGS [17], determining the camera poses is essentially equivalent to estimating the transformation of a collection of 3D Gaussian points. Next, we will begin by introducing an extra Gaussian parameter for precise local editing. Subsequently, we will explore relative pose estimation through incremental frame inclusion. Lastly, we will examine the scene expansion, alongside a discussion on the losses integrated into our global optimization strategy.

3D Gaussians Parameterization for Precise Editing. Projecting KEA (see Section 3.2) into 3D Gaussians, $\mathcal{H}$, using M for KEA identity assignment, is essential for accurate editing. Therefore, we introduce a vector, m associated with the Gaussian point, $h = \{\mu, \Sigma, c, \alpha, m\}$ in the 3D Gaussian set, $\mathcal{H}_i$ of the i_{th} frame. The parameter m is a learnable vector of length 2 corresponding to the number of labels in the segmentation map, M. We optimize the newly introduced parameter m to represent KEA identity during training. However, unlike the view-dependent Gaussian parameters, the KEA Identity remains uniform across different rendering views. Gaussian KEA identity ensures the continuous monitoring of each Gaussian's categorization as they evolve, thereby enabling the selective application of gradients, and the exclusive rendering of targeted objects, markedly enhancing processing efficiency in intricate scenes.

Next, we delve into the training pipeline inspired by [3, 8] in detail which consists of two stages: *(i)* Relative Pose Estimation, and *(ii)* Global 3D Scene Expansion.

Per Frame View Initialization. To begin the training process, , we randomly pick a specific frame, denoted as E_i. We then employ a pre-trained monocular depth estimator, symbolized by $\mathcal{D}$, to derive the depth map D_i for E_i. Utilizing D_i, which provides strong geometric cues independent of camera parameters, we initialize 3DGS with points extracted from monocular depth through camera intrinsics and orthogonal projection. This initialization step involves learning a set of 3D Gaussians $\mathcal{H}_i$ to minimize the photometric discrepancy between the rendered and current frames E_i. The photometric loss, $\mathcal{L}_{rgb}$, optimize the conventional 3D Gaussian parameters including color c, covariance Σ, mean μ, and opacity α. However, to initiate the KEA identity and adjust m_g for 3D Gaussians, merely relying on $\mathcal{L}_{rgb}$ is insufficient. Hence, we propose the KEA loss, denoted as $\mathcal{L}_{KEA}$, which encompasses the 2D mask M_i corresponding to E_i. We learn the KEA identity of each Gaussian point during training by applying $\mathcal{L}_{KEA}$ loss ($\mathcal{L}_{KEA}$). Overall, 3D Gaussian optimization is defined as,

$$\mathcal{H}_i^* = \arg\min_{c,\Sigma,\mu,\alpha} \mathcal{L}_{rgb}(\mathcal{R}(\mathcal{H}_i), E_i) + \arg\min_{m} \mathcal{L}_{KEA}(\mathcal{R}(\mathcal{H}_i), M_i), \tag{5}$$

where $\mathcal{R}$ signifies the 3DGS rendering function. The photometric loss $\mathcal{L}_{rgb}$ as introduced in [17] is a blend of $\mathcal{L}_1$ and D-SSIM losses:

$$\mathcal{L}_{rgb} = (1-\gamma)\mathcal{L}_1 + \gamma\mathcal{L}_{\text{D-SSIM}}, \tag{6}$$

$\mathcal{L}_{KEA}$ has two components to it. *(i)* 2D Binary Cross-Entropy Loss, and *(ii)* 3D Jensen-Shannon Divergence (JSD) Loss, and is defined as,

$$\mathcal{L}_{KEA} = \lambda_{BCE}\mathcal{L}_{BCE} + \lambda_{JSD}\mathcal{L}_{JSD} \tag{7}$$

Let $\mathcal{N}$ be the total number of pixels in the M, and $\mathcal{X}$ represent the set of all pixels. We calculate binary cross-entropy loss $\mathcal{L}_{BCE}$ as following,

$$\mathcal{L}_{BCE} = -\frac{1}{\mathcal{N}} \sum_{x\in\mathcal{X}} \Big[M_i(x) \log\left(\mathcal{R}(\mathcal{H}_i, m)(x)\right) \quad + (1 - M_i(x)) \log\left(1 - \mathcal{R}(\mathcal{H}_i, m)(x)\right) \Big] \tag{8}$$

where $M(x)$ is the value of the ground truth mask at pixel x, indicating whether the pixel belongs to the foreground (1) or the background (0). The sum computes the total loss over all pixels, and the division by $\mathcal{N}$ normalizes the loss, making it independent of the image size. A rendering operation, denoted as $\mathcal{R}(\mathcal{H}_i, m)(x)$, produces $m_{\mathcal{R}}$ for a given pixel x, which represents the weighted sum of the vector m values for the overlapping Gaussians associated with that pixel. Here, m and $m_{\mathcal{R}}$ both have a dimensionality of 2 which is intentionally kept the same as the number of classes in mask labels. We apply *softmax* function on $m_{\mathcal{R}}$ to extract KEA identity given as, KEA Identity = softmax($m_{\mathcal{R}}$). The *softmax* output is interpreted as either 0, indicating a position outside the KEA, or 1, denoting a location within the KEA.

To enhance the accuracy of Gaussian KEA identity assignment, we also introduce an unsupervised 3D Regularization Loss to directly influence the learning

of Identity vector m. This 3D Regularization Loss utilizes spatial consistency in 3D, ensuring that the Identity vector, m of the top k-nearest 3D Gaussians are similar in feature space. Specifically, we employ a symmetrical and bounded loss based on the Jensen-Shannon Divergence,

$$\mathcal{L}_{\text{JSD}} = \frac{1}{2YZ} \sum_{y=1}^{Y} \sum_{z=1}^{Z} \left[S(m_y) \log \left(\frac{2S(m_y)}{S(m_y) + S(m'_z)} \right) + S(m'_z) \log \left(\frac{2S(m'_z)}{S(m_y) + S(m'_z)} \right) \right] \quad (9)$$

Here, S indicates the *softmax* function, and m'_z represents the z^{th} Identity vector from the Z nearest neighbors in 3D space.

Relative Pose Initialization. Next, the relative camera pose is estimated for each new frame added to the training scheme. ${\mathcal{H}_i}^*$ is transformed via a learnable SE-3 affine transformation $\mathcal{M}_i$ to the subsequent frame $i+1$, where $\mathcal{H}_{i+1} = \mathcal{M}_i \odot \mathcal{H}_i$. Optimizing transformation $\mathcal{M}_i$ entails minimizing the photometric loss between the rendered image and the next frame E_{i+1},

$${\mathcal{M}_i}^* = \arg\min_{\mathcal{M}_i} \mathcal{L}_{rgb}(\mathcal{R}(\mathcal{M}_i \odot \mathcal{H}_i), E_{i+1}), \quad (10)$$

In this optimization step, we keep the attributes of ${\mathcal{H}_i}^*$ fixed to distinguish camera motion from other Gaussian transformations such as pruning, densification, and self-rotation. Applying the above 3DGS initialization to sequential image pairs enables inferring relative poses across frames. However, accumulated pose errors could adversely affect the optimization of a global scene. To tackle this challenge, we propose the gradual, sequential expansion of the 3DGS.

Gradual 3D Scene Expansion. As illustrated above, beginning with frame E_i, we initiate with a collection of 3D Gaussian points, setting the camera pose to an orthogonal configuration. Then, we calculate the relative camera pose between frames E_i and E_{i+1}. After estimating the relative camera poses, we propose to expand the 3DGS scene. This all-inclusive 3DGS optimization refines the collection of 3D Gaussian points, including all attributes, across I iterations, taking the calculated relative pose and the two observed frames as inputs. With the availability of the next frame E_{i+2} after I iterations, we repeat the above procedure: estimating the relative pose between E_{i+1} and E_{i+2}, and expanding the scene with all-inclusive 3DGS.

To perform all-inclusive 3DGS optimization, we increase the density of the Gaussians currently under reconstruction as new frames are introduced. Following [17], we identify candidates for densification by evaluating the average magnitude of position gradients in view-space. To focus densification on these yet-to-be-observed areas, we enhance the density of the universal 3DGS every I step, synchronized with the rate of new frame addition. We continue to expand the 3D Gaussian points until the conclusion of the input sequence. Through the repetitive application of both frame-relative pose estimation and all-inclusive scene expansion, 3D Gaussians evolve from an initial partial point cloud to a complete point cloud that encapsulates the entire scene over the sequence. In

our global optimization stage, we still utilize the $\mathcal{L}_{KEA}$ loss as new Gaussians are added during densification.

Pyramidal Feature Scoring. While our 2D consistent editing approach, detailed in Section 3.2, addresses various editing discrepancies, to rectify any residual inconsistencies in 2D editing, we introduce a pyramidal feature scoring method tailored for Gaussians in Key Editing Areas (KEA) identified with an identity of 1. This method begins by capturing the attributes of all Gaussians marked with KEA identity equal to 1 during initialization, establishing them as anchor points. With each densification step, these anchors are updated to mirror the present attributes of the Gaussians. Throughout the training phase, an intra-point cloud loss, $\mathcal{L}_{ipc}$ is utilized to compare the anchor state with the Gaussians' current state, maintaining that the Gaussians remain closely aligned with their initial anchors. $\mathcal{L}_{ipc}$ is defined as the weighted mean square error (MSE) between the anchor Gaussian and current Gaussian parameters with the older Gaussians getting higher weightage.

Regularizing Estimated Pose. Further, to optimize the estimated relative pose between subsequent Gaussian set, we introduce point cloud loss, $\mathcal{L}_{pc}$ similar as in [3]. While we expand the scene, $\mathcal{L}_{ipc}$ limits the deviation of the Gaussian parameters while $\mathcal{L}_{pc}$ regularizes the all-inclusive pose estimation.

$$\mathcal{L}_{pc} = D_{\text{Chamfer}}(\mathcal{M}_i^* \mathcal{H}_i^*, \mathcal{H}_{i+1}^*) \tag{11}$$

Given two Gaussians, h_i and h_j, each characterized by multiple parameters encapsulated in their parameter vectors $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ respectively, the Chamfer distance D_{Chamfer} between h_i and h_j can be formulated as:

$$D_{\text{Chamfer}}(h_i, h_j) = \sum_{p \in \boldsymbol{\theta}_i} \min_{q \in \boldsymbol{\theta}_j} \|p - q\|^2 + \sum_{q \in \boldsymbol{\theta}_j} \min_{p \in \boldsymbol{\theta}_i} \|q - p\|^2 \tag{12}$$

This equation calculates the Chamfer distance by summing the squared Euclidean distances from each parameter in h_i to its closest counterpart in h_j, and vice versa, thereby quantifying the similarity between the two Gaussians across all included parameters such as color, opacity, etc. Combining all the loss components results in the total loss function during scene expansion,

$$\mathcal{L}_T = \lambda_{rgb}\mathcal{L}_{rgb} + \lambda_{KEA}\mathcal{L}_{KEA} + \lambda_{ipc}\mathcal{L}_{ipc} + \lambda_{pc}\mathcal{L}_{pc} \tag{13}$$

where λ_{rgb}, λ_{KEA}, λ_{ipc} and λ_{pc} act as weighting factors for the respective loss terms.

4 Evaluation

4.1 Implementation Details

In our approach, we employ PyTorch [33] for the development, specifically focusing on 3D Gaussian splatting. GPT-3.5 Turbo [5] is used for identifying the editing attributes to identify the KEA. For segmentation purposes, SAM [20] is

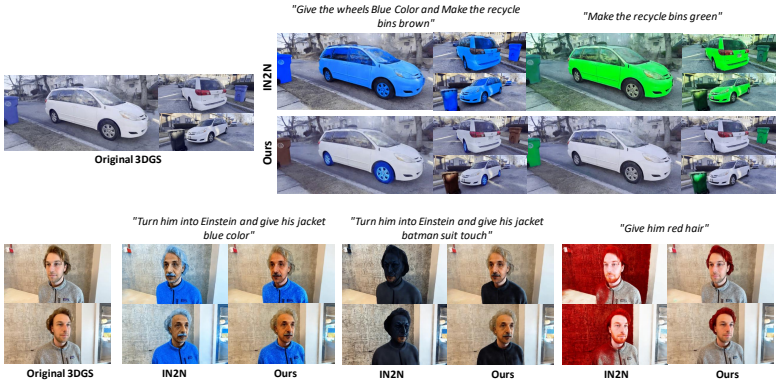

Fig. 4: Qualitative comparison of our method with the IN2N [11] over two separate scenes. When the editing prompt requests "Give the wheels Blue Color and Make the recyclebins brown," IN2N [11] inadvertently alters the complete van color to blue as well, instead of just changing the tire color. It must be noted that IN2N [11] uses poses from COLMAP, while *3DEgo* estimates poses while constructing the 3D scene.

used to generate the masks based on the key editing attributes identifying the KIA. For zero-shot point tracking, we employ a point-tracker as proposed in [34]. The editing tasks are facilitated by the Instruct Pix2Pix [4] 2D diffusion model by incorporating the masks to limit the editing within KEA. Additional details are in *supplementary material*.

4.2 Baseline and Datasets

We carry out experiments across a variety of public datasets as well as our prepared **GS25** dataset.

GS25 Dataset comprises 25 casually captured monocular videos using mobile phones for comprehensive 3D scene analysis. This approach ensures the dataset's utility in exploring and enhancing 360-degree real-world scene reconstruction technologies. To further assess the efficacy of the proposed 3D editing framework, we also conducted comparisons across 5 public datasets: (i) IN2N [11], (ii) Mip-NeRF [2],(iii) NeRFstudio Dataset [42], (iv) Tanks & Temples [21] and (v) CO3D-V2 [36]. We specifically validate the robustness of our approach on the CO3D dataset, which comprises thousands of object-centric videos. In our study, we introduce a unique problem, making direct comparisons with prior research challenging. Nonetheless, to assess the robustness of our method, we contrast it with

Table 1: Average runtime efficiency across 25 edits from the GS25 dataset (Approx. minutes).

Method	COLMAP	Model Initialization	Scene Editing
Instruct-N2N [11]	13min	22min	250min
Ours	✗	✗	25min

Fig. 5: Our approach surpasses Gaussian Grouping [50] in 3D object elimination across different scenes from GS25 and Tanks & Temple datasets. *3DEgo* is capable of eliminating substantial objects like statues from the entire scene while significantly minimizing artifacts and avoiding a blurred background.

state-of-the-art (SOTA) 3D editing techniques that rely on poses derived from COLMAP. Additionally, we present quantitative evaluations alongside pose-free 3D reconstruction approaches, specifically NoPeNeRF [3], and BARF [25]. In the pose-free comparison, we substitute only our 3D scene reconstruction component with theirs while maintaining our original editing framework unchanged. We present a time-cost analysis in Table 1 that underscores the rapid text-conditioned 3D reconstruction capabilities of 3DEgo.

4.3 Qualitative Evaluation

As demonstrated in Figure 4, our method demonstrates exceptional prowess in **local editing**, enabling precise modifications within specific regions of a 3D scene without affecting the overall integrity. Our method also excels in **multi-attribute editing**, seamlessly combining changes across color, texture, and geometry within a single coherent edit. We also evaluate our method for the **object removal task**. The goal of 3D object removal is to eliminate an object from a 3D environment, potentially leaving behind voids due to the lack of observational

Table 2: Comparing With Pose-known Methods. Quantitative evaluation of 200 edits across GS25, IN2N, Mip-NeRF, NeRFstudio, Tanks & Temples, and CO3D-V2 datasets against the methods that incorporate COLMAP poses. The top-performing results are emphasized in bold.

Datasets	DreamEditor			IN2N			Ours		
	CTIS↑	CDCR↑	E-PSNR↑	CTIS↑	CDCR↑	E-PSNR↑	CTIS↑	CDCR↑	E-PSNR↑
GS25 (Ours)	0.155	0.886	22.750	0.142	0.892	23.130	**0.169**	**0.925**	**23.660**
Mip-NeRF	0.149	0.896	23.920	0.164	**0.917**	22.170	**0.175**	0.901	**24.250**
NeRFstudio	0.156	0.903	23.670	**0.171**	0.909	**25.130**	0.163	**0.931**	24.990
CO3D-V2	0.174	0.915	24.880	0.163	0.924	25.180	**0.179**	**0.936**	**26.020**
IN2N	0.167	0.921	24.780	0.179	0.910	**26.510**	**0.183**	**0.925**	26.390
Tanks & Temples	0.150	0.896	23.970	**0.170**	0.901	23.110	0.164	**0.915**	**24.190**

Table 3: Comparing With Pose-Unknown Methods. Quantitative analysis of 200 edits applied to six datasets, comparing methods proposed for NeRF reconstruction without known camera poses. The top-performing results are emphasized in bold.

Datasets	BARF [25]			Nope-NeRF [3]			Ours		
	CTIS↑	CDCR↑	E-PSNR↑	CTIS↑	CDCR↑	E-PSNR↑	CTIS↑	CDCR↑	E-PSNR↑
GS25 (Ours)	0.139	0.797	20.478	0.128	0.753	19.660	**0.169**	**0.925**	**23.660**
Mip-NeRF	0.134	0.806	21.332	0.147	0.820	18.799	**0.175**	**0.901**	**24.250**
NeRFstudio	0.140	0.813	20.116	0.138	0.773	21.360	**0.163**	**0.931**	**24.990**
CO3D-V2	0.157	0.820	21.148	0.129	0.824	17.971	**0.179**	**0.936**	**26.020**
IN2N	0.150	0.829	22.092	0.161	0.818	22.604	**0.183**	**0.925**	**26.390**
Tanks & Temples	0.135	0.806	21.573	0.157	0.810	20.904	**0.164**	**0.915**	**24.190**

data. For the object removal task, we identify and remove the regions based on the 2D mask, M. Subsequently, we focus on inpainting these "invisible regions" in the original 2D frames using LAMA [41]. In Figure 5, we demonstrate our 3DEgo's effectiveness in object removal compared to Gaussian Grouping. Our method's reconstruction output notably surpasses that of Gaussian Grouping [50] in terms of retaining spatial accuracy and ensuring consistency across multiple views.

4.4 Quantitative Evaluation

In our quantitative analysis, we employ three key metrics: CLIP Text-Image Direction Similarity (CTIS) [9], CLIP Direction Consistency Score (CDCR) [11], and Edit PSNR (E-PSNR). We perform 200 edits across the six datasets listed above. We present quantitative comparisons with COLMAP-based 3D editing techniques in Table 2. Additionally, we extend our evaluation by integrating pose-free 3D reconstruction methods into our pipeline, with the performance outcomes detailed in Table 3.

Fig. 6: Our method, **3DEgo** achieves precise editing without using any SfM poses. To construct the IP2P+COLMAP 3D scene, we train nerfacto [42] model on IP2P [4] edited frames.

5 Ablations

To assess the influence of different elements within our framework, we employ PSNR, SSIM, and LPIPS metrics across several configurations. Given that images undergo editing before the training of a 3D model, our focus is on determining the effect of various losses on the model's rendering quality. The outcomes are documented in Table 4, showcasing IP2P+COLMAP as the baseline, where

images are edited using the standard IP2P approach [4] and COLMAP-derived poses are utilized for 3D scene construction.

Although the IP2P+COLMAP setup demonstrates limited textual fidelity due to editing inconsistencies (see Figure 6), we are only interested in the rendering quality in this analysis to ascertain our approach's effectiveness. Table 4 illustrates the effects of different optimization hyperparameters on the global scene expansion. The findings reveal that excluding $\mathcal{L}_{KEA}$ in the scene expansion process minimally affects rendering quality. On the other hand, omitting $\mathcal{L}_{ipc}$ leads to unwanted densification resulting in the inferior performance of the trained model.

Table 4: Ablation study results on **GS25** dataset.

Method	PSNR↑	SSIM↑	LPIPS↓
Ours	**27.86**	**0.90**	**0.18**
IP2P+COLMAP	23.87	0.79	0.23
Ours w/o L_{KEA}	26.73	0.88	0.19
Ours w/o L_{ipc}	22.46	0.0.78	0.24
Ours w/o L_{pc}	25.18	0.84	0.20

6 Limitation

Our approach depends on the pre-trained IP2P model [4], which has inherent limitations, especially evident in specific scenarios. For instance, Figure 7 shows the challenge with the prompt *"Make the car golden and give wheels blue color"*. Unlike IN2N [11], which introduces unspecific color changes on the van's windows. Our method offers more targeted editing but falls short of generating ideal results due to IP2P's limitations in handling precise editing tasks.

Fig. 7: Due to the limitations of the IP2P model, our method inadvertently alters the colors of the van's windows, which is not the desired outcome.

7 Conclusion

3DEgo marks a pivotal advancement in 3D scene reconstruction from monocular videos, eliminating the need for conventional pose estimation methods and model initialization. Our method integrates frame-by-frame editing with advanced consistency techniques to efficiently generate photorealistic 3D scenes directly from textual prompts. Demonstrated across multiple datasets, our approach showcases superior editing speed, precision, and flexibility. *3DEgo* not only simplifies the 3D editing process but also broadens the scope for creative content generation from readily available video sources. This work lays the groundwork for future innovations in accessible and intuitive 3D content creation tools.

Acknowledgement

This work was partially supported by the NSF under Grant Numbers OAC-1910469 and OAC-2311245.

References

1. Bao, C., Zhang, Y., Yang, B., Fan, T., Yang, Z., Bao, H., Zhang, G., Cui, Z.: Sine: Semantic-driven image-based nerf editing with prior-guided editing field. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20919–20929 (2023)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
3. Bian, W., Wang, Z., Li, K., Bian, J.W., Prisacariu, V.A.: Nope-nerf: Optimising neural radiance field with no pose prior. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4160–4169 (2023)
4. Brooks, T., Holynski, A., Efros, A.A.: Instructpix2pix: Learning to follow image editing instructions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18392–18402 (2023)
5. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
6. Chiang, P.Z., Tsai, M.S., Tseng, H.Y., Lai, W.S., Chiu, W.C.: Stylizing 3d scene via implicit representation and hypernetwork. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1475–1484 (2022)
7. Dong, J., Wang, Y.X.: Vica-nerf: View-consistency-aware 3d editing of neural radiance fields. Advances in Neural Information Processing Systems **36** (2024)
8. Fu, Y., Liu, S., Kulkarni, A., Kautz, J., Efros, A.A., Wang, X.: Colmap-free 3d gaussian splatting (2023), https://arxiv.org/abs/2312.07504
9. Gal, R., Patashnik, O., Maron, H., Chechik, G., Cohen-Or, D.: Stylegan-nada: Clip-guided domain adaptation of image generators. arXiv preprint arXiv:2108.00946 (2021)
10. Gao, W., Aigerman, N., Groueix, T., Kim, V.G., Hanocka, R.: Textdeformer: Geometry manipulation using text guidance. arXiv preprint arXiv:2304.13348 (2023)
11. Haque, A., Tancik, M., Efros, A.A., Holynski, A., Kanazawa, A.: Instruct-nerf2nerf: Editing 3d scenes with instructions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19740–19750 (2023)
12. Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., Cohen-Or, D.: Prompt-to-prompt image editing with cross attention control. arXiv preprint arXiv:2208.01626 (2022)
13. Hong, F., Zhang, M., Pan, L., Cai, Z., Yang, L., Liu, Z.: Avatarclip: Zero-shot text-driven generation and animation of 3d avatars. ACM Transactions on Graphics (TOG) **41**(4), 1–19 (2022)
14. Huang, Y.H., He, Y., Yuan, Y.J., Lai, Y.K., Gao, L.: Stylizednerf: consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18342–18352 (2022)

15. Jeong, Y., Ahn, S., Choy, C., Anandkumar, A., Cho, M., Park, J.: Self-calibrating neural radiance fields. In: ICCV (2021)
16. Karim, N., Khalid, U., Iqbal, H., Hua, J., Chen, C.: Free-editor: Zero-shot text-driven 3d scene editing. arXiv preprint arXiv:2312.13663 (2023)
17. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics (ToG) **42**(4), 1–14 (2023)
18. Khalid, U., Iqbal, H., Karim, N., Hua, J., Chen, C.: Latenteditor: Text driven local editing of 3d scenes. arXiv preprint arXiv:2312.09313 (2023)
19. Kim, S., Lee, K., Choi, J.S., Jeong, J., Sohn, K., Shin, J.: Collaborative score distillation for consistent visual editing. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), https://openreview.net/forum?id=0tEjORCGFD
20. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al.: Segment anything. arXiv preprint arXiv:2304.02643 (2023)
21. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (2017)
22. Kobayashi, S., Matsumoto, E., Sitzmann, V.: Decomposing nerf for editing via feature field distillation. arXiv preprint arXiv:2205.15585 (2022)
23. Li, Y., Lin, Z.H., Forsyth, D., Huang, J.B., Wang, S.: Climatenerf: Physically-based neural rendering for extreme climate synthesis. arXiv e-prints pp. arXiv–2211 (2022)
24. Li, Y., Dou, Y., Shi, Y., Lei, Y., Chen, X., Zhang, Y., Zhou, P., Ni, B.: Focaldreamer: Text-driven 3d editing via focal-fusion assembly. arXiv preprint arXiv:2308.10608 (2023)
25. Lin, C.H., Ma, W.C., Torralba, A., Lucey, S.: Barf: Bundle-adjusting neural radiance fields. In: ICCV (2021)
26. Liu, H.K., Shen, I., Chen, B.Y., et al.: Nerf-in: Free-form nerf inpainting with rgb-d priors. arXiv preprint arXiv:2206.04901 (2022)
27. Long, X., Guo, Y.C., Lin, C., Liu, Y., Dou, Z., Liu, L., Ma, Y., Zhang, S.H., Habermann, M., Theobalt, C., et al.: Wonder3d: Single image to 3d using cross-domain diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9970–9980 (2024)
28. Michel, O., Bar-On, R., Liu, R., et al.: Text2mesh: Text-driven neural stylization for meshes. In: CVPR 2022. pp. 13492–13502 (2022)
29. Nguyen-Phuoc, T., Liu, F., Xiao, L.: Snerf: stylized neural implicit representations for 3d scenes. arXiv preprint arXiv:2207.02363 (2022)
30. Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., Chen, M.: Glide: Towards photorealistic image generation and editing with text-guided diffusion models. arXiv preprint arXiv:2112.10741 (2021)
31. Noguchi, A., Sun, X., Lin, S., Harada, T.: Neural articulated radiance field. In: ICCV 2021. pp. 5762–5772 (2021)
32. Park, H.S., Jun, C.H.: A simple and fast algorithm for k-medoids clustering. Expert systems with applications **36**(2), 3336–3341 (2009)
33. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
34. Rajič, F., Ke, L., Tai, Y.W., Tang, C.K., Danelljan, M., Yu, F.: Segment anything meets point tracking. arXiv preprint arXiv:2307.01197 (2023)

35. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 (2022)
36. Reizenstein, J., Shapovalov, R., Henzler, P., Sbordone, L., Labatut, P., Novotny, D.: Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10901–10911 (2021)
37. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR 2022. pp. 10684–10695 (2022)
38. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22500–22510 (2023)
39. Saharia, C., Chan, W., Saxena, S.e.a.: Photorealistic text-to-image diffusion models with deep language understanding. NeurIPS 2022 **35**, 36479–36494 (2022)
40. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR (2016)
41. Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., Lempitsky, V.: Resolution-robust large mask inpainting with fourier convolutions. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 2149–2159 (2022)
42. Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., et al.: Nerfstudio: A modular framework for neural radiance field development. In: ACM SIGGRAPH 2023 Conference Proceedings. pp. 1–12 (2023)
43. Tschernezki, V., Laina, I., Larlus, D., Vedaldi, A.: Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In: 2022 International Conference on 3D Vision (3DV). pp. 443–453. IEEE (2022)
44. Wang, C., Chai, M., He, M., et al.: Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In: CVPR 2022. pp. 3835–3844 (2022)
45. Wang, C., Jiang, R., Chai, M., He, M., Chen, D., Liao, J.: Nerf-art: Text-driven neural radiance fields stylization. IEEE Transactions on Visualization and Computer Graphics (2023)
46. Weng, H., Yang, T., Wang, J., Li, Y., Zhang, T., Chen, C., Zhang, L.: Consistent123: Improve consistency for one image to 3d object synthesis. arXiv preprint arXiv:2310.08092 (2023)
47. Wu, Q., Tan, J., Xu, K.: Palettenerf: Palette-based color editing for nerfs. arXiv preprint arXiv:2212.12871 (2022)
48. Xu, T., Harada, T.: Deforming radiance fields with cages. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII. pp. 159–175. Springer (2022)
49. Yang, B., Bao, C., Zeng, J., Bao, H., Zhang, Y., Cui, Z., Zhang, G.: Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In: European Conference on Computer Vision. pp. 597–614. Springer (2022)
50. Ye, M., Danelljan, M., Yu, F., Ke, L.: Gaussian grouping: Segment and edit anything in 3d scenes. arXiv preprint arXiv:2312.00732 (2023)
51. Zhang, K., Kolkin, N., Bi, S., Luan, F., Xu, Z., Shechtman, E., Snavely, N.: Arf: Artistic radiance fields. In: European Conference on Computer Vision. pp. 717–733. Springer (2022)

52. Zhuang, J., Wang, C., Lin, L., Liu, L., Li, G.: Dreameditor: Text-driven 3d scene editing with neural fields. In: SIGGRAPH Asia 2023 Conference Papers. pp. 1–10 (2023)