

Editing 3D Scenes via Text Prompts Without Retraining

Shuangkang Fang^{ID}, Yufeng Wang^{ID}, Yi-Hsuan Tsai^{ID}, Wenrui Ding^{ID}, *Member, IEEE*, Yi Yang^{ID},
Shuchang Zhou^{ID}, and Ming-Hsuan Yang^{ID}, *Fellow, IEEE*

Abstract—Numerous diffusion models have been developed for 2D image synthesis and editing, and recently they are extended to 3D scene editing tasks. However, editing 3D scenes is still in its early stages, and the challenges of scene representations and multi-view consistency need to be addressed. A notable limitation of existing approaches is the need for specific modules for different edits and model retraining for each scene. To tackle these issues, we propose a novel and versatile text-driven 3D scene editing method, termed DN2N, which allows for the *direct* acquisition of the editing results without the requirement for retraining. Our method employs off-the-shelf text-based editing models of 2D images to modify the multi-view images of a 3D scene. A content filtering process is then applied to discard poorly edited images that disrupt 3D consistency. We consider the remaining inconsistency as a problem of removing noise perturbations and solve it by generating data with similar perturbation characteristics for training. We develop a versatile NeRF model structure and propose two novel cross-view regularization terms to help the DN2N mitigate these perturbations. Empirical results show that our method achieves multiple editing types based solely on text prompts, including but not limited to appearance editing, weather transition, object changing, and style transfer. Most importantly, DN2N exhibits a versatility of editing capabilities, eliminating the need to customize or retrain editing models for specific scenes or editing types. Namely, DN2N achieves comparable total editing time to the 3DGS-based editing method, enhancing its practical value.

Index Terms—3D scene editing, 3D object editing, text prompts, NeRF, IBRNet, diffusion model.

I. INTRODUCTION

SIGNIFICANT advances in neural radiance fields (NeRF) techniques [1], [2], [3], [4], [5], [6], [7], [8] and Gaussian-

Received 18 August 2025; revised 13 January 2026; accepted 21 February 2026. Date of publication 27 February 2026; date of current version 17 June 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62501036 and in part by the Fundamental Research Funds for the Central Universities. Recommended for acceptance by N. Khademi Kalantari. (*Corresponding author: Yufeng Wang.*)

Shuangkang Fang is with the School of Electrical and Information Engineering, Beihang University, Beijing 100191, China (e-mail: skfang@buaa.edu.cn).

Yufeng Wang and Wenrui Ding are with the Institute of Unmanned System, Beihang University, Beijing 100191, China (e-mail: wyfeng@buaa.edu.cn; ding@buaa.edu.cn).

Yi-Hsuan Tsai is with Atmanity Inc., Santa Clara, CA 95053 USA (e-mail: wasidennis@gmail.com).

Yi Yang and Shuchang Zhou are with StepFun Inc., Beijing 100190, China (e-mail: ahyangyi@gmail.com; zhousc6@xiaopeng.com).

Ming-Hsuan Yang is with the University of California at Merced, Merced, CA 95343 USA (e-mail: mhyang@ucmerced.edu).

The project website is available at <https://sk-fun.fun/DN2N>.

This article has supplementary downloadable material available at <https://doi.org/10.1109/TVCG.2026.3668359>, provided by the authors.

Digital Object Identifier 10.1109/TVCG.2026.3668359

Splatting [9], [10], [11], [12] have been modeled for specific types of 3D manipulation, such as appearance editing [13], [14], scene composition [15], [16], weather transformation [17], multiple editing [18], [19], and style transfer [20], [21], [22], [23]. Despite the demonstrated success, existing methods [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [24] often require user manipulations and typically rely on known editing types in advance, resulting in limited modification capabilities. To address these challenges, several approaches leverage existing multi-modal models [25], [26], [27] on 2D images to enhance editing capabilities, while utilizing NeRF [1] or Gaussian-Splatting [9] frameworks to represent 3D scenes. This combination has successfully enabled text-driven 3D scene editing, significantly improving the model's editing capabilities and user-friendliness. However, all the above methods are not versatile since they require training 10,000 models for 100 types of editing on 100 scenes. Thus, this paper aims to design a novel approach that provides diverse and user-friendly editing capabilities while allowing users to directly edit new scenes without retraining, i.e., *using one single model to perform different edits on various 3D scenes*.

With advances in NeRF models [3], [28], [29], one goal is to increase versatility for diverse scenes. Nevertheless, directly designing it this way poses two challenges: (1) insufficient task-specific training data to ensure strong versatility; (2) inherent 3D inconsistency due to directly using a 2D image editing model for 3D scenes. To address these issues, we propose directly modeling the inherent 3D inconsistency and leveraging existing tools to generate sufficient training data with similar characteristics. This enables the model to acquire robust versatility to mitigate this inconsistency. We show this idea of treating inconsistency as unknown perturbations in Fig. 2 and propose the Direct-NeRF2NeRF (DN2N) model.

Within DN2N, we first utilize a 2D editing model [27] to perform the preliminary editing on the multi-view images of a 3D scene, and then apply a designed content filter to remove several images with poor editing results. The remaining images may still contain inconsistent 3D results, which we consider as perturbations to the ideally edited images due to the inherent stochastic and diverse nature of the 2D model (see Fig. 2). To remove such perturbations, we generate numerous training data pairs with similar characteristics to train our model. First, we utilize the BLIP model [30] and GPT model [31] to obtain the input and target captions required for editing, then apply random slight edits to the 3D scene images to simulate the perturbations.

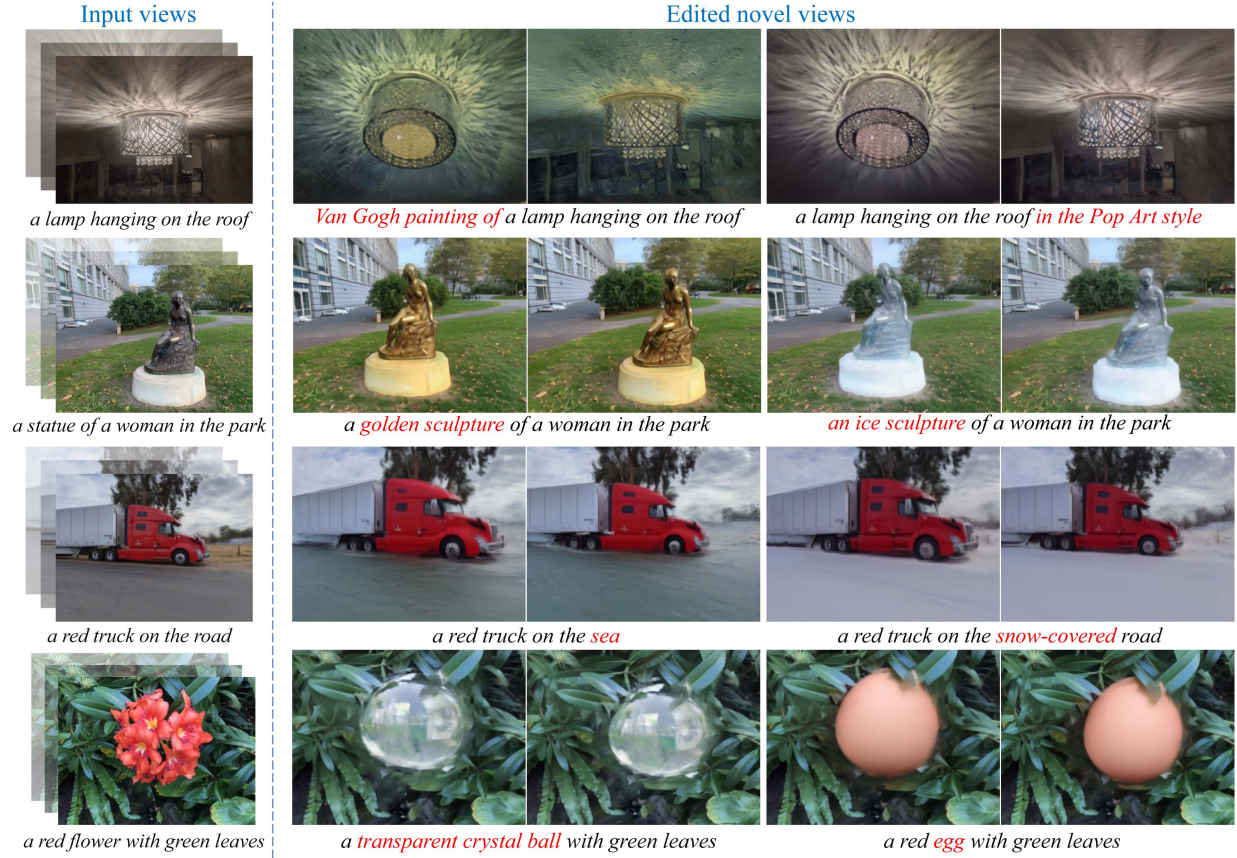


Fig. 1. Visualization results of our method in 3D scene editing. The proposed DN2N method enables users to obtain realistic and 3D consistent novel views that correspond to the target caption without retraining a new model.

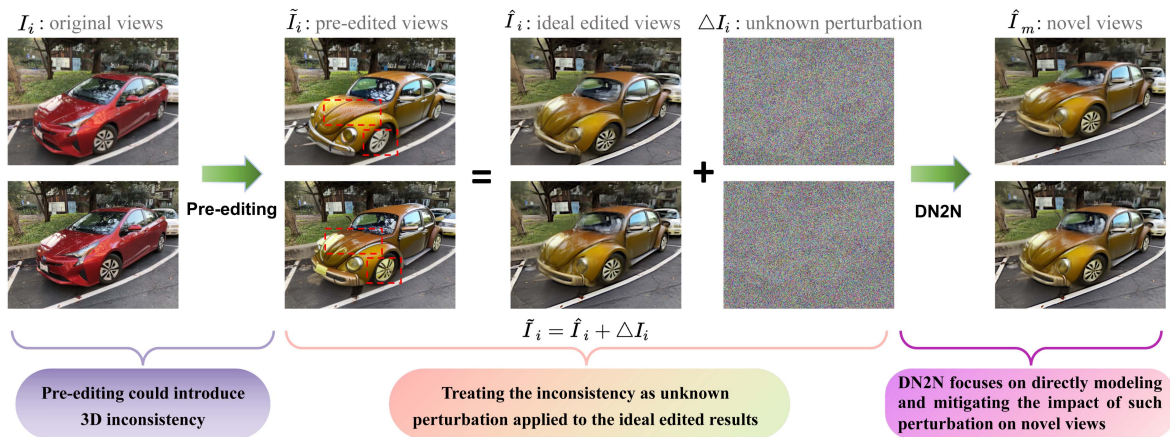


Fig. 2. Overview of DN2N. *Left*: While directly applying a 2D image editing model to pre-edit multi-views of a scene can generate an edited result $\tilde{I}_i$, it introduces noticeable 3D inconsistency (marked by red box). *Middle*: Unlike previous methods that iteratively train a new model to achieve 3D-consistent results, we consider inconsistency as an unknown perturbation ΔI_i to the desired results $\hat{I}_i$ and focus on modeling and mitigating ΔI_i . *Right*: Within DN2N, we propose a versatile model designed for 3D scene editing tasks. This model synthesizes 3D-consistent novel views from several 3D-inconsistent pre-edited views $\tilde{I}_i$, without the need to retrain new models.

As such, images with added random perturbations are utilized as inputs, while clean images serve as ground truth for training a versatile model. To further remove perturbations, we develop a NeRF architecture and introduce two cross-view regularization terms during training to facilitate 3D consistency, including the

self and neighboring views. The former requires the model to generate consistent results for the same target view derived from two different source views. Meanwhile, the latter enforces the overlapping pixel values between the target and adjacent views to be approximately close.

The main contributions of this work are:

- We develop a novel and versatile 3D scene editing framework, named DN2N. It employs off-the-shelf 2D editing models for 3D scene manipulation. The induced 3D inconsistency is modeled as perturbations and addressed by generating training data pairs with similar perturbation characteristics for optimization.
- We present a versatile NeRF model for 3D scene editing, incorporating multi-view aggregation and unedited image information to render consistent results. To train such a model, we generate over 480,000 training 3D scenes.
- We integrate cross-view regularization terms in the training process and develop a content filter in the inference process, enabling DN2N to acquire the capability to eliminate perturbations more efficiently.
- We conduct extensive experiments across multiple datasets and editing types. DN2N eliminates the need for scene-specific customization or retraining, achieving runtime and memory storage comparable to state-of-the-art methods.

II. RELATED WORK

Novel View Synthesis: Novel view synthesis (NVS) based on NeRF and Gaussian-Splatting has recently gained significant attention in the vision and graphics communities [1], [2], [3], [9], [10], [11], [12], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41]. As this work aims to exploit the potential of NVS techniques for 3D scene editing tasks, we review the NeRF models closely related to this task. NeRF represents the structure and appearance of a 3D scene by using a neural network that takes the spatial location and view direction as input and outputs the corresponding color and opacity at each pixel. Subsequent works further improve NeRF, such as speeding up the training process [2], [36], designing better sampling strategies [33], and enhancing versatility [3], [28], [29]. These NeRF models primarily involve learning a universal network across different scenes, leveraging features from multiple views to render novel views. However, a trade-off of generalizable models is their reliance on image data during the forward pass, which incurs additional storage overhead for these images. To repair degraded 3D representations, Difix3D+ [42] utilizes single-step diffusion to mitigate artifacts in under-constrained regions, while SimVS [43] employs a harmonization network to resolve multi-view inconsistencies in casual captures.

Diffusion-based Image Editing: Diffusion-based models have been widely used in image generation [44], [45], [46], [47], [48], [49], [50], [51]. Numerous text-based image editing methods have recently been developed, such as GLIDE [52] and Stable Diffusion [49]. Imagic [53] finetunes images according to text, and Prompt-to-prompt [54] preserves unedited regions by utilizing cross-attention information. In [55], Pix2pix-zero employs embedding vector mechanisms to establish controllable editing directions for images. Recently, InstructPix2Pix [26] trains a model by generating many text-editing image pairs using GPT [31] and Stable Diffusion. Null-text inversion [27] enhances image-controlled editing through an accurate inversion process. While these methods perform well in editing images,

they fail to achieve 3D-consistent edits when confronted with 3D scenes comprising multiple views. This limitation highlights the need for more sophisticated approaches to maintain consistency across multiple views in 3D scenes.

3D Scene Editing: Numerous 3D scene editing approaches have been developed based on point cloud [56], [57] and meshes [58], [59], [60], but their inherent representations limit scalability and editing flexibility. Recently, NeRF and Gaussian Splatting-based approaches have emerged for 3D scene editing. [10], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [24], [37], [57], [61], [62]. However, they are usually restricted to performing a single editing type. Several approaches are proposed to address this issue [63], [64], [65]. Instruct-N2N [65] enables diverse controllable editing of 3D scenes by pre-editing 2D images using InstructPix2Pix [26]. GaussianEditor [10] achieves comparable results by employing Gaussian-Splitting for scene representation. FocalDreamer [66] enables fine-grained editing by fusing base shapes with editable parts. TIPEditor [67] facilitates precise localization using reference images and bounding boxes, while DGE [68] ensures efficient 3DGS optimization by enforcing multi-view consistency via geometric priors. However, previous methods, such as Instruct-N2N or GaussianEditor, have two notable drawbacks. First, retraining is necessary for each scene, resulting in significant computation and memory overhead. Second, they require continuously generating new training data and updating model parameters during training, which is time-consuming and requires a lot of storage space. In contrast, our method addresses these two challenges by only using a single editing model for all 3D scenes and editing types without retraining, decreasing model storage consumption and training time.

III. PRELIMINARIES

Neural radiance fields: NeRF utilizes an implicit function to represent a 3D scene, which maps the spatial point $\mathbf{x} = (x, y, z)$ and view direction $\mathbf{d} = (\theta, \phi)$ to the density σ and color $\mathbf{c}$. The implicit function is represented by an MLP network, $F_{\Theta} : (\mathbf{x}, \mathbf{d}) \rightarrow (\sigma, \mathbf{c})$, where Θ represents the weights of the network. For a ray $\mathbf{r}$ originating at $\mathbf{o}$ with direction $\mathbf{d}$, the RGB value $\hat{\mathbf{C}}(\mathbf{r})$ of the ray is estimated through numerical quadrature of the color $\mathbf{c}$ and density σ of the N spatial points:

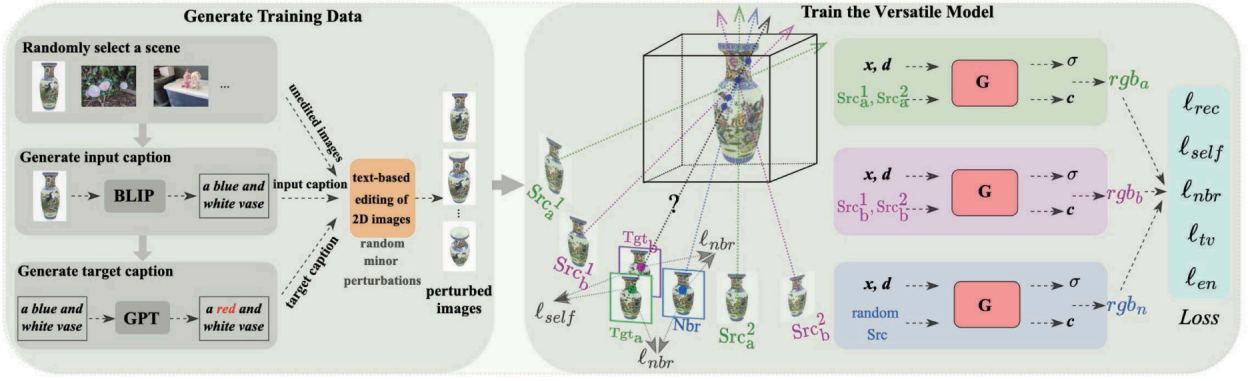
$$\hat{\mathbf{C}}(\mathbf{r}) = \sum_i^N T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i, \quad (1)$$

where $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$, and δ_i is the distance between adjacent sample points.

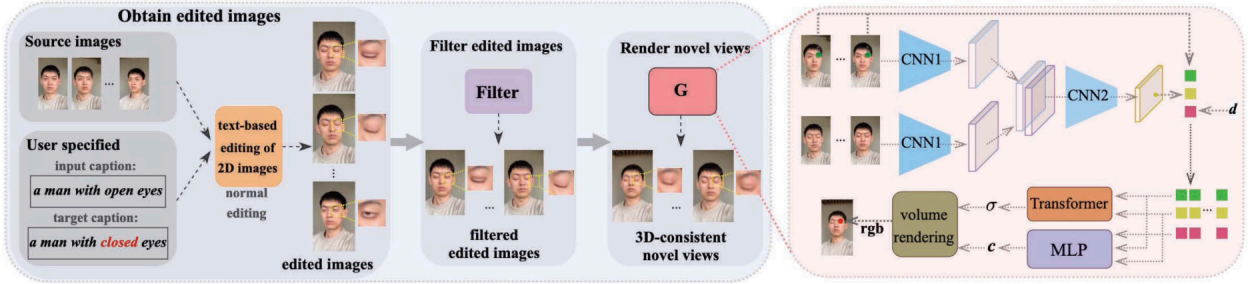
Text-guided diffusion models and image editing: Text-guided diffusion models aim to generate an output image z_0 from a random noise vector z_t under a textual condition $\mathcal{P}$. To denoise the input sequentially, the model ε_{θ} is trained to predict artificial noise by minimizing the objective function:

$$\min_{\theta} E_{z_0, \varepsilon \sim N(0, I), t \sim \text{Uniform}(1, T)} \|\varepsilon - \varepsilon_{\theta}(z_t, t, \mathcal{C})\|^2, \quad (2)$$

where $\mathcal{C}$ denotes the embedding of text condition, and z_t is a noised sample at timestamp t . For inference, given a vector z_T ,



(a) **Training stage.** *Left:* We randomly select a scene and feed its one image into the BLIP to obtain the input caption. Subsequently, we utilize the GPT model to rewrite the input caption to generate the target caption. According to the two captions, we use a fixed text-based editing method to construct a new perturbed scene with inconsistency for the training process (see Fig. 4 and Sec IV-C for details). *Right:* We design a versatile NeRF model G (see detailed architecture in supplementary material) and Sec IV-D) that is trained on over 480,000 constructed 3D scenes. The model G renders two target views $[Tgt_a, Tgt_b]$ with the same camera pose from two different groups of source views $[\{Src_a^*\}, \{Src_b^*\}]$, as well as renders its neighboring view Nbr . We use these rendered views to design cross-view regularization terms ℓ_{self} and ℓ_{nbr} (see Sec IV-E for loss design). Once the training is completed, the model can remove inconsistent perturbations. Thus, there is no need to retrain the model for new scenes, and the editing results can be directly obtained through the inference stage.



(b) **Inference stage.** *Left:* Given a new scene and its input caption, along with the target caption for the desired editing effect, we first apply standard magnitude editing to the scene's multi-views. *Middle:* We devise a content filter to eliminate the images with subpar editing results and compromised consistency (see Sec IV-F for the detailed filter process). *Right:* Finally, we utilize the well-trained model G to directly obtain the 3D-consistent novel views that conform to the target caption. These processes do not require retraining of the model.

Fig. 3. Illustration of the DN2N framework. See Section IV for the detailed description.

sequential prediction gradually removes its noise using a trained network for T steps [50]. Amplifying the effect of conditioned text is a significant challenge in generation. To address this issue, Ho et al. [69] propose a guidance technique that eliminates the need for a classifier in unconditional prediction and extends it to conditioned prediction scenarios. With the null text embedding, $\emptyset$, and a guidance scale parameter w , the guidance prediction is given by:

$$\tilde{\epsilon}_{\theta}(z_t, t, \mathcal{C}, \emptyset) = w \cdot \epsilon_{\theta}(z_t, t, \mathcal{C}) + (1 - w) \cdot \epsilon_{\theta}(z_t, t, \emptyset). \quad (3)$$

In this work, we adjust the parameters w and T to control the degree of image editing.

IV. METHOD

A. Overall Framework

Fig. 3 shows the main components of the DN2N model. It first involves training the model on a variety of scenes. For each scene, we first utilize BLIP [30] to obtain its description as the input caption and then employ GPT [31] to randomly modify this caption to create the target caption. We then use a 2D image editing model [27] to apply minor perturbations to the

scene based on these captions to obtain training data pairs. The training objective of our model is to remove these perturbations and reconstruct the 3D scene (see (8)). We devise a novel NeRF model structure with versatility for rendering consistent results. In addition, we introduce cross-view regularization terms to aid model training. Specifically, we use two sets of randomly sampled source views to render the same target view, resulting in Tgt_a and Tgt_b , respectively, and render a neighboring view around the target view to obtain Nbr . We then impose a consistency loss (see (9) and (10)) on the three rendered results. During inference, we begin by applying standard magnitude editing to a new scene and filter out images with poor editing effects (see (13)). Finally, we feed the filtered images into the well-trained versatile model to obtain edited novel views directly.

B. Mathematical Model

In this part, we systematically provide a theoretical derivation and mathematical modeling of DN2N and noise perturbations.

A 3D scene consists of N images and corresponding camera parameters, $\{I_i, P_i\}_{i=1}^N$. We employ a 2D editing model $\mathcal{F}$ to

obtain the pre-edited image $\tilde{I}_i$ for each I_i :

$$\tilde{I}_i = \mathcal{F}(I_i, C_{in}^i, C_{tgt}^i, \theta), \quad (4)$$

where C_{in}^i is input caption for the unedited image I_i , C_{tgt}^i is target caption for the edited image $\tilde{I}_i$, and θ is the hyper-parameters of model $\mathcal{F}$. After removing the poorly edited images in $\{\tilde{I}_i, P_i\}_{i=1}^N$, the resulting data is denoted as $\{\tilde{I}_m, P_m\}_{m=1}^M$, where $M \leq N$. When $P_k \neq P_m$, the versatile NeRF model G with parameter Θ predicts the target view $\hat{I}_m$ using K source views $\{\tilde{I}_k, P_k\}_{k=1}^K$:

$$\hat{I}_m = G(\{\tilde{I}_k, P_k\}_{k=1}^K, \Theta). \quad (5)$$

Assuming the ideal ground truth (GT) of edited images with 3D consistency is denoted as I_m^{gt} , the objective function is:

$$\arg \min_{\Theta} \sum_{m=1}^M \left\| I_m^{gt} - \hat{I}_m \right\|^2. \quad (6)$$

However, the ideal GT of the edited scene is not at our disposal. Thus, we express the pre-edited image $\tilde{I}_m$ as:

$$\tilde{I}_m = I_m^{gt} + \Delta I_m. \quad (7)$$

This implies that the inconsistent image $\tilde{I}_m$ can be perceived as the consistent GT image I_m^{gt} with perturbations ΔI_m . The consistent editing results can be obtained by $I_m^{gt} = \tilde{I}_m - \Delta I_m$. Thus, we introduce similar random perturbations ΔI_i to the clean 3D scene images $\{I_i\}_{i=1}^N$ by adjusting the parameters θ in $\mathcal{F}$. This process generates the perturbed dataset, $\{\tilde{I}_i, P_i\}_{i=1}^N$, where the unedited images $\{I_i\}_{i=1}^N$ serve as pseudo GT. This formulation theoretically addresses the challenges of perturbation modeling and dataset construction. It is important to note that the perturbation ΔI_m is not a fixed additive term, but rather conforms to the inherent editing characteristics of the model $\mathcal{F}$, enabling it to apply scene-specific perturbations during the editing process. Furthermore, the pseudo GT is equivalently substituted for the ideal GT used in inference editing, ensuring that the inconsistencies between them are similar. The model G can then be trained as follows:

$$\arg \min_{\Theta} \sum_{i=1}^N \left| I_i - G(\{\tilde{I}_k, P_k\}_{k=1}^K, \Theta \mid P_k \neq P_i) \right|^2. \quad (8)$$

C. Generate Training Data Pairs

Training the versatile model G in (8) needs ample data. Consequently, we construct vast training data pairs of perturbed scene and pseudo label, $\{I_i + \Delta I_i, I_i\}$, to enable the model to synthesize 3D-consistent novel views while effectively removing the perturbation ΔI_i .

Generate input and target captions: We utilize BLIP [30] and GPT [31] to generate training data pairs. As shown in Fig. 4, we randomly select a scene image into the BLIP to obtain its input caption, such as “a white shoe.” Then, we use GPT to rewrite it into a target caption, such as “a purple slipper.” The four prompts for GPT are: (1) List 100 famous painters; (2) List 50 famous painting schools; (3) List 100 famous paintings; and (4) Replace, add, or delete partial words in the following sentences: X (X is



Fig. 4. Examples of the generated perturbed scene. We utilize the BLIP model [30] and GPT model [31] to obtain the input and target captions required for editing, then add perturbations to the original scene to get the perturbed scene by controlling key parameters of the 2D editing model [27]. In this work, we construct over 480,000 diverse 3D scenes for training our versatile model.

the input caption). For (1), a chosen painter like “Van Gogh” could transform the caption “a blue and white vase” into “Van Gogh painting of a blue and white vase”. Similar procedures apply to (2) and (3). For (4), GPT might get a target caption like “a yellow and white cup.”

Generate perturbations: After obtaining the input and target captions, we can construct perturbed scenes with inconsistency. For each image of a scene, we apply random perturbations by controlling the hyper-parameters in (3).

D. Model Design

The proposed approach differs significantly from existing models designed solely for NVS tasks on clean data. Our method aims to generate consistent novel views from inconsistent, edited source views. Fig. 3 illustrates the novel architecture designed for model G . We design the following two modules to complete 3D editing tasks.

Feature aggregation: Directly editing scenes from multi-views can disrupt the consistency between these views and result in losing information from the original scene. To address this issue and enable the model to capture the disrupted scene information better, we first incorporate multi-view aggregation and integration of unedited image information to render consistent results. Specifically, we use a UNet-like network with the ResNet34 [70] backbone to extract features from both the unedited views I_i and pre-edited views $\tilde{I}_i$. These feature maps are then concatenated and input into subsequent CNN networks to derive the feature maps F_i .

Versatile volume rendering: When estimating the color c and density σ of sampled points along the rays, we use the MLP networks to integrate information from the edited images $\tilde{I}_i$, feature maps F_i , and viewing directions d . For density prediction, we adopt a design of transformer networks [71] inspired by IBRNet [3]. For color estimation, we incorporate additional pixel difference information among source views to capture the extent of editing across different perspectives, aiding the network in improved learning.

The detailed model architecture and the data propagation process are delineated in the supplementary material.

E. Loss Function

We use a training approach similar to that of the versatile NeRF models [3], [29], which involves predicting the target view based on several source views. Unlike these methods, our training data incorporates perturbations. In addition to generating novel views, the training objectives of our model also aim to eliminate perturbations. Thus, beyond the specific design of the network architecture discussed in Section IV-D, we develop two loss functions to remove these perturbations.

Self-view robustness loss: We use two independent predictions, $\mathcal{A}$ and $\mathcal{B}$, in this work. For $\mathcal{A}$, we use source views $\{Src_a^1, Src_a^2, \dots\}$ to predict the target view Tgt_a . For $\mathcal{B}$, we use different source views $\{Src_b^1, Src_b^2, \dots\}$ to predict the same target view Tgt_b . Then, we calculate the L2 loss to ensure consistency between the two predictions:

$$\ell_{self} = \|Tgt_a - Tgt_b\|^2. \quad (9)$$

Neighboring view consistency loss: Empirically, we observe that there are often noticeable texture or color discontinuities between adjacent views when rendering along a smooth camera path. To address this issue, we enforce a smooth transition between adjacent views. Specifically, we slightly perturb the pose corresponding to the target view and generate a neighboring view Tgt_{nbr} based on the perturbed pose. We then project the pixel points from the target view onto the neighboring view using the depth rendered by model G . We further minimize the following loss to reduce image discontinuities caused by changes in the viewing angle:

$$\ell_{nbr} = \|M(Tgt_a - Tgt_{nbr})\|^2 + \|M(Tgt_b - Tgt_{nbr})\|^2, \quad (10)$$

where M refers to a mask that discards some pixels on the target view that are not visible to the neighboring view.

We note that the weights of some sampled points are uniformly distributed along the rays, leading to floating objects in the predicted results and inaccurate depth estimates, which may affect the accuracy of the target view to neighboring view projection. Thus, we introduce an entropy loss [72] for the weights of the sampled points:

$$\ell_{en} = -\sum T_i(1 - \exp(-\sigma_i \delta_i) \log(T_i(1 - \exp(-\sigma_i \delta_i))), \quad (11)$$

where $T_i = \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j)$. σ_i is the density and δ_i denotes the distance between adjacent sampled points.

Total loss: During training, the model cannot render a complete image in a single forward pass due to GPU memory limitations. The loss is computed on the patch level, and the total loss function is as follows:

$$\ell = \ell_{rec} + \lambda_1 \ell_{self} + \lambda_2 \ell_{nbr} + \lambda_3 \ell_{en} + \lambda_4 \ell_{tv}, \quad (12)$$

where ℓ_{rec} is the reconstruction loss obtained from (8), and ℓ_{tv} is the total variation regularization term [73].

F. Content Filter

During the inference stage, applying normal amplitude edits to 3D scene images may lead to poor editing results for certain views. These poor editing results cannot be fully remedied solely through the model G . Thus, we first design a content filter to remove these edited images. The goal is to maintain the degree of editing consistency across various views. To achieve this, we consider the following two situations: (1). A smaller degree of editing implies fewer changes in the *edited image* relative to the *original image* and *input caption*, while a larger discrepancy with the *target text*. (2). Conversely, a higher degree of editing denotes more significant changes in the *edited image* compared to the *original*, bringing it closer to the *target text*.

The evaluation metrics used in the content filter can be measured through the relationship between the original image I_i , edited image $\tilde{I}_i$, original text description C_{in} , and target text description C_{tgt} . Combining evaluations of editing results from previous methods [64], [65], we propose a content filter based on the following four tuples:

$$\begin{cases} (a). \text{ SSIM}(I_i, \tilde{I}_i) \\ (b). \text{ CLIP}(\tilde{I}_i, C_{tgt}) \\ (c). \text{ CLIP}(I_i, \tilde{I}_i) \\ (d). \text{ CLIP}(I_i, \tilde{I}_i) - \text{CLIP}(C_{in}, C_{tgt}) \end{cases} \quad (13)$$

Metrics (a) and (b) assess the image similarity before and after editing. The metric (c) gauges the similarity between the edited image and the target caption, while the metric (d) evaluates the relative offset between the text and the image. These indicators assess the editing results on different dimensions. After this process, only minor perturbations remain, which can be eliminated using the well-trained model G . For all edited images, we compute and sort the four metrics individually. We then discard the top and bottom 10% of outliers, retaining only values close to the mean to reduce the differences among the edited images. We provide the impact of different discard ratios on the editing results in the supplementary materials.

V. EXPERIMENTS AND ANALYSIS

A. Implementation Details

Datasets: The training data for DN2N are sourced from Google Scanned Objects [74], NeRF-Synthetic [1], Spaces [75], and IBRNet-collect [3]. Specifically, Google Scanned Objects is trained at 512×512 resolution, NeRF-Synthetic at 800×800 , Spaces at 800×480 , and IBRNet-collect at 504×378 . The evaluation is conducted on entirely unseen datasets, including LLFF [76] and NeRF-Art [64], as well as outdoor datasets with characteristics entirely distinct from the training data, such as OMMO [77] (drone scene captured from a 360-degree drone perspective) and KITTI [78] (street scene captured from a vehicle perspective). All text prompts presented in the experimental results were excluded from the training process.

Training details: We use the PyTorch framework to implement our method and employ the Adam optimizer [79] with the initial learning rates of $1e-4$ for the CNN and $5e-4$ for the MLP. The training process runs for 300 K steps with a batch size of 500

rays. The initial values for the loss weight in (12) are set as $\lambda_1 = 1e-3$, $\lambda_2 = 1e-3$, $\lambda_3 = 1e-3$, and $\lambda_4 = 2e-3$, respectively. The loss l_{nbr} is computed after the iteration count exceeds 10K. During the training phase, we randomly select a variable number of source views ranging from 6 to 15, while using 15 source views during inference. The number of sampled points on a ray is set to 64.

Metrics: To evaluate the model’s versatility in removing perturbations, we compute the PSNR between the original clean scene and the scene after perturbation removal by the model. We conduct an additional user study to assess the editing results more comprehensively. To evaluate model efficiency, we report the run time for scene editing and model size. We also employ the quantitative CLIP Directional score (CDS) and CLIP Consistency score (CCS) as outlined by Instruct-N2N [65], and show this result in the supplementary materials.

Baselines: We evaluate the proposed DN2N model against the state-of-the-art NeRF methods, including PixelNeRF [80], MVSNeRF [28], IBRNet [3], and Neuray [29]. For 3D scene editing experiments, we draw comparisons with text-driven approaches such as NeRF-Art [64], Instruct-N2N [65], Clip-NeRF [63] and GaussianEditor [10], image-driven style transfer methods like ARF [22] and Learning-to-Style [56], and the DFF [14] method for scene decomposition based on semantics.

We generate training datasets and train our model on 8 V100 GPUs. The data generation takes approximately ten days, while model training requires about six days. The inference is performed on a single V100 GPU.

B. Versatility of DN2N

As shown in Figs. 1 and 5, DN2N supports a wide range of editing capabilities, including modifying scene appearance, altering weather conditions, performing style transfer, and replacing objects. Moreover, even when applied to datasets with characteristics entirely distinct from the training data, DN2N demonstrates remarkable generalization, as illustrated by the autonomous driving scenes in Fig. 5(b) and the 360-degree drone-captured scenes in Fig. 5(c). Notably, our method continues to produce correct editing results even when given unconventional target texts that differ substantially from the input text (see Fig. 5(d)). These results indicate that, by training on a dataset with perturbations generated through slight adjustments of editing parameters, DN2N can successfully generalize during inference to unseen text instructions and scenes, even in the presence of inconsistencies introduced by “normal editing.”

C. Qualitative Results and Comparisons

Text-driven 3D scene editing: We compare our approach with state-of-the-art text-driven editing methods in Fig. 6(a). NeRF-Art uses multiple loss functions to retrain a model when given target words but struggles with complex and precise editing effects. Instruct-N2N and GaussianEditor iteratively edit 2D images and retrain models, consuming significant resources. In contrast, DN2N is directly inferred after giving a text without any intervening training process. The editing results by DN2N

TABLE I
MODEL PERFORMANCE TO RESIST PERTURBATIONS. THE METRIC VALUES REFER TO THE DECREASE IN MODEL PERFORMANCE AFTER INTRODUCING PERTURBATIONS TO THE SCENE. A SMALLER VALUE INDICATES A STRONGER RESISTANCE TO INCONSISTENCY, MAKING THE MODEL MORE SUITABLE FOR 3D SCENE EDITING TASKS.

Dataset	PixelNeRF	MVSNeRF	IBRNet	Neuray	Ours
NeRF-Synthetic	9.53	5.94	6.08	7.6	0.88
DTU	7.76	4.62	4.12	5.89	0.97
LLFF	7.63	4.44	5.12	6.04	1.39

are more realistic and retain more areas unrelated to the target caption, as shown in Fig. 6(a).

Appearance editing: Editing an object’s appearance involves two implicit steps: locating the target area and modifying its appearance. As shown in Fig. 6(b), DN2N accurately identifies and edits the target area without retraining, aligning with the target description. While Clip-NeRF fails to locate the target area precisely. DFF employs DINO or Lseg for accurate localization but requires manual image editing and scene-specific training, making it less efficient than DN2N. Furthermore, both methods require training separate models for each scene, making them less efficient than DN2N.

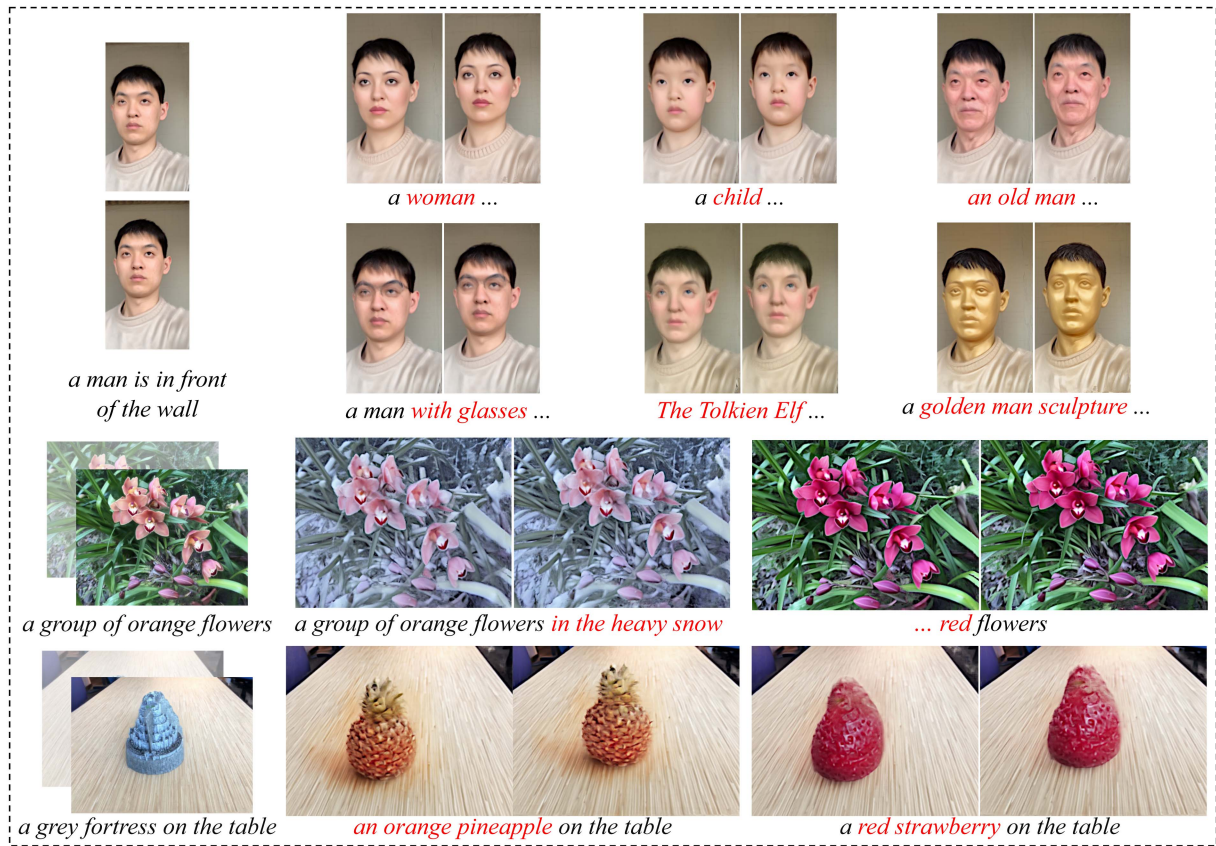
Style transfer: Fig. 6(c) compares DN2N with other style transfer methods. Learning-to-Style mimics reference image colors but misses curved strokes, while ARF overemphasizes strokes, degrading visual quality. In contrast, DN2N synthesizes the scenes by capturing the target style’s color and stroke. Furthermore, both Learning-to-Style and ARF necessitate selecting a reference image first and then training an editing model, making them less efficient and practical than the DN2N.

Geometric editing: We conduct comparative experiments on geometric editing, as shown in Fig. 7, which demonstrate that our method also performs well in this category of edits. It is important to note that Instruct-NeRF2NeRF and Gaussian-Editor require retraining and storing a new model for each edit, whereas our approach is training-free and offers an $O(1)$ model storage advantage.

D. Quantitative Results and Comparisons

Resistance to perturbations: DN2N is designed for scene editing and aims to eliminate the 3D-inconsistent perturbations introduced by 2D editing models. To this end, we evaluate the PSNR degradation after applying editing perturbations across different methods, as reported in Table I. The results show that existing methods suffer significant performance drops due to inconsistencies introduced during multi-view editing. In contrast, DN2N demonstrates much greater stability under these conditions. Moreover, Fig. 8 further illustrates the correctness of the perturbation modeling in DN2N, enabling it to effectively eliminate perturbations introduced by 2D editing models and achieve superior 3D consistency in the editing results.

Model efficiency: Table II shows the merits of DN2N over state-of-the-art methods in run time and model storage. Regarding run time, DN2N does not require training for new scenes or new types of edits, only necessitating inference time for editing, making it more time-efficient than, or comparable to, alternative



(a) Editing results on diverse scenes and editing types.



(b) Editing results on KITTI dataset that is the street view scenes captured by the car camera.



(c) Editing results on the OMMO dataset that is 360° scenes captured by the drone camera.



(d) Editing results on complex target descriptions.

Fig. 5. Versatile editing capabilities of DN2N.

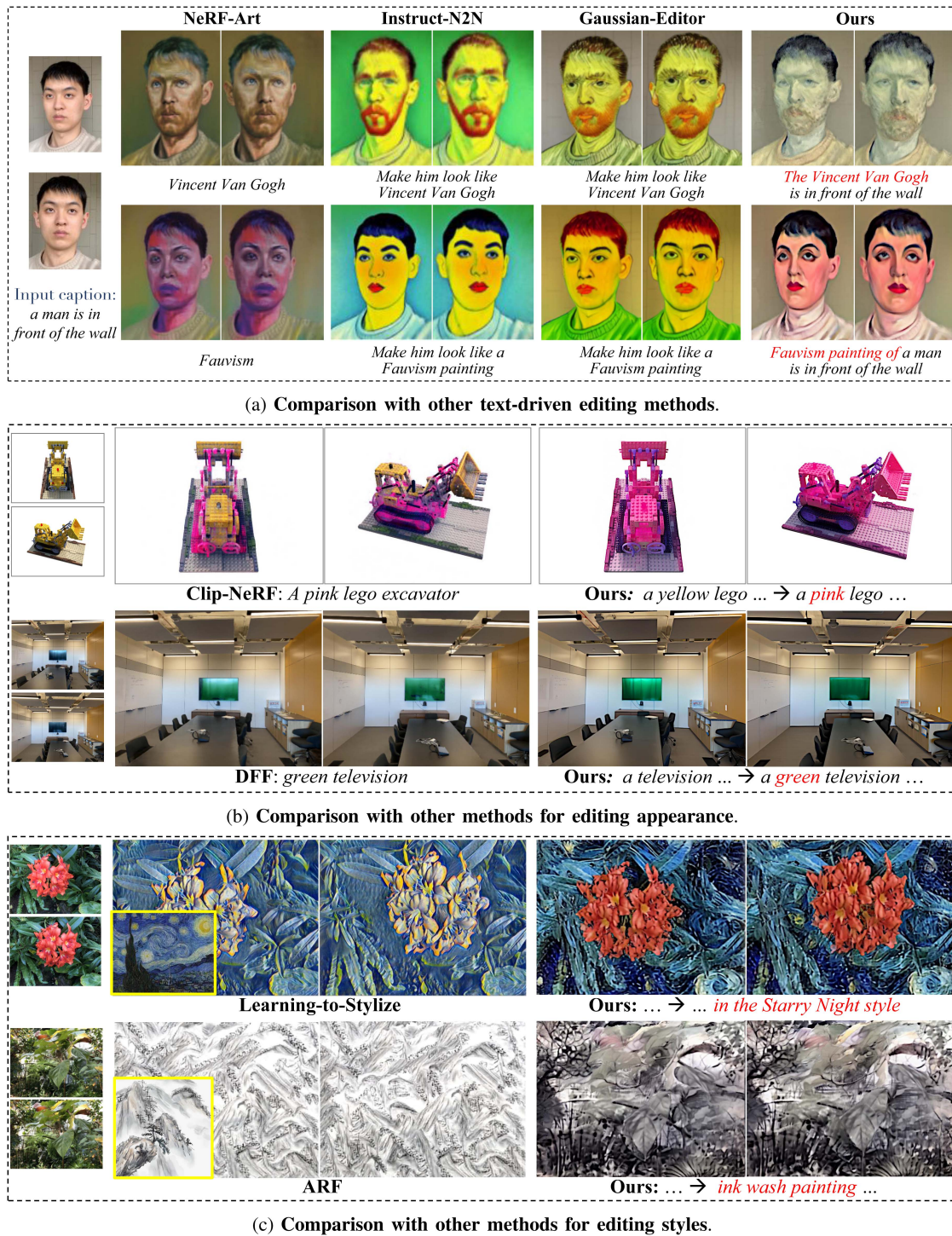


Fig. 6. Comparison with other 3D scene editing methods. DN2N achieves editing results comparable to or surpassing those of previous methods requiring retraining for every edit, while eliminating the need for repeated training iterations.

techniques. Regarding model storage, DN2N can be applied to all scenes and types of edits, and it entails a constant space complexity of $O(1)$ while other methods necessitate the space complexity of $O(n)$. It is important to note that our method inherits a limitation common to feed-forward approaches [3], [28], [29]: inference relies on extracting features from input images, necessitating the storage of source views. However, the advantage is that, compared to methods that require storing

model weights, managing, compressing, and distributing standardized image files is often far more maintainable.

E. User Study

We perform a user study of 50 participants to choose preferred editing results generated by DN2N and other methods. There are 1700 votes based on three evaluation metrics: 3D consistency,



Fig. 7. Comparison for geometric editing. Our approach performs well on geometric editing and requires no retraining, whereas baseline methods necessitate retraining and storing a new model for each individual edit.

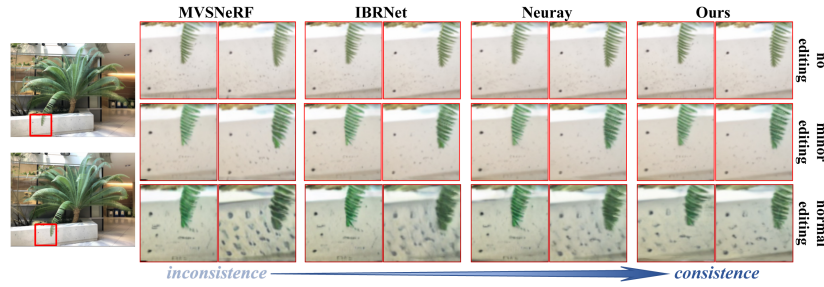


Fig. 8. Comparison of other NeRF models on the fern scenes. The greater the degree of editing, the more inconsistency it brings to the scene. DN2N can generate more consistent images than other methods.

TABLE II

QUANTITATIVE COMPARISON OF TIME AND SIZE FOR EDITING THE FLOWER SCENE IN LLFF DATASET. ‘TC’ AND ‘SC’ ARE TIME AND MODEL STORAGE COMPLEXITY, RESPECTIVELY. DN2N REDUCES THE MODEL’S STORAGE COMPLEXITY FROM $O(n)$ TO $O(1)$.

Methods	Time (hours)			TC	Size(MB)	SC
	train	edit	total			
ARF	0.36	0.05	0.41	$O(n)$	558	$O(n)$
DFF	0.34	0.09	0.43	$O(n)$	144	$O(n)$
Clip-NeRF	8.74	5.82	14.56	$O(n)$	29.4	$O(n)$
NeRF-Art	25.76	13.01	38.77	$O(n)$	18.3	$O(n)$
Instruct-N2N	0.32	1.03	1.35	$O(n)$	484	$O(n)$
GaussianEditor	0.25	0.12	0.37	$O(n)$	310	$O(n)$
DN2N (Ours)	0	0.37	0.37	$O(n)$	103	$O(1)$

preservation of the original scene content, and faithfulness to the text description. Fig. 9 shows that DN2N is overall favored in these evaluation metrics. For more details on the user study, please refer to the supplementary materials.

F. Visualization of Geometric Information

We visualize the depth maps before and after scene editing in Fig. 10. This is divided into two cases: first, when only the appearance is edited, the depth maps before and after exhibit

negligible differences. Second, for edits altering the geometry of objects, the depth maps correspondingly exhibit changes as well, validating the 3D awareness of our editing approach.

G. Ablation Studies and Analyses

Visual quality: We demonstrate the contribution of each component of DN2N in Fig. 11. Overall, the absence content filter results in significant image distortion. This can be attributed to the fact that applying normal amplitude edits to 3D scene images may produce extremely poor editing results for a few views, making it difficult for the model to generate consistent results without the content filter. In addition, omitting the data perturbation process or removing the cross-view regularization terms during training would significantly affect the model’s performance on edited results. This is primarily because the model needs to eliminate inconsistency from the perturbed data constructed, and constraints on the training process are required to foster the emergence of this capability. Furthermore, we evaluate DN2N’s ability to remove perturbations by applying several random disturbances to the scene depicted in Fig. 11. The PSNR results are presented in Table III. These results show that training the versatile model with perturbed data is crucial, and enforcing cross-view consistency can also enhance model performance.

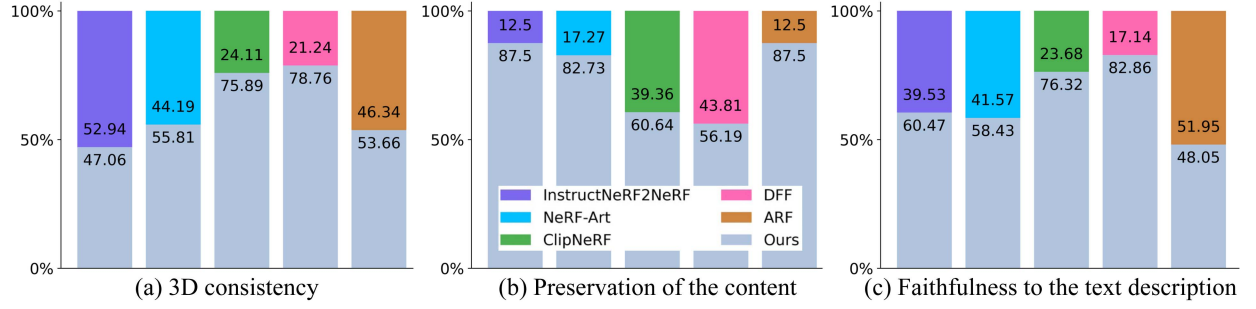


Fig. 9. User study. The overall performance of DN2N surpasses previous methods in three evaluation criteria.

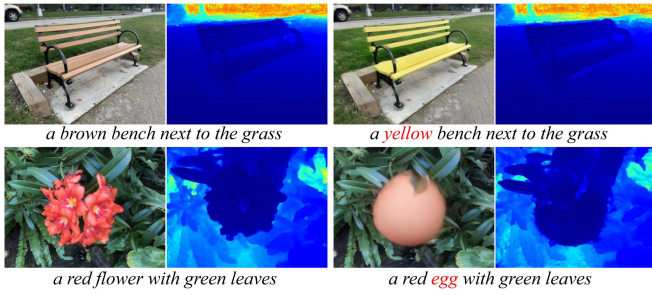


Fig. 10. Visualization of geometric information. When only editing the appearance (first row), the depth maps exhibit minor changes. For edits altering the geometry of objects (second row), the depth maps correspondingly exhibit changes as well, validating the 3D awareness of our approach.

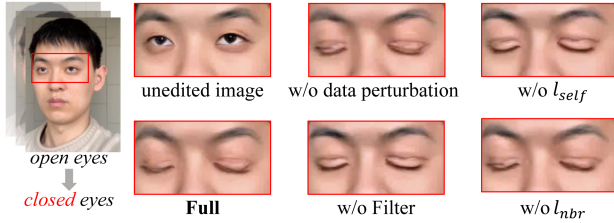


Fig. 11. Ablation studies for key components in DN2N. We recommend referring to the *video demo* in our supplementary materials to observe the consistency differences.

TABLE III
PSNR RESULTS OF ABLATION STUDIES FOR DATA PERTURBATION (DP), l_{self} AND l_{nbr} . EXPERIMENTS ARE CONDUCTED BY APPLYING FOUR RANDOM MINOR PERTURBATIONS (ID 1 ~ 4) TO THE SCENE EDITING IN FIG. 11.

Exp. ID	w/o DP	w/o l_{self}	w/o l_{nbr}	Full
1	17.24	19.89	21.05	21.76
2	18.98	21.5	21.92	23.11
3	18.73	21.16	22.39	24.21
4	18.97	20.53	22.19	22.27

3D consistency: We show ablations on consistency between different novel views. Without perturbed data, we observe significant color and gloss discontinuities between different novel views in the editing results, as shown in Fig. 12. Normal editing may cause substantial consistency disruption in 2D editing results at the inference stage, which the content filter can mitigate. Removing this filtering process leads to noticeable inconsistent

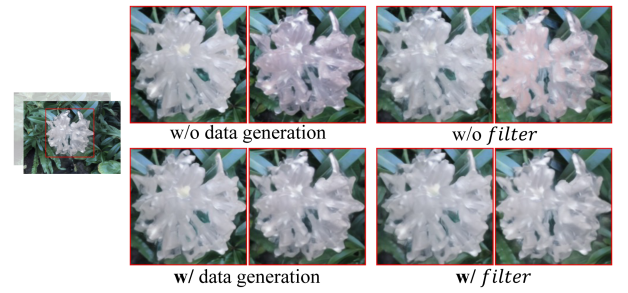


Fig. 12. Effect of data generation and content filter. Without the data generation and content filter, the rendered novel views may contain inconsistent glossiness and color.

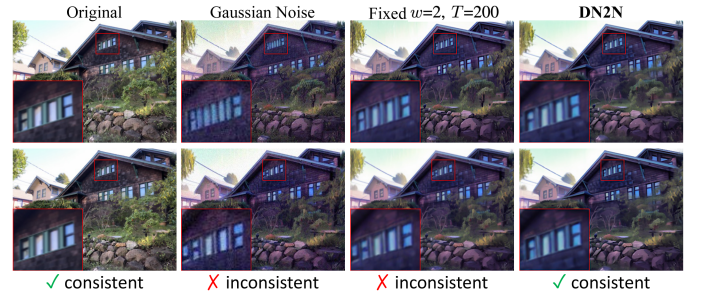


Fig. 13. Effect of different types of noise perturbation. We use diverse noise types to construct perturbed scenes for training and then use these models to edit the house into a cyberpunk style. DN2N performs favorably in eliminating inconsistency compared to random Gaussian or fixed-parameter noise type.

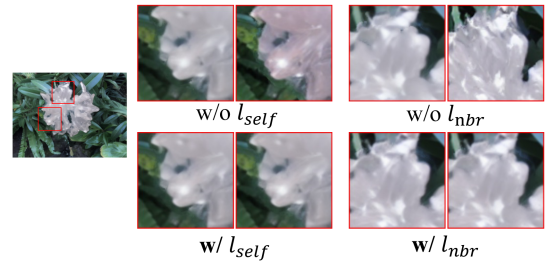


Fig. 14. Effect of the l_{self} and l_{nbr} . Without l_{self} , the 3D consistency of the rendering results under different source views is poor. Without l_{nbr} , generated images contain inconsistent color or brightness among different neighboring views.



Fig. 15. Failed editing cases. The editing ability is limited by the 2D multimodal models. In addition, the text format needs to follow the form required by the 2D model.

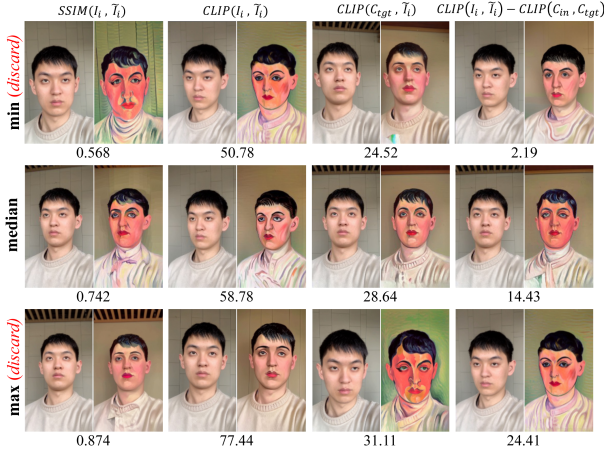


Fig. 16. Visualize the editing results of the 2D editing model, as well as the maximum, minimum, and median values for each metric of the content filter. The images that correspond to the minimum and maximum values often exhibit either low or excessive editing degrees. By discarding them, we can enhance the 3D consistency among the remaining images.

results. Fig. 14 shows that generated images contain noticeable color or brightness inconsistency when removing the l_{self} and l_{nbr} regularization terms.

Analysis of noise perturbation: We consider the inconsistency arising from multi-view editing to be inherent to the characteristics of 2D editing models. Consequently, we construct training data pairs exhibiting inconsistency by manipulating two crucial parameters in the editing process: guidance scale w and sample steps T , as detailed in Section IV. We empirically demonstrate that the DN2N trained on such data can eliminate inconsistency. To further validate this, we reconstruct the training data using two types of noise and trained corresponding models. The first approach simulates post-editing inconsistency using Gaussian random noise, while the second applies perturbations to the scene using a 2D editing model with fixed w and T values. The experimental results in Fig. 13 show that models trained using these two methods still contain significant 3D inconsistent results after editing. In contrast, DN2N effectively mitigated the inconsistency issue, showing that the noise perturbations we introduced closely approximate the real perturbations, as theoretically derived in Section IV-B.

Analysis of content filter: In Fig. 16, we present the maximum, minimum, and median values of the filtering criterion defined in

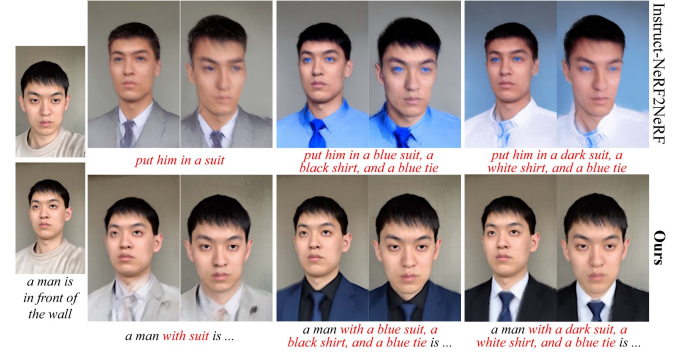


Fig. 17. The sensitivity of our method and Instruct-NeRF2NeRF to varying prompts. Our method performs better when the prompt provides precise descriptions, while Instruct-NeRF2NeRF lacks the capacity to handle complex prompts effectively.

(13), along with their corresponding original and edited images. It demonstrates that the metrics employed in the content filter effectively identify edits that are either insufficient or overly aggressive. A discard ratio that is too small fails to remove inadequate edits, while an excessively large ratio may erroneously eliminate satisfactory results. As an engineering trade-off, a discard ratio of 10% effectively removes these extreme editing cases, thereby facilitating superior 3D editing performance. For sensitivity analysis of the parameter discard ratio, please refer to the supplementary materials.

Sensitivity to varying prompts: We further analyze the sensitivity of different methods to diverse prompts. As shown in Fig. 17, both our method and Instruct-NeRF2NeRF exhibit a certain degree of prompt sensitivity. Some types of prompts may lead to suboptimal editing results, manifested as artifacts or blurriness, but overall, our method outperforms Instruct-NeRF2NeRF. In general, our approach achieves better results with more detailed prompts, likely because precise descriptions enable the 2D editing model to produce more consistent multi-view edits.

H. Limitation and Future Work

Experimental results show that DN2N exhibits strong generalization capability in scene editing tasks without requiring the per-scene or per-edit retraining that is necessary for existing methods. However, several limitations remain. First, employing a 2D multimodal model for 3D scene editing inherently inherits

the constraints of the underlying 2D approach. This limitation is common across methods in this research line. For instance, CLIP-NeRF [63] is constrained by CLIP [25], while Instruct-N2N [65] and GaussianEditor [10] are restricted by Instruct-Pix2Pix [26]. Similarly, our method is limited by Null-text [27]. As shown in Figs. 15(a) and 17, a single 2D editing model cannot robustly handle the full diversity of prompts, which can result in degraded editing performance. In addition, when significant textural or geometric conflicts arise among the input 2D edited views, our network tends to produce an “average” of these conflicting inputs to satisfy multi-view consistency constraints, leading to blurriness. Potential avenues for improvement include enhancing the consistency of upstream 2D edits, appropriately reducing the number of reference views, and incorporating prior models to refine the blurred outputs. Moreover, although our method eliminates the need for retraining, its rendering efficiency remains limited. However, the core idea of DN2N is highly extensible and could be further enhanced in the future by integrating efficient, generalizable 3D representations, such as 3DGS variants [81], [82], [83], [84], to improve our method’s rendering performance.

VI. CONCLUSION

In this work, we propose a text-driven method, DN2N, for editing 3D scenes that exhibits strong versatility and enables realistic novel view editing without additional training for each modification task. DN2N leverages existing 2D editing methods to perform initial editing of 3D scenes based on textual prompts. We then filter out poorly edited images, treating the remaining inconsistency as perturbations on top of consistently edited results. We present a NeRF model to eliminate such perturbation and provide effective training approaches, including creating sufficient training data and strengthening cross-view robustness. Extensive experimental results demonstrate DN2N makes editing easier, supports multiple editing capabilities, and eliminates the need for training on new scenes.

REFERENCES

- [1] B. Mildenhall, P.P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “NeRF: Representing scenes as neural radiance fields for view synthesis,” in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 405–421.
- [2] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” 2022, *arXiv:2201.05989*.
- [3] Q. Wang et al., “IBRNet: Learning multi-view image-based rendering,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4690–4699.
- [4] G. Liao, K. Zhou, Z. Bao, K. Liu, and Q. Li, “Ov-NeRF: Open-vocabulary neural radiance fields with vision and language foundation models for 3D semantic understanding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 12, pp. 12923–12936, Dec. 2024.
- [5] Z. Sheng, F. Liu, M. Liu, F. Zheng, and L. Nie, “Open-set synthesis for free-viewpoint human body reenactment of novel poses,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 12, pp. 12676–12691, Dec. 2024.
- [6] J. Ding et al., “Ray reordering for hardware-accelerated neural volume rendering,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 11, pp. 11413–11422, Nov. 2024.
- [7] Z. Liu, J. Su, G. Cai, Y. Chen, B. Zeng, and Z. Wang, “GeoRGS: Geometric regularization for real-time novel view synthesis from sparse inputs,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 12, pp. 13113–13126, Dec. 2024.
- [8] X. Zhu et al., “DFIE3D: 3D-aware disentangled face inversion and editing via facial-contrastive learning,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 9, pp. 8310–8326, Sep. 2024.
- [9] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3D Gaussian splatting for real-time radiance field rendering,” *ACM Trans. Graph.*, vol. 42, no. 4, pp. 1–14, 2023.
- [10] J. Fang, J. Wang, X. Zhang, L. Xie, and Q. Tian, “GaussianEditor: Editing 3D Gaussians delicately with text instructions,” 2023, *arXiv:2311.16037*.
- [11] T. Liu et al., “Fast generalizable Gaussian splatting reconstruction from multi-view stereo,” 2024, *arXiv:2405.12218*.
- [12] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park, “Compact 3D Gaussian representation for radiance field,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 21719–21728.
- [13] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, “NeRF in the wild: Neural radiance fields for unconstrained photo collections,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7219–7219.
- [14] S. Kobayashi, E. Matsumoto, and V. Sitzmann, “Decomposing NeRF for editing via feature field distillation,” 2022, *arXiv:2205.15585*.
- [15] J. Tang, X. Chen, J. Wang, and G. Zeng, “Compressible-composable NeRF via rank-residual decomposition,” 2022, *arXiv:2205.14870*.
- [16] M. Tancik et al., “Block-NeRF: Scalable large scene neural view synthesis,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8248–8258.
- [17] Y. Li, Z.-H. Lin, D. Forsyth, J.-B. Huang, and S. Wang, “ClimateNeRF: Physically-based neural rendering for extreme climate synthesis,” 2022, *arXiv:2211.13226*.
- [18] S. Fang, W. Xu, H. Wang, Y. Yang, Y. Wang, and S. Zhou, “One is all: Bridging the gap between neural radiance fields architectures with progressive volume distillation,” 2022, *arXiv:2211.15977*.
- [19] S. Fang et al., “PVD-AL: Progressive volume distillation with active learning for efficient conversion between different NeRF architectures,” 2023, *arXiv:2304.04012*.
- [20] Y.-H. Huang, Y. He, Y.-J. Yuan, Y.-K. Lai, and L. Gao, “StylizedNeRF: Consistent 3D scene stylization as stylized NeRF via 2D-3D mutual learning,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 18342–18352.
- [21] J. Gu, L. Liu, P. Wang, and C. Theobalt, “StyleNeRF: A style-based 3D aware generator for high-resolution image synthesis,” in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 1–25.
- [22] K. Zhang et al., “ARF: Artistic radiance fields,” in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 717–733.
- [23] Y. Chen et al., “UPST-NeRF: Universal photorealistic style transfer of neural radiance fields for 3D scene,” *IEEE Trans. Vis. Comput. Graph.*, vol. 31, no. 4, pp. 2045–2057, Apr. 2025.
- [24] T. Nguyen-Phuoc, F. Liu, and L. Xiao, “SNeRF: Stylized neural implicit representations for 3D scenes,” 2022, *arXiv:2207.02363*.
- [25] A. Radford et al., “Learning transferable visual models from natural language supervision,” in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.
- [26] T. Brooks, A. Holynski, and A. A. Efros, “InstructPix2Pix: Learning to follow image editing instructions,” 2022, *arXiv:2211.09800*.
- [27] R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or, “Null-text inversion for editing real images using guided diffusion models,” 2022, *arXiv:2211.09794*.
- [28] A. Chen et al., “MvsNeRF: Fast generalizable radiance field reconstruction from multi-view stereo,” in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 14124–14133.
- [29] Y. Liu et al., “Neural rays for occlusion-aware image-based rendering,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 7824–7833.
- [30] J. Li, D. Li, S. Savarese, and S. Hoi, “BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models,” 2023, *arXiv:2301.12597*.
- [31] T. Brown et al., “Language models are few-shot learners,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 1877–1901.
- [32] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, “Plenotrees for real-time rendering of neural radiance fields,” in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 5752–5761.
- [33] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P.P. Srinivasan, “Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields,” in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 5855–5864.
- [34] K. Zhang, G. Riegler, N. Snively, and V. Koltun, “NeRF: Analyzing and improving neural radiance fields,” 2020, *arXiv:2010.07492*.

- [35] C.-H. Lin, W.-C. Ma, A. Torralba, and S. Lucey, "BARF: Bundle-adjusting neural radiance fields," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 5741–5751.
- [36] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "TensorRF: Tensorial radiance fields," 2022, *arXiv:2203.09517*.
- [37] M. Ye, M. Danelljan, F. Yu, and L. Ke, "Gaussian grouping: Segment and edit anything in 3D scenes," 2023, *arXiv:2312.00732*.
- [38] T. Wu et al., "Recent advances in 3D Gaussian splatting," *Comput. Vis. Media*, pp. 1–30, 2024.
- [39] G. Chen and W. Wang, "A survey on 3D Gaussian splatting," 2024, *arXiv:2401.03890*.
- [40] B. Fei, J. Xu, R. Zhang, Q. Zhou, W. Yang, and Y. He, "3D Gaussian splatting as new ERA: A survey," *IEEE Trans. Vis. Comput. Graph.*, vol. 31, no. 8, pp. 4429–4449, Aug. 2025.
- [41] G. Wu et al., "4D Gaussian splatting for real-time dynamic scene rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 20310–20320.
- [42] J. Z. Wu et al., "Difix3D: Improving 3D reconstructions with single-step diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2025, pp. 26024–26035.
- [43] A. Trevithick et al., "SimVS: Simulating world inconsistencies for robust view synthesis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2025, pp. 16464–16474.
- [44] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2256–2265.
- [45] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 8780–8794.
- [46] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 6840–6851.
- [47] C. Saharia et al., "Photorealistic text-to-image diffusion models with deep language understanding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 36479–36494.
- [48] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022, *arXiv:2204.06125*.
- [49] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10684–10695.
- [50] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2020, *arXiv:2010.02502*.
- [51] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," 2020, *arXiv:2011.13456*.
- [52] A. Q. Nichol et al., "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 16784–16804.
- [53] B. Kavar et al., "Imagic: Text-based real image editing with diffusion models," 2022, *arXiv:2210.09276*.
- [54] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or, "Prompt-to-prompt image editing with cross attention control," 2022, *arXiv:2208.01626*.
- [55] G. Parmar, K. K. Singh, R. Zhang, Y. Li, J. Lu, and J.-Y. Zhu, "Zero-shot image-to-image translation," 2023, *arXiv:2302.03027*.
- [56] H.-P. Huang, H.-Y. Tseng, S. Saini, M. Singh, and M.-H. Yang, "Learning to stylize novel views," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 13869–13878.
- [57] F. Mu, J. Wang, Y. Wu, and Y. Li, "3D photo stylization: Learning to generate stylized novel views from a single image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16273–16282.
- [58] Q.-Y. Zhou and V. Koltun, "Color map optimization for 3D reconstruction with consumer depth cameras," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 1–10, 2014.
- [59] L. Höllein, J. Johnson, and M. Nießner, "StyleMesh: Style transfer for indoor 3D scene reconstructions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6198–6208.
- [60] F. Han, S. Ye, M. He, M. Chai, and J. Liao, "Exemplar-based 3D portrait stylization," *IEEE Trans. Vis. Comput. Graph.*, vol. 29, no. 2, pp. 1371–1383, 2021.
- [61] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. Lensch, "NeRD: Neural reflectance decomposition from image collections," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 12684–12694.
- [62] M. Boss, V. Jampani, R. Braun, C. Liu, J. Barron, and H. Lensch, "Neural-PIL: Neural pre-integrated lighting for reflectance decomposition," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 10691–10704.
- [63] C. Wang, M. Chai, M. He, D. Chen, and J. Liao, "CLIP-NeRF: Text-and-image driven manipulation of neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 3835–3844.
- [64] C. Wang, R. Jiang, M. Chai, M. He, D. Chen, and J. Liao, "NeRF-Art: Text-driven neural radiance fields stylization," 2022, *arXiv:2212.08070*.
- [65] A. Haque, M. Tancik, A. A. Efros, A. Holynski, and A. Kanazawa, "Instruct-NeRF2NeRF: Editing 3D scenes with instructions," 2023, *arXiv:2303.12789*.
- [66] Y. Li et al., "Focaldreamer: Text-driven 3D editing via focal-fusion assembly," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 3279–3287.
- [67] J. Zhuang, D. Kang, Y.-P. Cao, G. Li, L. Lin, and Y. Shan, "Tip-editor: An accurate 3D editor following both text-prompts and image-prompts," *ACM Trans. Graph.*, vol. 43, no. 4, pp. 1–12, 2024.
- [68] M. Chen, I. Laina, and A. Vedaldi, "DGE: Direct Gaussian 3D editing by consistent multi-view editing," in *Proc. Int. Conf. Comput. Vis.*, 2024, pp. 74–92.
- [69] J. Ho and T. Salimans, "Classifier-free diffusion guidance," 2022, *arXiv:2207.12598*.
- [70] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [71] A. Vaswani, "Attention is all you need," 2017, *arXiv:1706.03762*.
- [72] M. Kim, S. Seo, and B. Han, "InfoNeRF: Ray entropy minimization for few-shot neural volume rendering," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12912–12921.
- [73] L. I. Rudin and S. Osher, "Total variation based image restoration with free local constraints," in *Proc. IEEE Int. Conf. Image Process.*, 1994, pp. 31–35.
- [74] L. Downs et al., "Google scanned objects: A high-quality dataset of 3D scanned household items," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 2553–2560.
- [75] J. Flynn et al., "DeepView: View synthesis with learned gradient descent," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2367–2376.
- [76] B. Mildenhall et al., "Local light field fusion: Practical view synthesis with prescriptive sampling guidelines," *ACM Trans. Graph.*, vol. 38, no. 4, pp. 1–14, 2019.
- [77] C. Lu, F. Yin, X. Chen, T. Chen, G. Yu, and J. Fan, "A large-scale outdoor multi-modal dataset and benchmark for novel view synthesis and implicit scene reconstruction," 2023, *arXiv:2301.06782*.
- [78] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Res. Rev.*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [79] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [80] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelNeRF: Neural radiance fields from one or few images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4578–4587.
- [81] Y. Chen, Q. Wu, M. Li, W. Lin, M. Harandi, and J. Cai, "Fast feedforward 3D Gaussian splatting compression," 2024, *arXiv:2410.08017*.
- [82] C. Zhao, X. Huang, K. Yang, X. Wang, and Q. Wang, "Generalizable 3D Gaussian splatting for novel view synthesis," *Pattern Recognit.*, vol. 161, 2025, Art. no. 111271.
- [83] B. Ye, B. Chen, H. Xu, D. Barath, and M. Pollefeys, "Yonosplat: You only need one model for feedforward 3D Gaussian splatting," 2025, *arXiv:2511.07321*.
- [84] L. Jiang et al., "AnySplat: Feed-forward 3D Gaussian splatting from unconstrained views," *ACM Trans. Graph.*, vol. 44, no. 6, pp. 1–16, 2025.



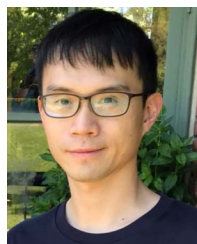
Shuangkang Fang is currently working toward the PhD degree with Beihang University. From 2020 to 2022, he worked full-time as a researcher with Megvii Inc., mainly focused on the development and deployment of various vision models. Since 2022, he has been an intern at StepFun Inc., centered on the representation and manipulation of 3D scenes. His work now aims to enhance the practicality of 3D scene editing technology by reducing model complexity, improving editing speed, and enhancing interactivity.



Yufeng Wang received the BS degree in communication engineering from Northwestern Polytechnical University, Xi'an, China, in 2015, and the PhD degree in Information and Communication System from Beihang University, Beijing, China, in 2021. He is an associate researcher with the Institute of Unmanned System, Beihang University. His current research interests include computer vision, machine learning, and their applications in remote sensing.



Yi Yang is received the graduation degree from Tsinghua University, in 2015 and the MEng degree from the Department of Computer Science and Technology. He has won gold medals in several world competitions, including the IOI and ISC. He is currently a senior Researcher at Megvii Inc. His research interests include image generation, deep learning compilers, and computer architecture.



Yi-Hsuan Tsai is received the PhD with Prof Ming-Hsuan Yang from the University of California, Merced. is the co-founder and CTO with Atmanity. Before this, he was a research scientist and tech lead manager with Google, working on edge AI for perception and scene understanding. He was the director of AI at Phiar Technologies, leading the AI team to improve real-world AR navigation, and was a senior researcher with NEC Laboratories America, working on computer vision and deep learning.



Shuchang Zhou is received the graduation from Tsinghua University, in 2004, the master's degree (National University of Singapore), in 2005, and a doctorate degree (Chinese Academy of Sciences), in 2015. He is now the Principal Scientist with Megvii Inc. Before joining Megvii, he worked with Google Research and the Chinese Academy of Sciences, focusing on large-scale machine learning, mathematical optimization, and Computer Architecture research.



Wenrui Ding (Member, IEEE) received the PhD degree in electrical and information engineering from Beihang University, China. She is currently in charge of the Intelligent Information Research Center with the Institute of Unmanned System, Beihang University. Her current research interests include image and 3D scene processing, pattern recognition, and command and control of the UAV system.



Ming-Hsuan Yang (Fellow, IEEE) is a professor in Electrical Engineering and Computer Science with the University of California at Merced, an adjunct professor with Yonsei University, and a Research Scientist with Google DeepMind. Yang received several paper awards, including Best Paper at ICML 2024, Longuet-Higgins Prize at CVPR 2023, Best Paper Honorable Mention at CVPR 2018, and Best Paper Honorable Mention at UIST 2017. He received an NSF CAREER Award in 2012 and a Google Faculty Award in 2009. He is a fellow of the ACM, and AAAI.