ProteusNeRF: Fast Lightweight NeRF Editing using 3D-Aware Image Context

Binglun Wang **Niladri Shekhar Dutt** **Niloy J. Mitra**

{binglun.wang.22, niladri.dutt.22}@ucl.ac.uk

University College London

Figure 1: We present ProteusNeRF, a fast and lightweight framework for interactive editing of NeRF assets via existing image manipulation tools. This is enabled by a novel 3D-aware image context that allows linking edits across multiple views. We show (original and novel views) using text-guided edits. These edits take 10-70 seconds. Please see the supplemental videos.

Abstract

Neural Radiance Fields (NeRFs) have recently emerged as a popular option for photo-realistic object capture due to their ability to faithfully capture high-fidelity volumetric content even from handheld video input. Although much research has been devoted to efficient optimization leading to real-time training and rendering, options for interactive editing NeRFs remain limited. We present a very simple but effective neural network architecture that is fast and efficient while maintaining a low memory footprint. This architecture can be incrementally guided through user-friendly image-based edits. Our representation allows straightforward object selection via semantic feature distillation at the training stage. More importantly, we propose a local 3D-aware image context to facilitate view-consistent image editing that can then be distilled into fine-tuned NeRFs, via geometric and appearance adjustments. We evaluate our setup on a variety of examples to demonstrate appearance and geometric edits and report 10-30× speedup over concurrent work focusing on text-guided NeRF editing. Video results can be seen on our project webpage at https://proteusnerf.github.io.

CCS Concepts

• ***Computing methodologies*** → ***Volumetric models;*** *Graphics systems and interfaces;*

1. Introduction

Neural Radiance Fields (NeRFs) [MST*20] have rapidly emerged as one of the most popular volumetric representations for casual capture of 3D objects. Benefiting from a significant body of improvements to the original formulation, it is now possible to optimize NeRFs in a matter of minutes and generate photo-realistic novel views at interactive framerates.

In this work, we focus on interactively editing such NeRF assets while preserving their original volumetric representa-

tion (note that we do not focus on approaches where one first distills NeRFs into textured surface meshes). A good editing system should be simple to use, expressive, light-weight, and encourage interactive manipulation. One approach is to enclose NeRF assets into volumetric cages [XH22] and then edit the underlying volumes using cage-based deformation setups. NeRFshop [JKK*23] presents an impressive realization of this workflow where users select and prescribe (deformation) handles in 3D. However, unlike cage based methods which focus on editing the geometry of the NeRFs, we focus on image-based workflows, primarily targeting appearance changes.

Inspired by the recent success of large text or image-conditioned generative models [RBL*22, RDN*22] that directly produce NeRF assets [MRP*23, PJBM22, JMB*22], we investigate the feasibility of an entirely image-based NeRF editing framework. In our setup, users only interact with assets using image-based interfaces for selection and editing. This allows users to benefit from existing image editing tools, both traditional and recent generative methods, without having to learn a new editing setup.

However, to realize the above goal, we first need to solve a few problems: (i) enable object selection, (ii) perform synchronized multi-view image edits, and (iii) update the NeRF assets. We demonstrate that achieving all the outlined problems is possible with feature distillation, a very simple adaptation to existing architectures, and a novel 3D-aware image context, respectively. Technically, once objects are selected, our method alternates between two phases: (a) using the pre-trained NeRF to produce 3D-aware image context that can be edited using existing image manipulation frameworks – traditional or generative, and (b) interleaving between geometric and appearance updates to distill the image edits into an edited NeRF asset. The resultant edits are lightweight (30-50kB/edit) and fast (10 seconds-70 seconds/edit) to realize. We encode the edits as residual updates, which being localized and lightweight can be stored as edit tokens. Such edits can be enabled or disabled in a post-edit setup, akin to layered edit updates in traditional image editing workflows.

In summary, we introduce a simple, fast, and lightweight NeRF editing setup. We demonstrate the effectiveness of the proposed framework on a diverse set of edit scenarios. In our experiments, we achieve 10-30× speedup over concurrent generative NeRF edit setups [HTE*23].

2. Related Work

Novel view synthesis. Neural Radiance Fields (NeRFs) [MST*20] have been highly successful at generating photo-realistic novel views of a 3D scene by optimizing an object-specific MLP to simultaneously model geometry and appearance. However, training a global MLP as a NeRF representation is a slow process and therefore several improvements have been proposed [GKJ*21, MESK22, HSM*21, YLT*21]. For example, ReluFields [KRWM22] use a uniform grid, while Plenoxels [FTC*22] and DVGO [SSC22] utilize a sparse voxel grid for 3D scene reconstruction using distributed and localized representations. More relevant to ours, structured representation in the form of collection of 2D representations have been proposed to enable efficient storage. For example, triplane representations such as K-planes [FKMW*23], EG3D [CLC*22], and TensoRF [CXG*22], having fewer variables to optimize (i.e., storage requirement being quadratic versus cubic in grid resolution), and can achieve high quality results within minutes.

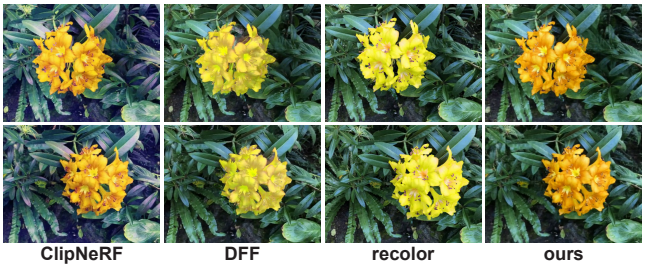

Figure 2: Visual comparison of color editing. CLIP-NeRF [WCH*22] sees color bleeding into the global scene, and DFF [KMS22] shows undesirable color changes in the pistil and unnatural color gradient. Our approach matches the impressive results of RecolorNeRF [GWHD23], while offering a more intuitive and flexible framework. Although RecolorNeRF is faster than other methods, taking about 2-3 minutes to train, ours (ProteusNeRF) takes 10 seconds.

Image editing. In traditional image manipulation, decades of mature research exists behind image editing tools (e.g., Gimp, Photoshop) that can perform operations such as tone mapping, color, contrast, hue changes, etc. In contract, recent breakthroughs in generative models such as VAEs [KW13, HGPR20], GANs [GPAM*20, WSW21], and diffusion models [HJA20, CHIS23] have led to remarkable results in image editing by utilizing image priors distilled from large image collections. For instance, DragGAN [PTL*23] enables image deformations by allowing a user to move a set of handle points towards target points, while InsturctPix2Pix [BHE23] and ControlNet [ZA23] allow image editing using simple text prompts.

Editing NeRFs. Recent advances have primarily focused on text-to-3D conditional generation [MRP*23, PJBM22, JMB*22] or artistic stylization of existing NeRF assets [NPLX22, CTT*22, HHY*22, HTS*21]. Methods which fo-

Table 1: **Comparison wrt related/concurrent works.** Limited options are available to explore geometric and appearance edits of NeRFs interactively (Fast). Ours strikes a balance between being expressive and lightweight, while being able to make limited geometric changes (Geo.), add small geometric details (Add.), delete selected objects (Del.), or make larger appearance changes (App.).

Method	App.	Geo.	Del.	Add	Fast
PaletteNeRF [KLB*23]	✔	✘	✘	✘	✘
RecolorNeRF [GWHD23]	✔	✘	✘	✘	✔
ICE-NeRF [LK23]	✔	✘	✘	✘	✔
DreamEditor [ZWL*23]	✔	✔	✘	✔	✘
CLIPNeRF [WCH*22]	✔	✔	✘	✘	✘
NeRFshop [JKK*23]	✘	✔	✔	✔	✘
InstructN2N. [HTE*23]	✔	✔	✘	✔	✘
DFF [KMS22]	✔	✔	✔	✔	✘
InpaintNeRF360 [WZAS23]	✔	✘	✔	✘	✘
ProteusNeRF (Ours)	✔	✔	✔	✔	✔

cus on directly generating 3D content from text lack fine-grained control on the generated scene and hence synthesize arbitrary scenes, which does not help to make changes to an existing or captured scene. Stylization methods such as CLIP-NeRF [WCH*22], NeRF-Art [WJC*22], and Blending NeRF [GAL23] edit a scene by optimizing global stylistic similarity of reconstructed views and an edit text prompt in the CLIP latent space [RKH*21]. These methods are restricted to making changes in the global scene, therefore, to enable local edits, Distilled Feature Fields [KMS22] and Neural Feature Fusion Fields [TLLV22a] propose to distill features from large scale 2D pre-trained models into 3D to perform selective editing of regions. DreamEditor [ZWL*23] first distills the NeRF into a mesh-based field and then uses score distillation [PJBM22] to optimize any local edit. PaletteNeRF [KLB*23], RecolorNeRF [GWHD23], and ICE-NeRF [LK23] can produce color changes by decomposing a scene into multiple color palettes but the editing landscape is very limited. Moreover, the color based selection is not very robust compared to feature based selection and can hence produce unwanted recoloring.

More closely related to ours, in order to enable a more intuitive interface for editing, are the concurrent works of Inpaint-NeRF360 [WZAS23] and InstructNeRF2NeRF [HTE*23]. They propose to use natural language as instruction to guide the editing process. InpaintNeRF360 [WZAS23] updates the NeRF scene with inpainted images using segmentation masks obtained from point-based prompts and accurate bounding boxes using depth information from the pre-trained NeRF. InstructNeRF2NeRF [HTE*23] iteratively updates the dataset by modifying the rendered images from the pre-trained NeRF using InstructPix2Pix [BHE23]. To facilitate volumetric (geometric)) editing, Deforming-NeRF [XH22] first encloses the foreground object in a cage and then deforms it by maneuvering the vertices of the cage. NeRFshop [JKK*23] improves this paradigm to enable interactive object selection using scribbles.

Although existing methods produce good edited results, to maintain multi-view consistency, re-training the NeRF remains a computationally expensive problem, taking 30 minutes to 2 hours per edit. Taking inspiration from TextMesh [TMT*23], which combines renders from four orthogonal views to process them using diffusion jointly, we create a 3D-aware image-grid context to enable fast re-training via a novel TriPlaneLite architecture while maintaining view-consistency. As shown in Table 1, our method can handle a variety of edits while only taking a fraction of the time compared to existing methods. Note that many of these are concurrent/unpublished works and we could not get code access for comparison.

3. Background

3.1. Neural Radiance Fields Revisited

Neural Radiance Fields (NeRFs) [MST*20] generate photo-realistic novel views of a 3D scene by representing it as a radiance field approximated by a neural network, usually an MLP. It takes as input a 3D spatial location ($\mathbf{p} := (x, y, z)$) and a view direction (θ, ϕ), and produces as output the corresponding color, $\mathbf{c}_{3D}$ (i.e., R, G, B) and volume density (σ). To render a pixel in an image, NeRF projects rays from the camera, and then samples points along the rays in 3D space. Using classical volumetric rendering [PD84], the colors and densities are converted into pixel colors by integrating the sampled color along the ray as,

$$c_{2D} := \sum_i T_i \alpha_i \mathbf{c}^i_{3D}, \tag{1}$$

where $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$ and $T_i = \prod_j^{i-1}(1 - \alpha_j)$; $\mathbf{c}^i_{3D}$ denotes the color on the i-th point on the ray and δ_i is the distance between samples along the ray.

3.2. Radiance Feature Fields

Radiance Feature Fields [KMS22, TLLV22b] extends NeRFs to capture semantic features to enable the selection of specific regions in a scene by distilling features from large scale self-supervised image models such as DINO [CTM*21] into 3D feature fields (f_{sem}). To render the color (c_{2D}) and semantic features (f_{2D}) of a pixel, it follows volume rendering as,

$$\begin{aligned} c_{2D} &:= \sum_i T_i \alpha_i \mathbf{c}^i_{3D} \\ f_{2D} &:= \sum_i T_i \alpha_i \mathbf{f}^i_{sem}, \end{aligned} \tag{2}$$

where we additionally optimize/distill volumetric features $\mathbf{f}_{3D}$ at each location in space.

3.3. Representing NeRFs as Tri-planes

Storing a distributed NeRF in a volume runs into cubic storage space in the resolution of the underlying grid. We can significantly reduce the training time and memory footprint for NeRFs by representing it as a tri-plane [CLC*22]. This approach maps a 3D spatial location ($\mathbf{p} := (x, y, z)$) to 3 hidden feature vectors as inputs to the NeRF by interpolating among three distinct feature planes, P_{xy}, P_{xz}, and P_{yz}, via orthographically projecting $\mathbf{p}$ to the three planes, and adding (or concatenating) the feature vectors. K-planes [FKMW*23] alternately restructures the information by multiplying the feature vectors instead of adding them.

4. ProteusNeRF

Overview. Starting from a set of posed images (i.e., images with associated cameras $\{I_i, C_i\}_{i=1:n}$), we first optimize a NeRF represented using our TriPlaneLite representation (Section 4.1). The user then chooses a camera view and indicates an image patch from the rendered frame to select any object she wishes to modify. Based on the image selection, we use the distilled semantic features to automatically select an object region, denoted by a 3D mask $\mathcal{M}_{\text{sel}}$, relying on similarity in the tri-plane feature field. We then render images from orthogonal views to form a 2×2 *3D-aware image context* to provide guidance (Section 4.2). The user then uses image editing tools, either a traditional image editor (e.g., Gimp, Photoshop) or a text-guided generative editor (e.g., InstructPix2Pix [BHE23]), to edit the guidance image. For edits involving only appearance changes, thanks to our TriPlaneLite architecture, we train a residual MLP with a minimal memory footprint (36KB) to facilitate rapid changes to the selected object (~10 seconds per edit). This approach also facilitates progressive edits. For edits involving geometric modifications, we fine-tune the whole NeRF (i.e.,

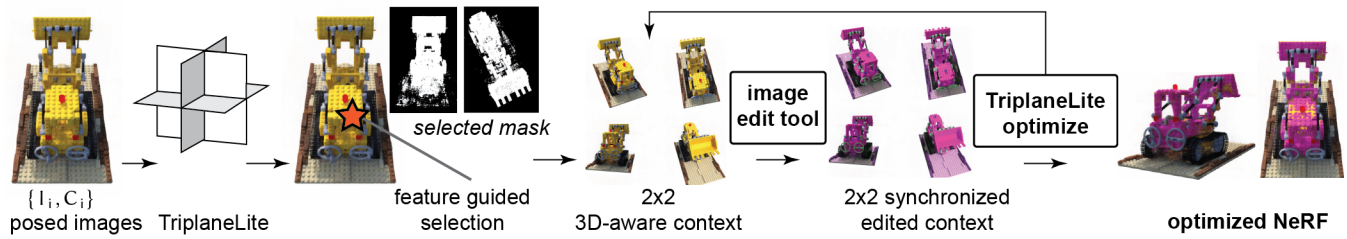

Figure 3: We present ProteusNeRF that takes in a set of posed images and encodes it as feature-distilled NeRF in a TriplaneLite representation. The user can easily select a part (yellow legos) that gets converted to a 3D mask $\mathcal{M}_{\text{sel}}$. We generate a novel 3D-aware image context that allows editing via imaging tools while still producing view-coherent edits. This edited context is then converted back to view-consistent NeRFs by fine-tuning the TriplaneLite. The context image is updated and the process is iterated (2-3 times in our examples). Editing, primarily appearance editing, runs at interactive framerates.

both geometry and appearance branches) on the 3D-aware image context, which is iteratively updated to capture the updated geometry (i.e., NeRF density). In the generative editing setup, we modify the image context using Stable Diffusion [RBL*22] with conditional controls produced using rendered depth and Canny edge maps via a pre-trained ControlNet [ZA23]. This allows us to fine-tune on a small set of images (4 images for a 2×2 image-grid context) for each iteration, resulting in accelerated re-training of the NeRF (~70 seconds for each edit). Figure 3 shows an overview of our algorithm.

4.1. TriPlaneLite: Residual Tri-plane Feature Field

We use a tri-plane [CLC*22] to represent a NeRF owing to its efficiency and compactness. We first distill the captured geometry from the tri-plane using an MLP ϕ_{geom} to map to density σ and geometric features f_{geom}. We then map f_{geom} to volumetric semantic features, f_{sem}, using an MLP ϕ_{sem} with supervision from a large-scale pre-trained model (we have used DINO [CTM*21] features in our experiments). Based on the intuition that it is possible to learn the color of a point directly from its semantic features, we distill the semantic features using another MLP ϕ_{color} to learn color (r, g, b). Given a camera, we 'render' final pixel colors and semantic features are obtained using volumetric rendering. See Figure 6. As a simplification, we ignore view dependent effects by eliminating the viewing direction as an input to the NeRF, similar to EG3D [CLC*22].

Once the NeRF is pre-trainined on a scene, the user can select an image patch of the object to indicate her region of interest. Based on the 2D query patch mask ($\mathcal{M}^q_{2D}$), we compute the mean of the 2D feature vectors of each pixel of the patch, $\bar{f}^q_{2D}$. The volumetric semantic features (f_{sem}) are thresholded against $\bar{f}^q_{2D}$ to obtain a final 3D mask of the object, $\mathcal{M}_{\text{sel}}$. See Figure 4.

4.2. 3D-Aware Image Context

To edit the NeRF, we need to optimize it on a new set of edited images. While it is possible to make simple color changes while maintaining view consistency, the problem is non-trivial for making large appearance or geometric modifications. This problem is exacerbated for generative models such as Stable Diffusion [RBL*22], which can produce markedly different images resulting in geometric inconsistencies in the edited NeRF. To address this challenge, we introduce a simple yet highly effective solution that we term as 3D-aware image context: merging multiple images into a single image grid when editing. Given a selected camera pose C, we generate a 2×2 grid of images around C using renderings from using rotated cameras by $(\pm\Delta, \pm\Delta)$ along latitude/longitude on the view sphere. We used $\Delta = 5°$ in our tests.

This allows the generative model to share the same latent code when modifying the image, resulting in the edited object's coherent appearance and geometry. The solution can be seen as a crude approximation to learning an attention map by giving local image context. This context also applies to traditional image editing software as various operations such as contrast adjustment, can be uniformly applied across multiple images. Furthermore, it simplifies the editing process as the user only needs to make a single modification.

4.3. NeRF Editing

We divide the editing process into two separate workflows depending on the task.

4.3.1. Smaller appearance-only edits

For appearance changes using traditional image processing such as recoloring, contrast changes, altering the white balance, etc., there are many mature image editing tools such

Figure 4: Once the input posed image $\{I_i, C_i\}_{i=1:n}$ are feature-distilled into TriplaneLite, the user can select a region in any of the images (shown in orange here), which is then used to extract a 3D mask $\mathcal{M}_{\text{sel}}$. Suppressing the corresponding signal in the mask, reveals the background across views.

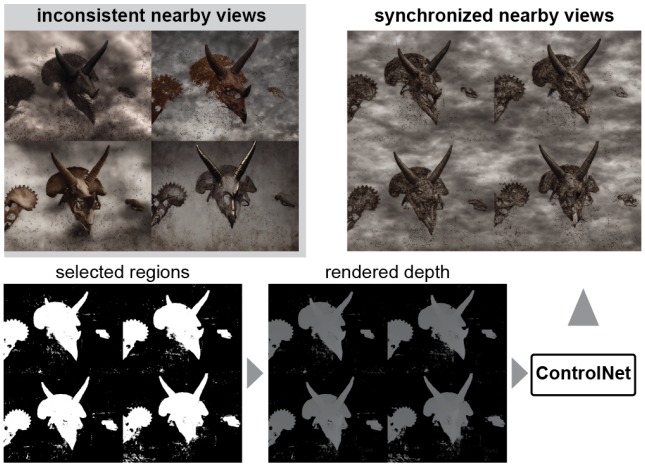

Figure 5: (Top-left) Without any 3D guidance, edits from nearby (camera) views can be inconsistent, and hence, any intended edits can get lost during subsequent NeRF refinement. Instead, we advocate using 3D-aware image context, a simple change that encourages edit consistency among nearby views. Given the selected object mask, we extract depth-only rendering on a 2×2 grid of nearby cameras to give guidance to a pre-trained ControlNet [ZA23], optionally augmented with feature edges and/or text prompts. (Top-right) These edits generated using 3D-aware image context are more consistent and preserved during subsequent NeRF fine-tuning.

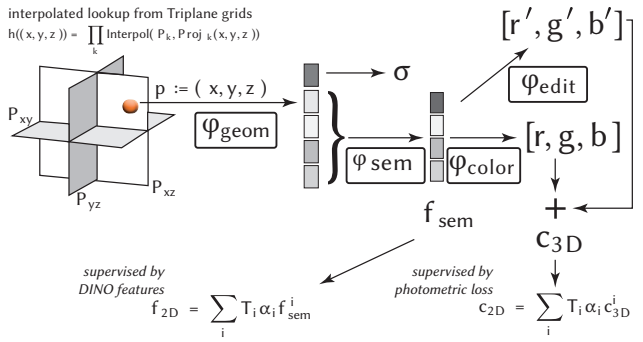

Figure 6: We encode an object as a NeRF using our TriplaneLite structure that takes as input a point $\mathbf{p} := (x, y, z)$ and encodes it as features $h(\mathbf{p})$ by projection, lookup, and interpolation of features from three planar grids (P_{xy}, P_{yz}, P_{xz}). We then enable learning via four different MLPs $\phi_{geom}, \phi_{sem}, \phi_{color}$, and ϕ_{edit}, to factorize density, semantic features, color, and residual appearance respectively. Training is supervised via photometric loss and distillation of image space semantic features (DINO features). This enables semantic selection (see Figure 4). Furthermore, the structuring lets us interactively receive appearance updates while requiring a low memory overhead (36KB/edit).

as Gimp and Adobe Photoshop. Generative methods such as InstructPix2Pix [BHE23] enable more creative appearance changes using natural language as instruction. To edit the NeRF on the updated images, we propose adding a simple 3-layer residual MLP ϕ_{edit} with a minimal footprint of 36KB. See Figure 6. During re-training on the 2×2 edited 3D-aware image context, we freeze all layers of the NeRF except ϕ_{edit} to learn the new color. Essentially, this operation learns a new mapping of the distilled features to the target color values provided by the guidance context images. Training such a small network, while keeping geometric encoding fixed, converges on merely four images after two epochs, resulting in rapid optimization as,

$$c_{2D} := \sum_i T_i \alpha_i c_{3D\,i} \odot (\sim \mathcal{M}_{\text{sel}}) + \sum_i T_i \alpha_i (c_{3D\,i} + \phi_{edit}(f_{sem\,i})) \odot \mathcal{M}_{\text{sel}}. \tag{3}$$

4.3.2. Larger edits

We fine-tune the entire NeRF for larger appearance or geometry changes using an iteratively updated 3D-aware image context. Specifically, we edit the image context using Stable Diffusion and add conditional control to preserve the local properties of the object. We utilize image inpainting to modify the object selectively, i.e., the masked region, along with ControlNet [ZA23] with depth maps and Canny edge maps for guidance. The depth map is trivial to obtain as, after pretraining, the NeRF can accurately represent the scene, and can obtained by simply volume rendering the sampled density values. See Figure 5.

Let $I_i^j \in I$ be the original 3D-aware image context and $E_i^j \in E$ be the edited image context, where i is a camera position chosen at random and j is the epoch. For an edited 2×2 image context, $E^0 := \{E_a^0, E_b^0, E_c^0, E_d^0\}$, after the first epoch, we fix two cells in the grid with E_a^0 and E_b^0 without masking to serve as guidance for the next iterations of the edit. This simple adaptation allows the generative method to fill the unmasked region using the previously edited images as references, thus ensuring coherence in the edited images across epochs.

4.4. Optimization

We re-train the NeRF on the edited 3D-aware image context (E_i^j) using MSE with *L1* regularization to encourage sparsity in the tri-plane parameters. *L1* regularization helps to reduce the problem of floaters and artifacts during rendering in the edited NeRF, particularly for geometric edits. The total loss, L, is the sum of the reconstruction MSE and the *L1* penalty,

$$L := \frac{1}{mn} \sum_{q=1}^{n} \sum_{p=1}^{m} (E_p^q - c_{2Dp}^q)^2 + \lambda \sum |W_P|, \tag{4}$$

where m and n are the height and width of the rendered image, respectively; $p \in 1 : m$ and $q \in 1 : n$ are the pixel locations, λ is a constant to control the regularization, and W_P are the weights of the tri-plane parameters (P_{xy}, P_{xz}, and P_{yz}).

5. Results

Dataset. To demonstrate the effectiveness of our method, we show experiments on scenes from the LLFF [MSOC*19] and Blender Synthetic [BMV*22] datasets. We select three scenes from the LLFF dataset – flower, horns, and fern – and choose the Lego sequence from the Blender Synthetic dataset. To show the efficacy of our method on casually captured videos, we utilize the face sequence from InstructNeRF2NeRF

Table 2: Our method is significantly faster than InstructNeRF2NeRF (IN2N) [HTE*23] while occupying a fraction of its memory footprint, particularly when utilizing the residual MLP ϕ_{edit} for appearance changes. We measure the time required of ours for image size (504 × 378) and IN2N for image size (497 × 369). All experiments are run on a single Nvidia A100 GPU.

Method	Settings	Time ↓ (minutes)	Storage↓ (KB)
IN2N	3000 iterations (small edits)	∼ 10	∼171844
	30000 iterations (large edits)	∼ 100	∼171844
IN2N w/ fixed context	3000 iterations (small edits)	∼ 13	∼171844
IN2N w/ fixed context	15000 iterations (large edits)	∼ 65	∼171844
Ours using MLP ϕ_{edit}	788 iterations (small edits)	∼ 0.25	36
Ours re-training w/ fixed context	4467 iterations (large edits)	∼ 1.08	32917
Ours re-training w/ iterative context	4467 iterations (large edits)	∼ 1.42	32917

Figure 7: Although our 3D-aware image context helps to synchronize edits across nearby views, inconsistencies can still occur. We iterate between context-guided image edits, distillation into a refined NeRF, and regenerating new guidance images. Typically, we found that 2-3 iterations was enough to strike a balance between expressive edits and interactive performance.

[HTE*23], and capture a video of a bear doll on iPhone12 Pro using Polycam [pol23] processed with Nerfstudio [TWN*23b].

Implementation details. During NeRF pretraining, we uniformly sample 96 and 192 rays for the LLFF dataset and Blender Synthetic datasets, respectively. The architecture is optimized using Adam [KB14] for 30,000 iterations on all views. During editing, for small appearance changes, we train our MLP ϕ_{edit} in about 800 iterations for each edited image. For larger edits involving geometric changes, we re-train the NeRF for 4467 iterations with a batch size of 512, a learning rate of 2×10^{-4}, and L1 regularization ($\lambda = 2 \times 10^{-1}$).

We test the official implementation (v0.1.0) of InstructNeRF2NeRF using default settings in Nerfstudio [TWN*23a]. For the bear and face data sets, we used the default settings of Nerfstudio v0.3.2 for pre-training for 30,000 iterations with a batch size of 4096. In addition, we extended the 3D-aware image context to the iterative dataset update method of InstructNeRF2NeRF to update the image every 40 iterations using a 2x2 context.

5.1. Qualitative Evaluation

Our method can handle a wide range of edits, including appearance changes, small geometry changes, and addition and deletion of objects, while maintaining a fast and lightweight framework. We compare our method against related methods to recolor a flower scene as illustrated in Figure 2. We encourage the readers to explore the additional supplementary material, which includes a web page for video comparison. CLIP-NeRF [WCH*22] bleeds the color changes into the global scene causing unwanted recoloring. While the editing is more localized for DFF [KMS22], we see unnatural color gradients. In addition, the pistil is recolored, which is undesirable. We utilize traditional image editors in our workflow to recolor the 3D-aware image context. Although RecolorNeRF too shows desirable results, its editing framework is less intuitive and limited to simple color changes. Our approach demonstrates the efficacy of image processing tools for making precise color changes in NeRFs. This is further highlighted in the accurate contrast tone mapping edits in Figure 8.

To show larger appearance changes, several examples in Figure 8 show changes in texture, color, and geometry. The creative edits utilizing our generative workflow show substantial appearance changes while being local such as editing the appearance of horns, fern and flower. Our method demonstrates adaptability by accommodating small geometric changes, such as adding a vase in the fern sequence or a cowboy hat in the face scene, or removing the flower (Figure 4).

5.2. Quantitative Evaluation

Our method is significantly faster than existing works, which can take thirty minutes to two hours for larger appearance changes or geometric modifications. We compare the time and storage required by our method to edit a scene against InstructNeRF2NeRF in Table 2. For smaller appearance only edits, our method shows ∼ 40x speedup over InstructNeRF2NeRF while occupying a fraction of the memory space when training is done on MLP ϕ_{edit}. The minimal memory footprint of MLP ϕ_{edit} (36KB) paves the way for layered editing of NeRFs, which would not only help to make fine-grained changes in the scene but also allow the user to preserve the editing history. For larger edits, our method is ∼ 65x faster

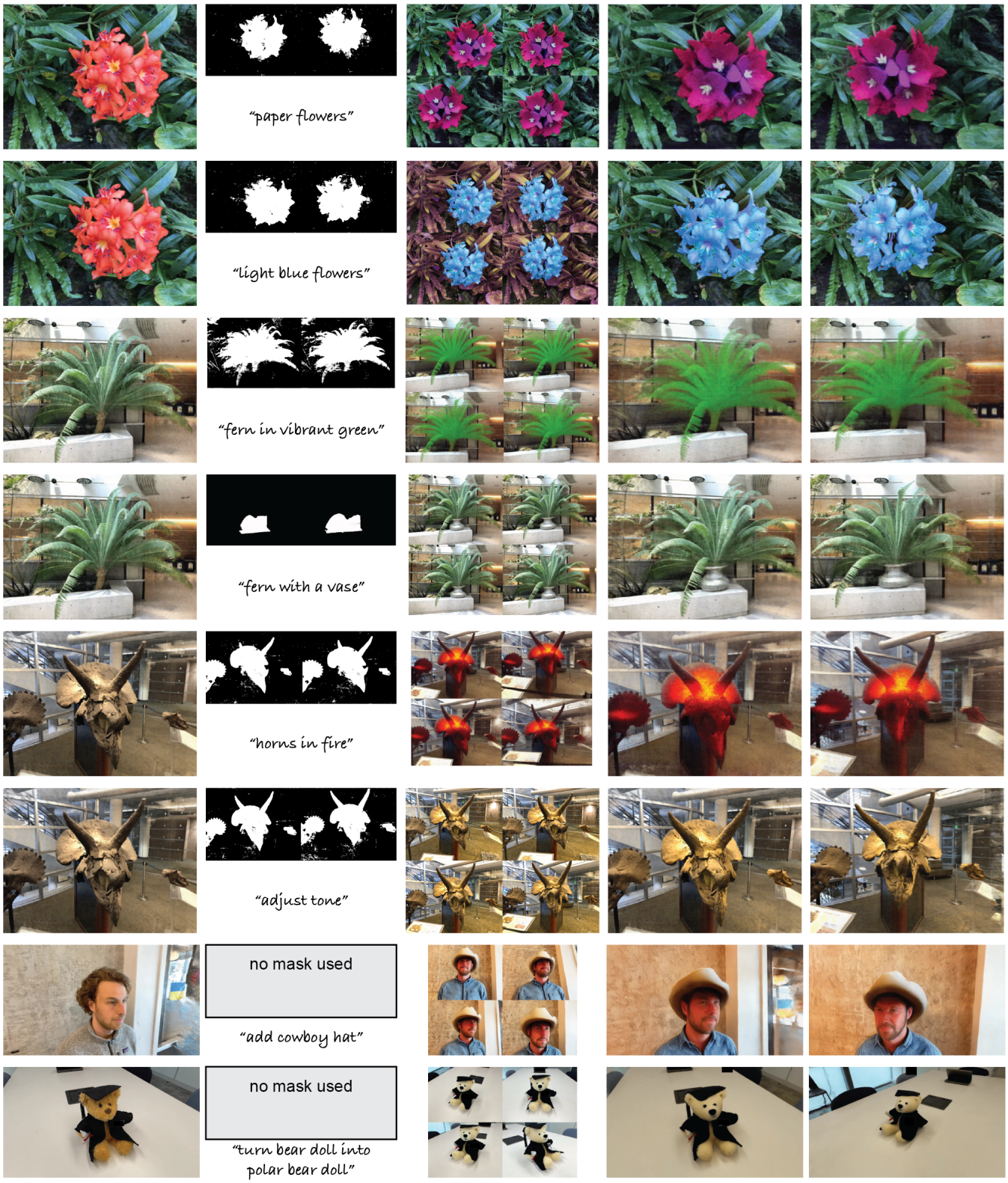

Figure 8: Results gallery. (Left-to-right) Input scene; selection masks with target edit description; 3D-aware image context (one iteration shown); final NeRF in source view and a novel view. Please see the supplementary for video results.

when using iterative context and $\sim$ 130x faster for a fixed context.

5.3. Ablation

We probe the effectiveness of the iterative 3D-aware image context in this section. We compare quantitative and qualitative results against three baselines: (i) editing using a single view (without using any contextual information); (ii) editing using a iterative single view update; (iii) editing using a fixed 3D-aware image context. Figure 9 shows the potency of our 3D-aware image context as it achieves the best visual results. Without contextual information, there is no coherence among the edited images, resulting in inconsistent views. When training is done on a single view, performance on novel views is significantly reduced as there is no geometric prior to appropriately map the edited appearance and geometry from a single edited 2D image to 3D. While a fixed 3D-aware image context can ease the problem, iteratively updating the context allows us to provide additional views, enhancing the convergence of the network.

This hypothesis is further supported by the results of our user study presented in Figure 10. The study involved 6 edited scenes and 20 participants, each tasked with choosing among 4 options the video that most accurately matched a provided text prompt and selecting the video that was the most view-consistent. An overwhelming 66% of the participants found that edits using 3D-aware image context were more similar to the provided text prompt and 70.8% of the participants found that using 3D-aware image context led to more view-consistent edits. Among the participants who preferred edits utilizing image context, 57.7% of them found that results with iterative context led to higher similarity, and 61.1% of them found that utilizing iterative context led to more view-consistent results.

Figure 9: Ablation. We ablate our design choices. (Top-to-bottom) Results with edits coming from a single view result in floaters; with edits from 4 views but not synchronized context leads to marginal edits; with a fixed 3D-aware image context fails to take into account updated geometry; and finally, with 3 iterations of 3D-aware image context updates, we get geometric and appearance edits. See also corresponding user preferences in Table 3. Target edit, *'turn fern into ice'*. Please see the supplementary webpage for ablation videos.

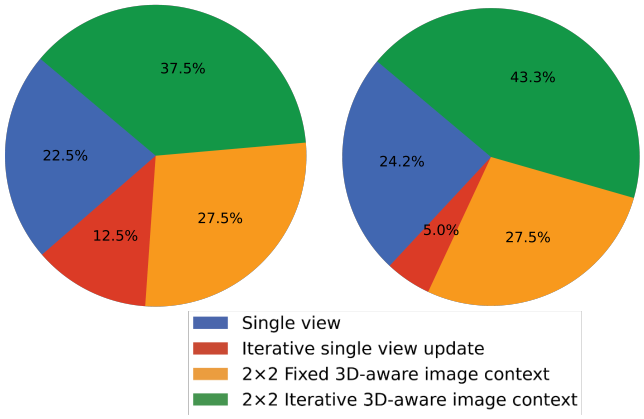

Figure 10: User study on ablation results. Left: edited result is the most similar to the text prompt. Right: edited result is the most view-consistent. For a vast majority of the edited scenes, participants prefer results using 3D-aware image context, particularly when using an iterative approach.

We cannot test the baselines on novel views of edited scenes due to the lack of ground truth data. Instead, we show that the iterative 3D-aware image context is consistent in geometry and appearance by using LPIPS [ZIE*18] to compute the perceptual similarity between the edited context (E_i) and the corresponding rendered image (R_i). Intuitively, scene editing optimized on a single fixed view should yield the lowest LPIPS as it has completely overfitted to that view. We show in Table 3 that even when using an iteratively updated 2×2 3D-aware image context consisting of 4 views per epoch, we achieve comparable results to a single overfitted view. This subsequently implies that our 2x2 context image is view-consistent. When the network is trained on an iterative single-view update, although the scene should be ideally overfitted to the single view, it is outperformed by the 3D-aware image context as the training is continuously destabilized by inconsistent edits across iterations. See our supplementary video results.

Table 3: LPIPS scores on ablated design choices, tested on the final edited view(s). Our iterative 2×2 3D-aware image context consisting of 4 views per epoch, attains comparable LPIPS to a single overfitted view and outperforms the iterative single view update which is inconsistent across iterations.

Input	LPIPS ↓
Single view	0.073
Iterative single-view update	0.115
2×2 Fixed 3D-aware image context	0.086
2×2 Iterative 3D-aware image context	0.089

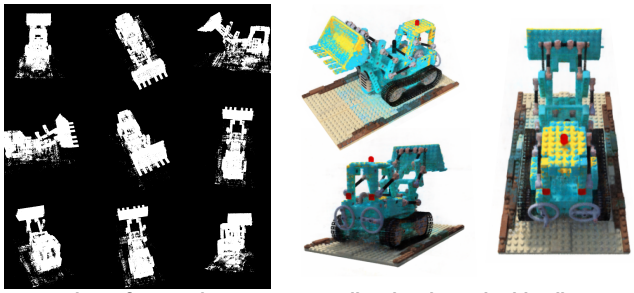

Figure 11: Our method can produce imperfect mask $\mathcal{M}_{\text{sel}}$ when DINO based semantic feature distillation fails to capture in-class color variations fully. In addition, specularity contradicts our assumption of a uniform diffuse appearance. Here, the inferior mask results in changes in the Lego scene to produce color bleeding (blue into yellow). One possible solution to address this issue would be to combine feature and color-based selection.

6. Conclusions

We have presented ProteusNeRF as a fast and lightweight framework that supports object-centric NeRF edits at interactive rates. We observe that existing NeRFs can be restructured to facilitate semantic selection via a feature distillation at the training stage, and any subsequent appearance change can then simply be interpreted as a remapping operation from the selected features to the target colors. This greatly simplifies the selection as well as lightweight storage of appearance edits via residual MLPs, enabled by our TriPlaneLite architecture. Further, we introduce editing via a novel 3D-aware image context that allows users to leverage existing image editing setups for NeRF editing.

Our approach has several limitations that we plan to address in future explorations.

(i) *Handle larger geometric changes:* While we support only finer geometric changes, we would like to explore how to enable larger geometric changes at interactive rates, without having to re-optimize the NeRF from scratch or distill the NeRF asset into textured meshes.

(ii) *Reduce floaters:* In our current setup, geometric changes can introduce undesirable floaters as we do not have any priors to regularize density during the ambiguous 2D-to-3D uplifting stage. In the future, we would like to utilize shape priors to regularize against these artifacts, possibly using a diffusion-based solution [WWT*23].

(iii) *Support specular effects:* In the future we would like to extend appearance edits to handle view-dependent specular effects. One possibility is to create NeRFs with a factorized diffuse and specular representation of color, and then apply appearance edits to only diffuse channels. However, the challenge is to make such a factorization without sacrificing the simplicity of the NeRF capture process. Moreover, it is unclear how to create view-dependent edits using the 3D-aware image context.

(iv) *Unifying NeRF editing and generation:* Finally, we would like to link generative image editing and 3D-aware NeRF guidance more closely to explore direct 3D-aware latent code blending (i.e., for latent diffusion), across views, towards a synchronized multi-view/image diffusion model [RBL*22, BTYLD23] for text-guided editing, while maintaining interactive framerates.

References

[BHE23] 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* (2023), pp. 18392–18402. 2, 3, 5

[BMV*22] 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* (2022), pp. 5470–5479. 5

[BTYLD23] Bar-Tal O., Yariv L., Lipman Y., Dekel T.: Multidiffusion: Fusing diffusion paths for controlled image generation. *arXiv preprint arXiv:2302.08113* (2023). 9

[CHIS23] Croitoru F.-A., Hondru V., Ionescu R. T., Shah M.: Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023). 2

[CLC*22] Chan E. R., Lin C. Z., Chan M. A., Nagano K., Pan B., De Mello S., Gallo O., Guibas L. J., Tremblay J., Khamis S., et al.: Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 16123–16133. 2, 3, 4

[CTM*21] Caron M., Touvron H., Misra I., Jégou H., Mairal J., Bojanowski P., Joulin A.: Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 9650–9660. 3, 4

[CTT*22] 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* (2022), pp. 1475–1484. 2

[CXG*22] Chen A., Xu Z., Geiger A., Yu J., Su H.: Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)* (2022). 2

[FKMW*23] Fridovich-Keil S., Meanti G., Warburg F. R., Recht B., Kanazawa A.: K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR* (2023). 2, 3

[FTC*22] Fridovich-Keil and Yu, Tancik M., Chen Q., Recht B., Kanazawa A.: Plenoxels: Radiance fields without neural networks. In *CVPR* (2022). 2

[GAL23] Gordon O., Avrahami O., Lischinski D.: Blended-nerf: Zero-shot object generation and blending in existing neural radiance fields. *arXiv preprint arXiv:2306.12760* (2023). 3

[GKJ*21] Garbin S. J., Kowalski M., Johnson M., Shotton J., Valentin J.: Fastnerf: High-fidelity neural rendering at 200fps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 14346–14355. 2

[GPAM*20] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y.: Generative adversarial networks. *Communications of the ACM 63*, 11 (2020), 139–144. 2

[GWHD23] Gong B., Wang Y., Han X., Dou Q.: Recolornerf: Layer decomposed radiance field for efficient color editing of 3d scenes. *arXiv preprint arXiv:2301.07958* (2023). 2, 3

[HGPR20] Harshvardhan G., Gourisaria M. K., Pandey M., Rautaray S. S.: A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review 38* (2020), 100285. 2

[HHY*22] 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* (2022), pp. 18342–18352. 2

[HJA20] Ho J., Jain A., Abbeel P.: Denoising diffusion probabilistic models. *Advances in neural information processing systems 33* (2020), 6840–6851. 2

[HSM*21] Hedman P., Srinivasan P. P., Mildenhall B., Barron J. T., Debevec P.: Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 5875–5884. 2

[HTE*23] Haque A., Tancik M., Efros A. A., Holynski A., Kanazawa A.: Instruct-nerf2nerf: Editing 3d scenes with instructions. *arXiv preprint arXiv:2303.12789* (2023). 2, 3, 6

[HTS*21] Huang H.-P., Tseng H.-Y., Saini S., Singh M., Yang M.-H.: Learning to stylize novel views. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 13869–13878. 2

[JKK*23] Jambon C., Kerbl B., Kopanas G., Diolatzis S., Drettakis G., Leimkühler T.: Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 6*, 1 (2023). 2, 3

[JMB*22] Jain A., Mildenhall B., Barron J. T., Abbeel P., Poole B.: Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 867–876. 2

[KB14] Kingma D. P., Ba J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 6

[KLB*23] Kuang Z., Luan F., Bi S., Shu Z., Wetzstein G., Sunkavalli K.: Palettenerf: Palette-based appearance editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 20691–20700. 2, 3

[KMS22] Kobayashi S., Matsumoto E., Sitzmann V.: Decomposing nerf for editing via feature field distillation. *Advances in Neural Information Processing Systems 35* (2022), 23311–23330. 2, 3, 6

[KRWM22] Karnewar A., Ritschel T., Wang O., Mitra N.: Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings* (New York, NY, USA, 2022), SIGGRAPH '22, Association for Computing Machinery. URL: https://doi.org/10.1145/3528233.3530707, doi:10.1145/3528233.3530707. 2

[KW13] Kingma D. P., Welling M.: Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013). 2

[LK23] Lee J.-H., Kim D.-S.: Ice-nerf: Interactive color editing of nerfs via decomposition-aware weight optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 3491–3501. 2, 3

[MESK22] Müller T., Evans A., Schied C., Keller A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG) 41*, 4 (2022), 1–15. 2

[MRP*23] Metzer G., Richardson E., Patashnik O., Giryes R., Cohen-Or D.: Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 12663–12673. 2

[MSOC*19] Mildenhall B., Srinivasan P. P., Ortiz-Cayon R., Kalantari N. K., Ramamoorthi R., Ng R., Kar A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)* (2019). 5

[MST*20] Mildenhall B., Srinivasan P. P., Tancik M., Barron J. T., Ramamoorthi R., Ng R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). 1, 2, 3

[NPLX22] Nguyen-Phuoc T., Liu F., Xiao L.: Snerf: stylized neural implicit representations for 3d scenes. *arXiv preprint arXiv:2207.02363* (2022). 2

[PD84] Porter T., Duff T.: Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (1984), pp. 253–259. 3

[PJBM22] Poole B., Jain A., Barron J. T., Mildenhall B.: Dreamfusion: Text-to-3d using 2d diffusion. *arXiv* (2022). 2, 3

[pol23] polycam: Polycam- lidar and 3d scanner for iphone and android., 2023. 6

[PTL*23] Pan X., Tewari A., Leimkühler T., Liu L., Meka A., Theobalt C.: Drag your gan: Interactive point-based manipulation on the generative image manifold. In *ACM SIGGRAPH 2023 Conference Proceedings* (2023), pp. 1–11. 2

[RBL*22] Rombach R., Blattmann A., Lorenz D., Esser P., Ommer B.: High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 10684–10695. 2, 4, 9

[RDN*22] Ramesh A., Dhariwal P., Nichol A., Chu C., Chen M.: Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125 1*, 2 (2022), 3. 2

[RKH*21] Radford A., Kim J. W., Hallacy C., Ramesh A., Goh G., Agarwal S., Sastry G., Askell A., Mishkin P., Clark J., et al.: Learning transferable visual models from natural language supervision. In *International conference on machine learning* (2021), PMLR, pp. 8748–8763. 3

[SSC22] Sun C., Sun M., Chen H.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR* (2022). 2

[TLLV22a] 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)* (2022), IEEE, pp. 443–453. 3

[TLLV22b] Tschernezki V., Laina I., Larlus D., Vedaldi A.: Neural Feature Fusion Fields: 3D distillation of self-supervised 2D image representations. In *Proceedings of the International Conference on 3D Vision (3DV)* (2022). 3

[TMT*23] Tsalicoglou C., Manhardt F., Tonioni A., Niemeyer M., Tombari F.: Textmesh: Generation of realistic 3d meshes from text prompts. *arXiv preprint arXiv:2304.12439* (2023). 3

[TWN*23a] Tancik M., Weber E., Ng E., Li R., Yi B., Kerr J., Wang T., Kristoffersen A., Austin J., Salahi K., Ahuja A., McAllister D., Kanazawa A.: Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings* (2023), SIGGRAPH '23. 6

[TWN*23b] 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* (2023), pp. 1–12. 6

[WCH*22] Wang C., Chai M., He M., Chen D., Liao J.: Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 3835–3844. 2, 3, 6

[WJC*22] Wang C., Jiang R., Chai M., He M., Chen D., Liao J.: Nerf-art: Text-driven neural radiance fields stylization. *arXiv preprint arXiv:2212.08070* (2022). 3

[WSW21] Wang Z., She Q., Ward T. E.: Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Computing Surveys (CSUR) 54*, 2 (2021), 1–38. 2

[WWT*23] Warburg F., Weber E., Tancik M., Holynski A., Kanazawa A.: Nerfbusters: Removing ghostly artifacts from casually captured nerfs, 2023. arXiv:2304.10532. 9

[WZAS23] Wang D., Zhang T., Abboud A., Süsstrunk S.: Inpaintnerf360: Text-guided 3d inpainting on unbounded neural radiance fields. *arXiv preprint arXiv:2305.15094* (2023). 2, 3

[XH22] Xu T., Harada T.: Deforming radiance fields with cages. In *European Conference on Computer Vision* (2022), Springer, pp. 159–175. 2, 3

[YLT*21] Yu A., Li R., Tancik M., Li H., Ng R., Kanazawa A.: Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 5752–5761. 2

[ZA23] Zhang L., Agrawala M.: Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543* (2023). 2, 4, 5

[ZIE*18] Zhang R., Isola P., Efros A. A., Shechtman E., Wang O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 586–595. 8

[ZWL*23] Zhuang J., Wang C., Liu L., Lin L., Li G.: Dreameditor: Text-driven 3d scene editing with neural fields. *arXiv preprint arXiv:2306.13455* (2023). 2, 3