Learning Naturally Aggregated Appearance for Efficient 3D Editing

Ka Leong Cheng[1,2], Qiuyu Wang[2], Zifan Shi[1,2], Kecheng Zheng[2,3], Yinghao Xu[2,4], Hao Ouyang[1,2], Qifeng Chen[1†], Yujun Shen[2†]

[1]HKUST [2]Ant Group [3]CAD&CG, ZJU [4]Stanford

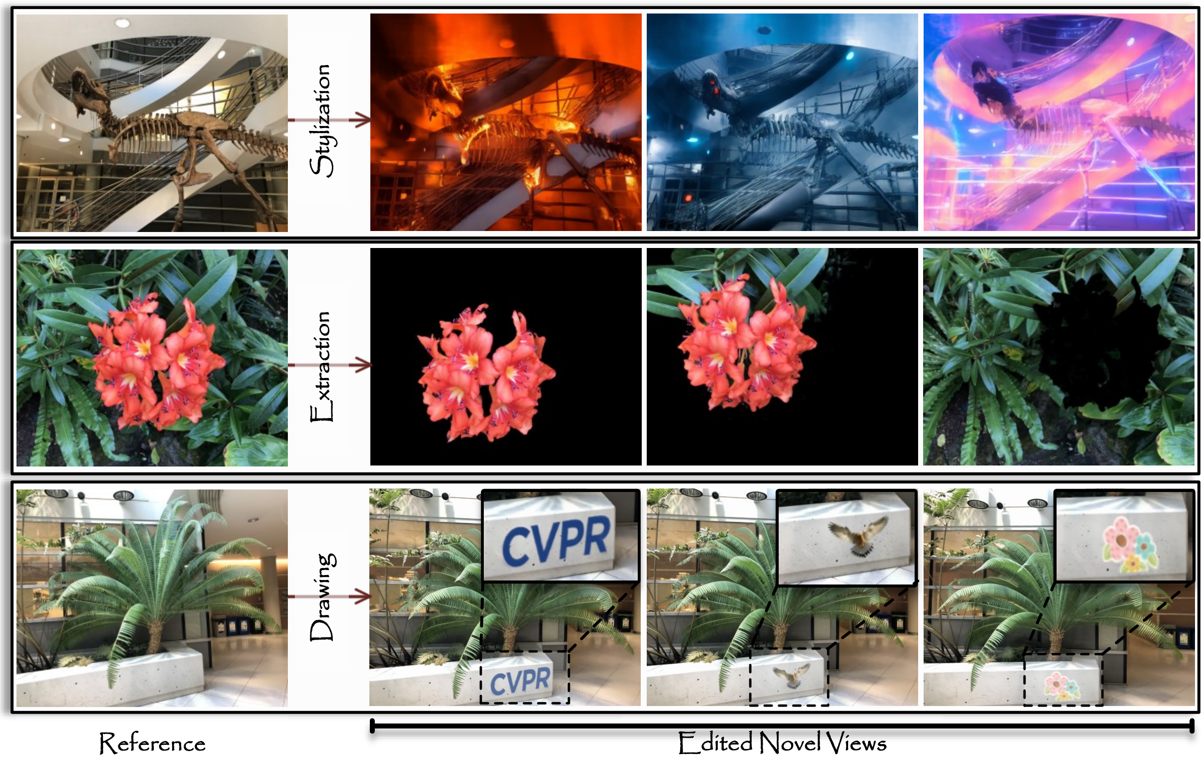


Figure 1. **AGAP** aggregates 3D appearance as natural 2D canonical images. With 2D image processing tools, our method enables various ways of 3D editing, including (a) scene stylization, (b) content extraction, and (c) interactive drawing, without further re-optimization.

Abstract

Neural radiance fields, which represent a 3D scene as a color field and a density field, have demonstrated great progress in novel view synthesis yet are unfavorable for editing due to the implicitness. In view of such a deficiency, we propose to replace the color field with an explicit 2D appearance aggregation, also called canonical image, with which users can easily customize their 3D editing via 2D image processing. To avoid the distortion effect and facilitate convenient editing, we complement the canonical image with a projection field that maps 3D points onto 2D pixels for texture lookup. This field is carefully initialized with a pseudo canonical camera model and optimized with offset regularity to ensure ***naturalness*** *of the aggregated appearance. Extensive experimental results on three datasets suggest that our representation, dubbed AGAP, well supports various ways of 3D editing (e.g., stylization, interactive drawing, and content extraction) with no need of re-optimization for each case, demonstrating its generalizability and efficiency. Project page is available at https://felixcheng97.github.io/AGAP/.*

†Corresponding author

1. Introduction

While recent advancements in 3D representations like neural radiance fields (NeRF) [31] have shown impressive reconstruction capabilities for real-world scenes, the need for further progress in 3D editing arises as the desire to recreate and manipulate these scenes. The field of 3D editing has witnessed significant development in recent years. Traditional 3D modeling approaches [21, 42, 43, 55] typically rely on reconstructing scenes using meshes. By combining meshes with texture maps, we can enable appearance editing during the rendering process. However, these methods usually face difficulties in obtaining detailed and regular texture maps, typically in complex scenes, which significantly hinders effective editing and compromises user-friendliness.

Recent neural radiance fields offer high-quality scene reconstruction capabilities, but manipulating the implicit 3D representation embedded within neural networks is inherently complex and non-straightforward. Existing NeRF-based editing approaches can be mainly divided into two categories: some methods like [8, 9, 52, 56, 59, 63] target geometry editing, usually taking advantage of meshes, while the other [14, 24, 36, 60, 62] focuses on 3D stylization using images or texts as style guidance. However, re-optimizing the pre-trained NeRF models is necessary to incorporate the desired editing effects into the underlying 3D representation, resulting in time-consuming processes. Consequently, it is crucial to develop a user-friendly framework that can efficiently and effectively support various edits within a single model.

In this paper, we introduce a novel editing-friendly representation **`AGAP`** with naturally **Ag**gregated **Ap**pearance, consisting of a 3D density grid for geometry estimation and a canonical image plus a projection field for appearance or texture modeling. Our method attempts to link the 3D representation with a natural 2D aggregated canonical representation. Concretely, a learnable canonical image is designed as the interface for editing, which aggregates the appearance by projecting the 3D radiance to a natural-looking image by the associated projection field. To ensure the naturalness of the aggregated canonical image with strong representation capacity, the associated projection field is carefully initialized by a pseudo canonical camera model and complemented by a learned view-dependent offset. The underlying 3D structure is modeled by an explicit 3D density grid. After a scene is optimized through volume rendering, `AGAP` supports various ways of 3D editing in a *user-friendly* setting by applying different 2D image processing tools on the canonical images *without re-optimization* of the original model. We evaluate the effectiveness of our method on three datasets, including LLFF [30], Replica [13, 46], and Instruct-NeRF2NeRF [14] in various editing tasks, which are scene stylization, content extraction, and texture editing (*i.e.*, drawing). Experimental results further show that `AGAP` can achieve on-par performance with NeRF [31] in terms of PSNR with a hash grid-based projection field.

2. Related Work

Implicit 3D representation. 3D modeling [1, 15, 17, 19, 27, 45, 50, 51] is pivotal in computer graphics and computer vision. Traditionally, explicit representations such as voxels and meshes have been employed for 3D shape modeling, but they often face challenges related to detail preservation and limited flexibility in processing. In contrast, implicit 3D representation like NeRF [31], SDF [37, 52, 57], Occupancy Networks [29], describing 3D scenes through continuous implicit functions, excels in capturing detailed geometry with improved fidelity. Many further works aim at improving NeRF in terms of various aspects, such as modeling capacity [2, 3], generative modeling [6, 12, 44, 53], and camera pose estimation [25, 26, 54]. In particular, methods like DVGO [47], Plenoxels [11], InstantNGP [32], TensoRF [7] focus on improving the convergence speed of volume rendering for 3D scenes by modeling the geometry and appearance with explicit grid representations. Our method leverages this technique for our density grid and canonical image as well, enabling efficient and rapid convergence of 3D modeling.

Neural scene editing. Existing research on NeRF editing can be broadly categorized into two: one focuses on editing the geometry [8, 9, 52, 56, 59, 63]; the other, known as style-based editing [14, 24, 36, 60, 62], aims to achieve scene stylization. Our research aligns with the latter category. Many NeRF stylization methods [16, 33, 60] have adopted techniques from 2D image stylization with style loss and content loss on images for NeRF optimization. While these methods can deliver 3D-consistent editing, they are primarily limited to global texture modifications and lack flexibility. Later, CLIP-NeRF [49] incorporates text conditions by regularizing the CLIP embeddings of the global scene with input prompts. Subsequent studies [24] extract 2D features such as DINO [5, 34] for local editing. Most recently, Instruct-Nerf2Nerf [14] proposes an iterative approach to edit the input images using pre-trained diffusion models [4] for underlying NeRF optimization. Despite achieving high-fidelity editing results, these methods necessitate optimization for each text prompt or reference image, which is inefficient.

Neural atlases. Our work shares similar insights with the research area of neural atlases [20, 35, 58], which decompose videos into a canonical form with learned deformations, thereby enabling consistent video editing. Approaches such as neural layered atlases [20] employ an implicit network to distinguish foreground and background movement, dividing them into distinct layers. The CoDeF [35] methodology represents 2D videos using content deformation fields by integrating 3D hash tables [32]. However, these approaches lack 3D priors, limiting the effectiveness of 3D viewpoint changes in 3D scene editing.

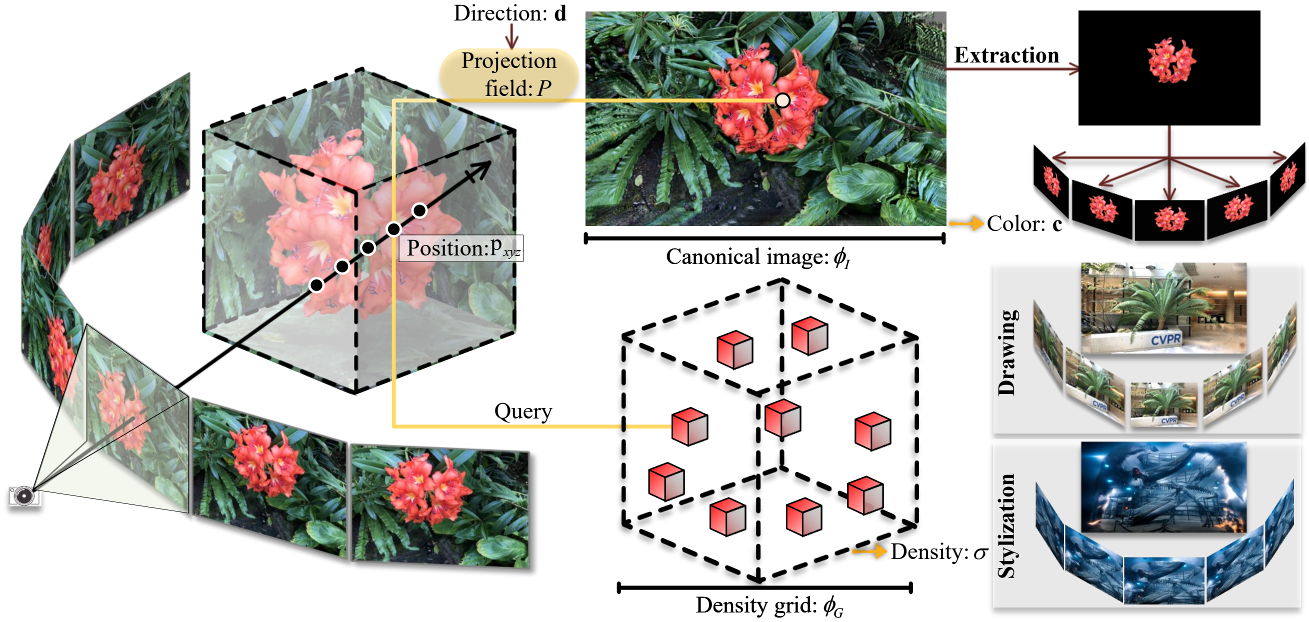


Figure 2. **The overall pipeline**. `AGAP` consists of two components: (1) an explicit 3D density grid ϕ_G to estimate geometry for density σ; (2) an explicit canonical image ϕ_I with an associated view-dependent projection field P to aggregate appearance for color $\mathbf{c}$. By performing 2D image processing on the canonical image, our method enables various editing (*e.g.*, content extraction, interactive drawing, and scene stylization) through volume rendering without the need for re-optimization.

3. Method

Formally, given a set of multi-view training images $\mathcal{I}$, our method models the scene appearance by an explicit canonical image ϕ_I plus a corresponding implicit projection field P inspired by [35, 38, 39]; the scene geometry is estimated by an explicit 3D density grid ϕ_G. With such a representation, one can render different views of the scene through volume rendering. Our key property is that by explicitly editing the canonical image ϕ_I, it can propagate the edited appearance to the whole scene through the projection field P without any further re-optimization. An overview of our framework is shown in Fig. 2.

3.1. Preliminary

Volume rendering [18, 31] accumulates colors and densities of the 3D points sampled along the camera rays to render images. For a given camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ denoted by its origin $\mathbf{o} \in \mathbb{R}^3$ and direction $\mathbf{d} \in \mathbb{R}^3$, we sample N points $\{\mathbf{r}(t_i)\}_{i=1}^N$ along the ray defined by a sorted distance vector $\mathbf{t} = [t_1, ..., t_N]^T \in \mathbb{R}^N$.

NeRF [31] models the 3D scene implicitly and leverage MLP networks to decode the density $\sigma_i = \texttt{MLP}(\mathbf{r}(t_i))$ and the view-dependent color $\mathbf{c}_i = \texttt{MLP}(\mathbf{r}(t_i), \mathbf{d})$ of a point located at $\mathbf{r}(t_i)$ on the ray with viewing direction $\mathbf{d}$. To render the image pixel $\hat{C}(\mathbf{r})$, we apply discretized volume rendering by Max [28] along the N sampled ray points with δ_i denoting the distance to the nearby sampled points:

$$\begin{aligned} \hat{C}(\mathbf{r}) &= \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i, \text{ where} \\ T_i &= \exp(-\sum_{j=1}^{i-1} \sigma_j \delta_j). \end{aligned} \tag{1}$$

Training such MLPs for 3D radiance field modeling requires observed images with known camera poses. Specifically, NeRF model is optimized by minimizing the average $\mathcal{L}_2$ distance between the rendered pixel color $\hat{C}(\mathbf{r})$ and the ground-truth pixel color $C(\mathbf{r})$:

$$\mathcal{L}_{color} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r} \in \mathcal{R}} \left\| \hat{C}(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2. \tag{2}$$

3.2. Model Formulation

Our representation disentangles the density and color of the scene and uses different modalities for modeling.

Density. Similar to DVGO [47], we adopt a voxel-grid representation to model the 3D density for an efficient query. Given a particular query point $\mathbf{p}_{xyz} \triangleq \mathbf{r}(t_i) \in \mathbb{R}^3$, we can obtain the corresponding density $\sigma \in \mathbb{R}$ via a trilinear interpolation, followed by a Softplus activation:

$$\sigma = \texttt{Softplus}(\texttt{GridSample}(\mathbf{p}_{xyz}, \phi_G)), \tag{3}$$

where ϕ_G denotes the one-channel voxel grid with learnable parameter ϕ_g at a voxel resolution size of $N_x \times N_y \times N_z$.

Concerning the fact that in 3D reconstruction, textures are applied to the surface of the mesh to provide visual details such as colors, patterns, and material properties [17, 51], we hope our model can obtain a coarsely accurate density estimation to perceive the 3D geometry at the early training stage, facilitating the subsequent learning of appearance aggregation. Hence, we opt for an explicit voxel-grid representation to achieve fast convergence instead of utilizing an implicit MLP such as NeRF [31]. Such a choice is also proven to be crucial by our experiments in Sec. 4.4.

Appearance. In order to empower explicit 3D editing capabilities, we formulate the color appearance by an explicit canonical image $\phi_I \in \mathbb{R}^{H\times W\times 3}$ plus an associated view-dependent implicit projection field $P(\cdot,\cdot) : (\mathbb{R}^3, \mathbb{R}^3) \to \mathbb{R}^2$, where H and W represent the image height and width, respectively. This formulation maps a given query point $\mathbf{p}_{xyz}$ in the 3D field with viewing direction $\mathbf{d}$ to the projected 2D point $\mathbf{p}_{uv}$ on the canonical image ϕ_I.

The design of the projection field is crucial to achieving naturalness and completeness in the learned canonical image. We model the projection field P in a residual manner, which consists of a non-learnable canonical projection initialization P_c and a projection offset learning P_o. The canonical projection initialization P_c plays a significant role in ensuring a natural-looking canonical image, while the projection offset P_o aims to address view-dependent effects and handle occlusions present in complex scenes.

Specifically, the canonical projection initialization $P_c(\cdot) : \mathbb{R}^3 \to \mathbb{R}^2$ projects the query 3D point $\mathbf{p}_{xyz}$, to an initial 2D projection point $\tilde{\mathbf{p}}_{uv}$ on the canonical image. Further elaboration on the selection of an appropriate projection initializing function P_c will be provided in Sec. 3.3. The projection offset $\Delta\mathbf{p}_{uv}$ is modeled by P_o with parameter weights ϕ_P:

$$\Delta\mathbf{p}_{uv} = P_o(\mathbf{p}_{xyz}, \mathbf{d}; \phi_P). \tag{4}$$

For simplicity, we omit the 3D positional encoding γ_p of $\mathbf{p}_{xyz}$ and the viewing direction encoding γ_d of $\mathbf{d}$ in Eq. (4). The final projection point $\mathbf{p}_{uv}$ can be derived as follows:

$$\mathbf{p}_{uv} = \tilde{\mathbf{p}}_{uv} + \Delta\mathbf{p}_{uv} = P_c(\mathbf{p}_{xyz}) + P_o(\mathbf{p}_{xyz}, \mathbf{d}; \phi_P). \tag{5}$$

The projection point $\mathbf{p}_{uv}$ is then used to query the RGB color $\mathbf{c} \in \mathbb{R}^3$ from the canonical image ϕ_I via interpolation:

$$\mathbf{c} = \mathtt{Sigmoid}(\mathtt{GridSample}(\mathbf{p}_{uv}, \phi_I)). \tag{6}$$

3.3. Canonical Projection Initialization

Learning a good projection field P for 3D-to-2D projection is indeed a non-trivial research challenge [10], particularly for 360-degree inward-facing scenes [3, 31]. Thus, we relax the 3D scene of interest in our approach, focusing on forward-facing scenes [30] and outward-facing panorama data [13]. The crucial step in achieving a natural-looking canonical image as the aggregated appearance is to position a pseudo canonical camera into the scene for a careful canonical projection initialization as P_c.

Forward-facing data and NDC representation. Similar to NeRF, we utilize the normalized device coordinate (NDC) space to model the real forward-facing captures. In perspective projection, the 3D points in the truncated pyramid frustum in camera (eye) coordinates are mapped to a cube $[-1, 1]^3$ in NDC space. The pseudo canonical camera is defined as the average camera pose positioned at the world origin. By doing so, the perspective projection f_p can map any given point $\mathbf{p}_{xyz}$ in the world coordinates to its corresponding NDC point $(p_{x'}, p_{y'}, p_{z'}) \triangleq f_p(\mathbf{p}_{xyz})$. The coordinate range of the voxel grid ϕ_G is defined by the bounding box in NDC space. Please refer to the original NeRF paper [31] for the detailed derivation of NDC.

One notable property of NDC is that for a ray $\mathbf{r}$ origin from the camera center, the resulting projected camera ray $\mathbf{r}'$ in NDC space is always perpendicular to the $x'y'$ plane. This implies that all points along $\mathbf{r}'$ share identical values for $p_{x'}$ and $p_{y'}$. Leveraging this property, we can have a simple and appropriate initialization for the projection field:

$$\tilde{p}_u = p_{x'}, \quad \tilde{p}_v = p_{y'}, \tag{7}$$

where we have the notation $(\tilde{p}_u, \tilde{p}_v) \triangleq \tilde{\mathbf{p}}_{uv}$.

Panorama data and contracted representation. The training images for panorama scenes are typically captured by cameras positioned around the global origin, with outward-facing views covering a 360-degree field of view. We choose to place the pseudo canonical camera at the global origin and define the Equirectangular projected image as a feasible canonical projection initialization.

Drawing inspiration from the smooth coordinate transforms in [3, 40] for 360-degree inward-facing unbounded data, we introduce a new contracted formulation f_c specifically for outward-facing (unbounded) panorama scenes:

$$f_c(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|}\left(1 - \frac{1}{\|\mathbf{x}\| + 1}\right), \tag{8}$$

where $\mathbf{x} \in \mathbb{R}^3$ is a 3D point in Euclidean space. This design transforms the points in such a way that they are distributed proportionally to the disparity in a unit sphere. Accordingly, the voxel grid ϕ_G is a cube with range $[-1, 1]^3$.

Using the contracted representation in Eq. (8), we have $(p_{x'}, p_{y'}, p_{z'}) \triangleq f_c(\mathbf{p}_{xyz})$. The canonical projection initialization P_c from $\mathbf{p}_{xyz}$ in 3D space to the 2D canonical image pixel $(\tilde{p}_u, \tilde{p}_v) \triangleq \tilde{\mathbf{p}}_{uv}$ can be formulated as follows:

$$\begin{aligned} \tilde{p}_u &= \tan^{-1}\left(\frac{p_{y'}}{p_{x'}}\right) \in [-\pi, \pi], \\ \tilde{p}_v &= \sin^{-1}(p_{z'}) \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]. \end{aligned} \tag{9}$$

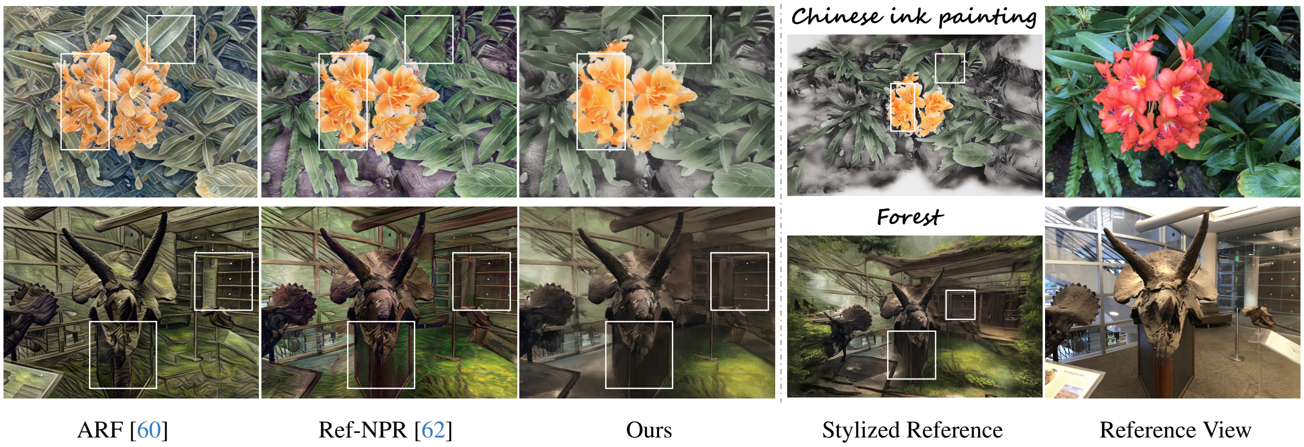


Figure 3. **Visual comparison of novel-view stylization results** on the LLFF dataset given different reference images. Our method can better preserve textural consistencies as highlighted in the figure.

3.4. Design and Regularization

Positional and hash encoding. The projection offset in Eq. (4) employs Fourier positional encoding or multi-resolution hash encoding to capture high-frequency information. For γ_d, we specifically employ positional encoding [48]; as for γ_p, we choose either positional encoding or hash encoding [32]. The positional encoding is defined as $\gamma_{pe}(\cdot) : \mathbb{R}^3 \to \mathbb{R}^{3\times(1+2K)}$ to encode 3-dimensional vector $\mathbf{x}$ up to K frequencies as $\gamma_{pe}(\mathbf{x}) = [\mathbf{x}, F_1(\mathbf{x}), ..., F_K(\mathbf{x})]$, where $F_k(\mathbf{x}) = [\sin(2^k\mathbf{x}), \cos(2^k\mathbf{x})]$; for hash encoding, we have $\gamma_h(\cdot) : \mathbb{R}^3 \to \mathbb{R}^{3+DK}$ to encode the vector $\mathbf{x}$ by a K-resolution hash grid with D-dimensional feature per layer as $\gamma_h(\mathbf{x}) = [\mathbf{x}, H_1(\mathbf{x}), ..., H_K(\mathbf{x})]$, where $H_k(\mathbf{x})$ is a D-dimensional feature vector interpolated by $\mathbf{x}$ at the k-th resolution.

Anneal encoding. Motivated by Nerfies [38], the positional or hash encoding can incorporate an optional annealing learning strategy. To achieve this, we introduce a weight factor $w_k^n = \frac{1}{2}(1 - \cos(\alpha_k^n \pi))$ for some encoded frequency F_k^n or H_k^n at some training step n, such that we have $F_k^n(\cdot) = w_k^n F_k(\cdot)$ and

$$\alpha_k^n = \min(\max(\frac{n - N_s}{N_e - N_s}K - k, 0.0), 1.0), \tag{10}$$

where N_s and N_e denote the start and end steps for anneal encoding, respectively. The strategy aims to facilitate the learning of low-frequency details and gradually incorporate high-frequency bands as the training progresses.

Progressive training. Similar to [3, 31, 47], we apply progressive scaling for our voxel grid ϕ_G and canonical image ϕ_I for a coarse-to-fine learning process. At specific scaling-up milestone steps, we increase the ϕ_G voxel count by a factor of 2 and the ϕ_I pixel count by a factor of 4.

Projection regularization. In order to obtain a visually natural canonical image ϕ_I, one important regularization is to avoid the deviation from the perception by the defined pseudo canonical camera. We find the following simple regularization works well and stabilizes the training:

$$\mathcal{L}_{uv} = \|\Delta \mathbf{p}_{uv}\|_2^2 . \tag{11}$$

Total variation regularization. To mitigate floating densities, we incorporate total variation regularization [41] $\mathcal{L}_{tv}$ into the density grid ϕ_G. This regularization term is particularly beneficial during the initial stages of training.

Optimization objective. The final optimization process of our method to model the scene for efficient editing can be formulated as follows:

$$\phi_G^*, \phi_I^*, \phi_P^* = \underset{\phi_G, \phi_I, \phi_P}{\arg\min}\, \mathcal{L}_{color} + \mathcal{L}_{uv} + \mathcal{L}_{tv}. \tag{12}$$

4. Experiments

4.1. Experimental Setup

Datasets. The *LLFF* [30] dataset comprises 8 real-world scenes, where each scene is accompanied by several training images captured by handheld cameras placed in a rough grid pattern. We evaluate this dataset at a resolution of 1008×756 after downscaling the images by a factor of 4.

The *Replica* [46] dataset is a collection of various high-quality and high-resolution 3D reconstructions of indoor scenes with clean and dense geometry. We evaluate the 14 panorama scenes as processed in SOMSI [13], where each scene is rendered as a grid of equally spaced spherical images at a resolution of 1024×512.

The *Instruct-NeRF2NeRF* [14] dataset consists of 6 scenes captured in either an object-centric or forward-facing manner. We leverage this dataset specifically to model and edit forward-facing scenes that prominently feature human subjects for qualitative evaluation.

Figure 4. **Visual comparison of scene stylization results** on the *Instruct-NeRF2NeRF* dataset given different text prompts.

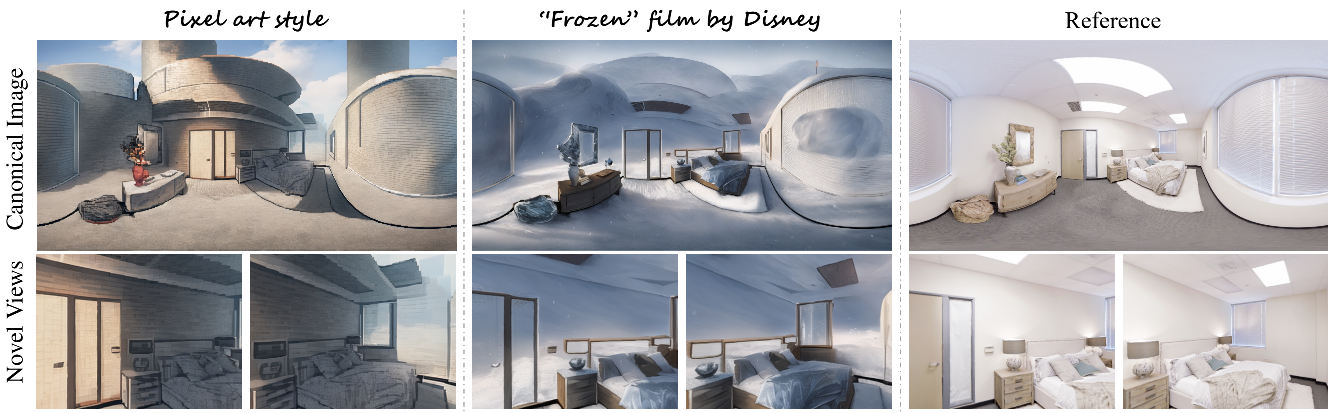


Figure 5. **More visualization of scene stylization results** on the *panorama Replica* dataset given different text prompts.

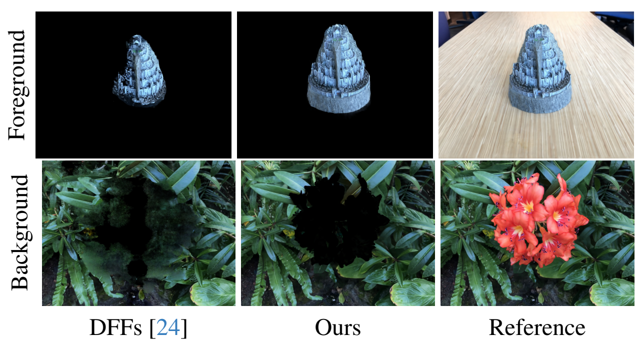


Figure 6. **Visual comparison of foreground and background extraction results** on the *LLFF* dataset.

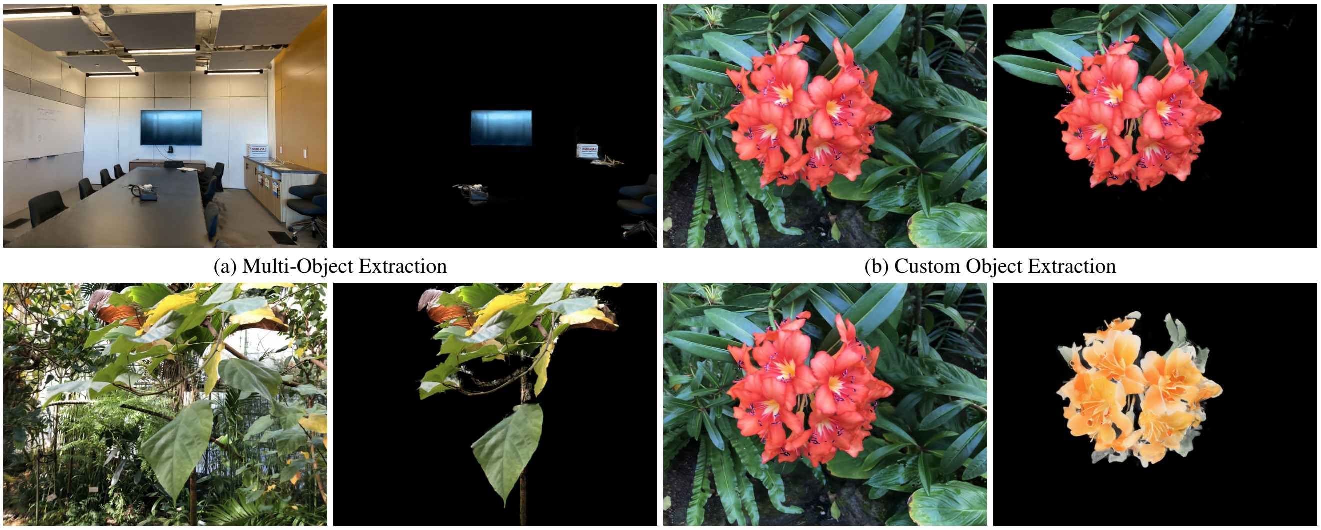
Implementation details. Our 3D editing pipeline involves a two-step process: training a per-scene reconstruction model and subsequent explicit edits on the canonical image ϕ_I for 3D scene editing. By default, we optimize a per-scene model for 60k steps using the Adam optimizer [22] with an initial learning rate of 0.1 for both the explicit 3D density grid ϕ_G and 2D canonical image ϕ_I, and a learning rate of 0.001 for the implicit projection field P with learnable parameter ϕ_P. All experiments are conducted and tested on a single RTX A6000 GPU. For further details and hyperparameters used in training different models, please refer to the supplementary materials.

4.2. Evaluation on Editability

By performing explicit edits on the canonical image ϕ_I, our model is capable of propagating the editing effects through the learned projection field P. This enables various 3D scene editing functionalities, such as scene stylization, context extraction, and texture editing. Specifically, except for explicit texture editing, we leverage several state-of-the-art models for 2D image editing. Specifically, we utilize the prompt-guided *ControlNet* [61] for scene stylization and *Segment-anything (SAM)* [23] for content extraction.

Scene stylization. We conduct a comparative analysis between our method and three state-of-the-art stylization methods: ARF [60], Ref-NPR [62], and Instruct-NeRF2NeRF [14]. All three baseline methods require additional optimization processes to achieve stylization with a specific style, somehow limiting their model practicability. Specifically, ARF and Ref-NPR rely on one or multiple stylized reference images, while Instruct-NeRF2NeRF utilizes text prompts through Diffusion models [4] for guidance.

In Fig. 3, we demonstrate some comparing visualizations evaluated on the LLFF dataset, where we can see that both ARF and Ref-NPR cannot edit the 3D scene to be as visually consistent as our method with respect to the given reference style image. This behavior can be attributed to ARF and Ref-NPR's approach of adjusting NeRF model weights through loss functions to fit the given style, resulting in implicit control over the final edited style of the scene. In contrast, by editing the explicit canonical RGB



(a) Multi-Object Extraction
(b) Custom Object Extraction
(c) Complex Object Extraction
(d) Extraction + Stylization

Figure 7. **More visualization of content extraction results** in different complex settings on the *LLFF* dataset.

Edited Novel View 1 | Edited Novel View 2 | Edited Novel View 3 | Reference

Figure 8. **Visualization of texture editing (*i.e.*, drawing) results** rendered at different novel viewpoints on the *LLFF* dataset.

image ϕ_I of our method, we can propagate the appearance to the 3D color field through the projection function P, enabling explicit appearance control for editing. We further showcase some visual results of the Instruct-NeRF2NeRF dataset using different text prompts for the third baseline in Fig. 4. Although both of our methods can successfully edit the scene into the desired style, optimizing the NeRF model using the Instruct-NeRF2NeRF method takes more than 10 hours to achieve stationary performance, while our method requires no extra re-optimization. Lastly, we present more visual editing results on the panorama Replica data in Fig. 5.

Content extraction. We evaluate our method with the state-of-the-art DFFs [24] method, which enables NeRFs to decompose a specific object given a text or image-patch query. In Fig. 6, we show some comparing visualization of foreground and background extraction. The evaluation of baseline DFFs is only based on text query according to its official codebase. As shown in Fig. 7, we demonstrate more visualization with various extraction goals, including multi-object extraction, custom-design content extraction, and complex object extraction, and we can even do stylization specifically on the extracted object.

Texture editing. We further present an additional application to do textural appearance editing of the scene by drawing or painting the canonical image. In Fig. 8, we can observe that our method ensures both appearance and 3D consistency in novel views.

Table 1. **The reconstruction results** on the *LLFF* dataset in terms of PSNR. Our PE-based models, which exhibit superior editing capacity, demonstrate satisfactory performance, while our hash-based models achieve reconstruction comparable to NeRF [31] and DVGO [47].

	Room	Fern	Leaves	Fortress	Orchids	Flowers	T-Res	Horns	Average
LLFF [30]	28.42	22.85	18.52	29.40	18.52	25.46	24.15	24.70	24.13
NeRF [31]	32.70	25.17	20.92	31.16	20.36	27.40	26.80	27.45	26.50
DVGO [47]	31.43	25.08	21.03	30.47	20.37	27.59	27.17	27.56	26.34
Ours (PE)	30.22	22.77	19.96	29.15	18.39	26.38	25.85	25.97	24.83
Ours (Hash)	32.10	24.13	20.64	30.12	20.10	27.47	27.24	27.81	26.20

Table 2. **The reconstruction results** on the *Replica* dataset.

	PSNR ↑	SSIM ↑
SOMSI [13]	39.54	0.986
Ours (PE)	38.42	0.979
Ours (Hash)	38.68	0.976

Table 3. **Ablation studies** of model components on *trex* scene.

Settings	PE	Hash
I. Implicit density grid ϕ_G	19.29	18.22
II. Implicit canonical image ϕ_I	22.53	22.17
III. No initialization P_c	23.18	27.56
IV. No offset P_o	23.81	23.88
V. No viewdir in offset P_o	25.50	26.76
Full model	25.85	27.24

4.3. Trade-off between Fidelity and Editability.

As a neural editing method without the need for re-optimization, we compensate reconstruction quality for editing capacity. In this section, we examine the reconstruction capacity of our method by trying different designs for learning the projection offset P_o, whose inputs are the query 3D position $\mathbf{p}_{xyz}$ and its viewing direction $\mathbf{d}$. Specifically, our findings indicate that utilizing a positional encoding (PE) for position leads to superior editing capacity, whereas models with hash encoding exhibit higher reconstruction performance compared to the PE-based models. The corresponding PSNR results are shown in Tabs. 1 and 2.

4.4. Ablation Study

Additionally, we conduct ablation studies to evaluate the effectiveness of different model components in terms of both editability and reconstruction fidelity. Specifically, we choose the *trex* scene in the LLFF dataset as our evaluation. The corresponding reconstruction statistics in terms of PSNR are shown in Tab. 3.

Explicitness and implicitness. Setting I and II replace the explicit representation of the density grid ϕ_G and the canonical image ϕ_I with implicit representations modeled by MLP networks, respectively, where their statistical result drop significantly. Moreover, in setting 1, it fails to learn accurate geometries for reconstruction, and in setting 2, it fails to learn a satisfying canonical image for editing purposes. Based on these findings, we decide to utilize explicit representations for both the density grid ϕ_G and the canonical image ϕ_I.

Canonical projection initialization. Setting III ablates the canonical projection initialization P_c. The performance without P_c is quite tricky, where the PE-based model experiences a drop while the hash-based model shows an increase. Nonetheless, both models lose the ability to perform editing tasks as the learned canonical images ϕ_I deviate from natural images to latent color maps.

Learnable projection offset. In Setting IV, the removal of view-dependence from the learnable projection offset P_o leads to a minor decrease in performance, as the model no longer considers viewing directions. In Setting V, where the entire projection offset P_o is eliminated, a significant drop in performance is observed. More visual analyses are shown in the supplementary materials.

5. Discussion and Conclusion

The key to aggregating 3D appearance as an image for editing is to find a natural mapping from 3D points to 2D pixels. While such a mapping is non-trivial when it comes to 360-degree inward-facing scenes, and hence we mainly explore the potential of our method with forward-facing and panorama outward-facing data. As the field of neural scene representation progresses, we anticipate future work can successfully address this issue.

In summary, we propose `AGAP`, an editing-friendly and efficient solution for neural 3D scene editing. We leverage an explicit 2D appearance aggregation to replace the color field in NeRFs. Such a design allows users to directly perform 2D edits on images, eliminating the need for dealing with geometries or implicit fields, thus simplifying the editing process. Our approach showcases superior editing results on multiple datasets without any laborious re-optimization procedure.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In *Proceedings of ICML*, 2018. 2

[2] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of ICCV*, 2021. 2

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of CVPR*, 2022. 2, 4, 5, 13

[4] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of CVPR*, 2023. 2, 6

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of ICCV*, 2021. 2

[6] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of CVPR*, 2022. 2

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

[8] Jun-Kun Chen, Jipeng Lyu, and Yu-Xiong Wang. NeuralEditor: Editing neural radiance fields via manipulating point clouds. In *Proceedings of CVPR*, 2023. 2

[9] Yu Deng, Jiaolong Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *Proceedings of CVPR*, 2021. 2

[10] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic Parameterizations of Surface Meshes. *Computer Graphics Forum*, 2002. 4

[11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of CVPR*, 2022. 2

[12] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A style-based 3d aware generator for high-resolution image synthesis. In *Proceedings of ICLR*, 2022. 2

[13] Tewodros Habtegebrial, Christiano Couto Gava, Marcel Rogge, Didier Stricker, and Varun Jampani. SOMSI: spherical novel view synthesis with soft occlusion multi-sphere images. In *Proceedings of CVPR*, 2022. 2, 4, 5, 8, 13

[14] Ayaan Haque, Matthew Tancik, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of ICCV*, 2023. 2, 5, 6, 13

[15] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *Proceedings of CVPR*, pages 2821–2830, 2018. 2

[16] Yihua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. Stylizednerf: Consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. In *Proceedings of CVPR*, 2022. 2

[17] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *Proceedings of ICCV*, 2017. 2, 4

[18] James T. Kajiya and Brian Von Herzen. Ray tracing volume densities. In *Proceedings of SIGGRAPH*, pages 165–174, 1984. 3

[19] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of ECCV*, 2018. 2

[20] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. *TOG*, 40(6):210:1–210:12, 2021. 2

[21] Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of SGP*, 2006. 2

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015. 6, 12

[23] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *CoRR*, abs/2304.02643, 2023. 6

[24] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *Advances in NeurIPS*, 2022. 2, 6, 7

[25] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: bundle-adjusting neural radiance fields. In *Proceedings of ICCV*, 2021. 2

[26] Yen-Chen Lin, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *Proceedings of IROS*, 2021. 2

[27] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *TPAMI*, 38(10):2024–2039, 2016. 2

[28] Nelson L. Max. Optical models for direct volume rendering. *TVCG*, 1(2):99–108, 1995. 3

[29] Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of CVPR*, 2019. 2

[30] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: practical view synthesis with prescriptive sampling guidelines. *TOG*, 38 (4):29:1–29:14, 2019. 2, 4, 5, 8, 13

[31] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf:

Representing scenes as neural radiance fields for view synthesis. In *Proceedings of ECCV*, pages 405–421, 2020. 1, 2, 3, 4, 5, 8, 13

[32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *TOG*, 41(4):102:1–102:15, 2022. 2, 5, 13

[33] Thu Nguyen-Phuoc, Feng Liu, and Lei Xiao. Snerf: stylized neural implicit representations for 3d scenes. *TOG*, 41(4): 142:1–142:11, 2022. 2

[34] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *CoRR*, arXiv/2304.07193, 2023. 2

[35] Hao Ouyang, Qiuyu Wang, Yuxi Xiao, Qingyan Bai, Juntao Zhang, Kecheng Zheng, Xiaowei Zhou, Qifeng Chen, and Yujun Shen. Codef: Content deformation fields for temporally consistent video processing. *CoRR*, abs/2308.07926, 2023. 2, 3

[36] Hong-Wing Pang, Binh-Son Hua, and Sai-Kit Yeung. Locally stylized neural radiance fields. *CoRR*, abs/2309.10684, 2023. 2

[37] Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of CVPR*, pages 165–174, 2019. 2

[38] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of ICCV*, pages 5845–5854, 2021. 3, 5, 13

[39] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: a higher-dimensional representation for topologically varying neural radiance fields. *TOG*, 40(6):238:1–238:12, 2021. 3

[40] Christian Reiser, Richard Szeliski, Dor Verbin, Pratul P. Srinivasan, Ben Mildenhall, Andreas Geiger, Jonathan T. Barron, and Peter Hedman. MERF: memory-efficient radiance fields for real-time view synthesis in unbounded scenes. *TOG*, 42(4):89:1–89:12, 2023. 4

[41] Leonid Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992. 5

[42] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of CVPR*, 2016. 2

[43] Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Proceedings of ECCV*, 2016. 2

[44] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: generative radiance fields for 3d-aware image synthesis. In *Advances in NeurIPS*, 2020. 2

[45] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of CVPR*, 2019. 2

[46] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Yuheng Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard A. Newcombe. The replica dataset: A digital replica of indoor spaces. *CoRR*, abs/1906.05797, 2019. 2, 5, 13

[47] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of CVPR*, pages 5449–5459, 2022. 2, 3, 5, 8, 13

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in NeurIPS*, pages 5998–6008, 2017. 5, 13

[49] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of CVPR*, 2022. 2

[50] Jinglu Wang, Bo Sun, and Yan Lu. Mvpnet: Multi-view point regression networks for 3d object reconstruction from A single image. In *Proceedings of AAAI*, 2019. 2

[51] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single RGB images. In *Proceedings of ECCV*, 2018. 2, 4

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

[53] Qiuyu Wang, Zifan Shi, Kecheng Zheng, Yinghao Xu, Sida Peng, and Yujun Shen. Benchmarking and analyzing 3d-aware image synthesis with a modularized codebase. In *Advances in NeurIPS*, 2023. 2

[54] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. Nerf-: Neural radiance fields without known camera parameters. *CoRR*, abs/2102.07064, 2021. 2

[55] Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. Neutex: Neural texture mapping for volumetric neural rendering. In *Proceedings of CVPR*, 2021. 2

[56] Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *Proceedings of ECCV*, 2022. 2

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

[58] Vickie Ye, Zhengqi Li, Richard Tucker, Angjoo Kanazawa, and Noah Snavely. Deformable sprites for unsupervised video decomposition. In *Proceedings of CVPR*. IEEE, 2022. 2

[59] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: Geometry editing of neural radiance fields. In *Proceedings of CVPR*, 2022. 2

[60] Kai Zhang, Nicholas I. Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. ARF: artistic radiance fields. In *Proceedings of ECCV*, 2022. 2, 5, 6

[61] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of ICCV*, 2023. 6

[62] Yuechen Zhang, Zexin He, Jinbo Xing, Xufeng Yao, and Jiaya Jia. Ref-npr: Reference-based non-photorealistic radiance fields for controllable scene stylization. In *Proceedings of CVPR*, 2023. 2, 5, 6

[63] Yanshu Zhang, Shichong Peng, Alireza Moazeni, and Ke Li. PAPR: proximity attention point rendering. In *Advances in NeurIPS*, 2023. 2

Appendix

A. Ablation Analysis

The ablation studies (settings I to V) presented in the main paper are conducted for both PE and hash models on the *trex* scene in the LLFF dataset. In this section, we supply the ablation studies with more corresponding visual analysis, offering more detailed insights. Specifically, we present the reconstruction PSNR statistics in Tab. A1, which match those reported in the main paper. For visual results, Fig. A1 shows the learned canonical image ϕ_I and the novel viewpoint image with the pseudo canonical camera pose plus its corresponding depth map for each ablation setting. The last row results in Tab. A1 and Fig. A1 indicate that the PE model achieves a satisfactory reconstruction and a superior canonical image ϕ_I for explicit editing, whereas the hash model excels in terms of reconstruction quality.

Setting I: Replace the explicit density grid ϕ_G with an implicit one modeled by MLP networks. The learning rate for the replaced implicit density field is set to 10^{-3}. We can see from Tab. A1 that the reconstruction qualities drop significantly as they fail to learn correct depths as shown by subfigure I-C and I-F in Fig. A1.

Setting II: Replace the explicit canonical image ϕ_I with an implicit one modeled by MLP networks. The learning rate for the replaced implicit canonical field is also configured to be 10^{-3}. Based on the findings in Tab. A1, the reconstruction performance of both the PE and hash models experiences degradation of more than 3 dB and 5 dB, respectively. Further examination of row II in Fig. A1 reveals that while both models demonstrate satisfactory depth map learning capabilities, they exhibit limited capacities to generate high-quality canonical images for editing purposes.

Table A1. **Ablation studies** of model components on *trex* scene. Same as in the main paper.

Settings	PE	Hash
I. Implicit density grid ϕ_G	19.29	18.22
II. Implicit canonical image ϕ_I	22.53	22.17
III. No initialization P_c	23.18	27.56
IV. No offset P_o	23.81	23.88
V. No viewdir in offset P_o	25.50	26.76
Full model	25.85	27.24

Setting III: Remove the canonical project initialization P_c in the projection field P. Although in Tab. A1 for setting III, we find that the PE model achieves satisfactory performance and the hash model even performs better than our full hash model in terms of PSNR, the learned canonical images (*i.e.*, subfigure III-A and III-D in Fig. A1) are optimized to be latent color representations instead of natural images, resulting in a loss of editability.

Setting IV: Remove the project offset P_o in the projection field P. Recall that the learned canonical image ϕ_I is determined by placing a pseudo canonical camera at the global origin. By comparing subfigure IV-A and Full-A of the PE models, we can see that without having the projection offset, the canonical image (subfigure IV-A) cannot "look through" the foreground ceiling, sharing similar behaviors as the canonical view (column B). However, by adding the projection offset, we enable the capacity of our method to handle occlusions. As demonstrated in subfigure Full-A, the final canonical image ϕ_I generated by our full models learns to naturally aggregate the occluded background ceiling within the image by leveraging the projection offsets from 3D points to 2D pixels. Please also refer to the corresponding zoomed-in patches in Fig. A2.

Setting V: Remove view dependency in the projection offset P_o of the projection field P. The omission of viewing directions as the input for projection offset P_o, as observed from the last two rows in Tab. A1 and Fig. A1, leads to a marginal decrease in reconstruction quality and yields slightly inferior canonical images.

B. Training Details

Our 3D editing pipeline involves a two-step process: (1) we first train a per-scene reconstruction model using the proposed `AGAP` representation, which includes an explicit density grid ϕ_G, an explicit canonical image ϕ_I, and an associated projection field ϕ_P; (2) we can then perform explicit 2D edits on the canonical image ϕ_I for 3D scene editing, including scene stylization, content extraction, and texture editing. All experiments, including training on various scenes from different datasets, are conducted and tested on a single RTX A6000 GPU, with specific hyperparameter

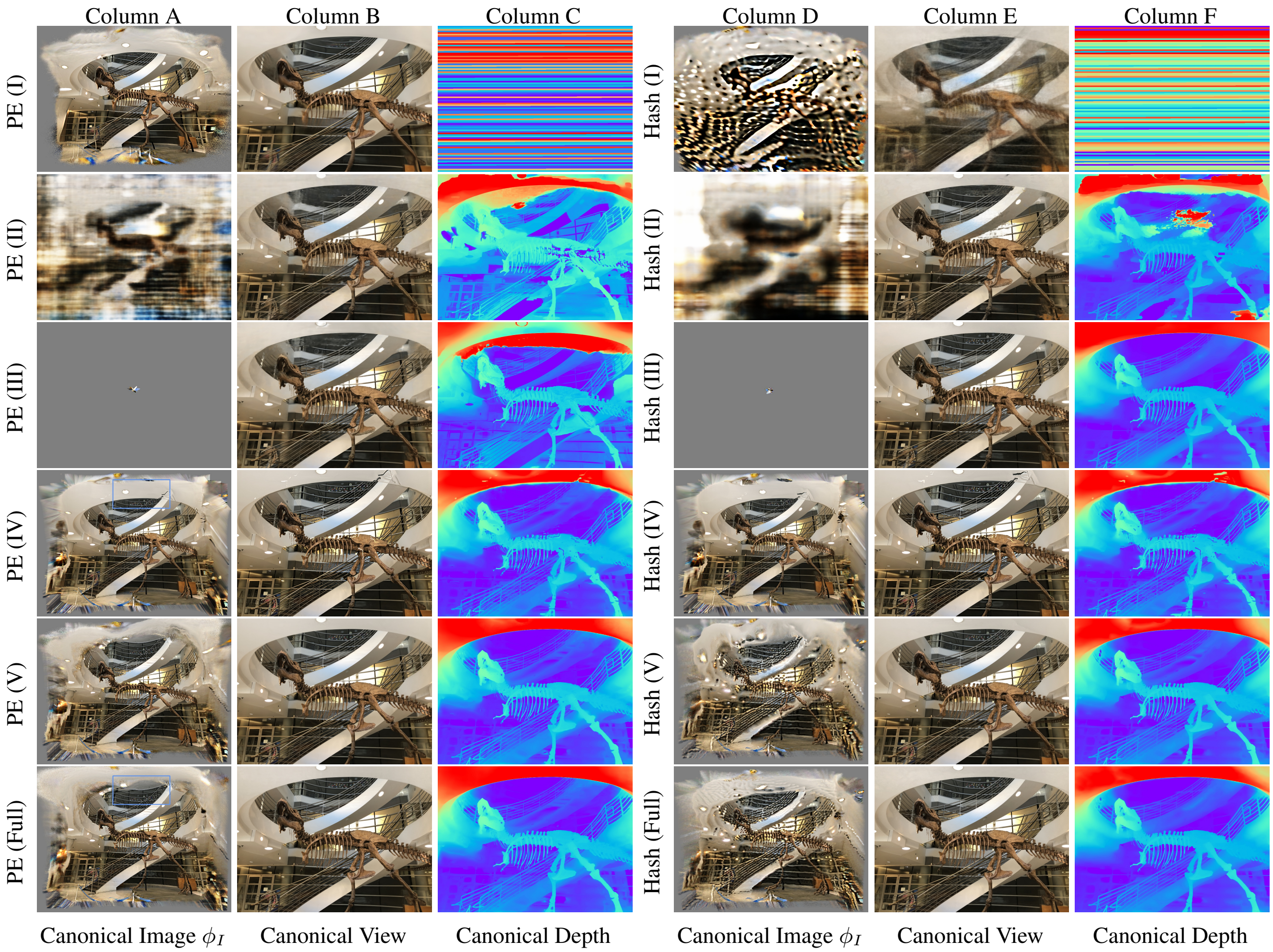


Figure A1. **Visualization analysis** of the ablation studies for both PE and hash models conducted on the *trex* scene in the LLFF dataset. Columns A and D are the learned canonical image ϕ_I. Columns B and E are the rendered novel view images at the canonical position (global origin) with the canonical pose (identity rotation). Columns C and F are the corresponding depth maps of the rendered novel views. The zoomed-in patches for subfigure IV-A and Full-A are shown in Fig. A2.

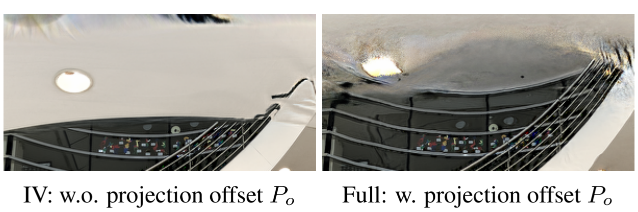


Figure A2. **Zoomed-in patches** of the canonical images ϕ_I of the PE model with setting IV (subfigure IV-A) and our full PE model (subfigure Full-A). Without the projection offset, the canonical image fails to include the occluded background ceiling information. Our full model can handle occlusion and naturally aggregate the background ceiling inside the canonical image.

details outlined in Tab. A2.

Optimization. In the first stage, we employ the Adam optimizer [22] to optimize a per-scene model for 60k steps with an initial learning rate of 0.1 for both the explicit 3D density grid ϕ_G and 2D canonical image ϕ_I, and a learning rate of 0.001 for the implicit projection field P with learnable parameter ϕ_P. The optimization of the entire model involves an objective function comprising three main components: (1) an average $\mathcal{L}_2$ photometric loss $\mathcal{L}_{color}$ between the rendered pixel color $\hat{C}(\mathbf{r})$ and the ground-truth color $C(\mathbf{r})$; (2) a projection regularization $\mathcal{L}_{uv}$ aimed at minimizing the projection offset $\Delta\mathbf{p}_{uv}$; and (3) a total variation regularization applied to the density grid ϕ_G.

Weight factor. As stated in the main paper, the final optimization process of our method to model the scene for efficient editing can be formulated as follows:

$$\phi_G^*, \phi_I^*, \phi_P^* = \underset{\phi_G, \phi_I, \phi_P}{\arg\min}\ \mathcal{L}_{color} + \mathcal{L}_{uv} + \mathcal{L}_{tv}, \tag{A1}$$

Table A2. **Hyperparameters** for training various scenes in the LLFF [30], Replica [13, 46], and InstructNeRF2NeRF (IN2N) [14] datasets.

	Common						PE Models					Hash Models			
	Image Size	Weight Factor		Direction $\mathbf{d}$			Position $\mathbf{p}_{xyz}$					Position $\mathbf{p}_{xyz}$			
		λ_{uv}	λ_{tv}	Type	K	Anneal	Type	K	Anneal	N_s	N_e	Type	D	K	Anneal
LLFF	(768, -)	10^{-5}	10^{-5}	PE	4	✗	PE	8	✓	4000	8000	Hash	2	16	✗
Replica	(768, 1536)	10^{-1}	10^{-4}	PE	4	✗	PE	8	✓	4000	8000	Hash	2	16	✗
IN2N	(768, -)	10^{-5}	10^{-5}	PE	4	✗	PE	8	✓	4000	8000	Hash	2	16	✗

where the second and the third terms are controlled by their corresponding weight factors λ_{uv} and λ_{tv}, respectively. To be specific, the weight factor λ_{uv} is set as 10^{-5} for forward-facing scene and 10^{-1} for panorama scenes; the weight factor λ_{tv} is set as 10^{-5} for forward-facing scene and 10^{-4} for panorama scenes. Note that for panorama data, we set both the projection and total variation regularization to be relatively large because the depths of indoor scenes in the Replica dataset are sometimes ambiguous due to large walls in the background, where the total variation term is disabled after 20000 steps to learn depths in details.

Progressive training. Similar to [3, 31, 47], we apply progressive scaling for our voxel grid ϕ_G and canonical image ϕ_I for a coarse-to-fine learning process. By gradually refining the resolution of both representations, we enable a more detailed and comprehensive learning process.

At specific scaling-up milestone steps, we increase the ϕ_G voxel count by a factor of 2 and the ϕ_I pixel count by a factor of 4. For the forward-facing datasets (*i.e.*, LLFF and InstructNeRF2NeRF), the voxel grid ϕ_G is scaled up at $\{2000, 4000, 6000, 8000\}$ training steps and the canonical image ϕ_I is scaled up at $\{8000, 16000\}$ training steps. For the panorama Replica dataset, the voxel grid ϕ_G is scaled up at $\{2000, 4000, 6000, 8000, 10000, 12000, 14000, 16000\}$ training steps, and the canonical image ϕ_I is scaled up at $\{4000, 8000, 12000, 16000\}$ training steps.

Size of voxel grid ϕ_G and canonical image ϕ_I. After the progressive scaling up, The final resolution of the voxel grid ϕ_G is set as $384 \times 384 \times 256$ for forward-facing scenes and $320 \times 320 \times 320$ for panorama scenes.

In all the experiments, we set the final height H_I of the learnable explicit canonical image ϕ_I as 768. For panorama data, the canonical image width W_I is set to be 1536 according to the definition of Equirectangular projection. For forward-facing data, the canonical image width W_I is adaptively calculated according to the width-height aspect ratio of the training images and the computed bounding box of the scene in NDC space. Denoting the bounding box in NDC space as (x'_{min}, x'_{max}) in x' dimension, (y'_{min}, y'_{max}) in y' dimension, and $(z'_{min}, z'_{max}) = (-1, 1)$ in z' dimension and the aspect ratio as r_I, we can then calculate the canonical image width as:

$$W_I = H_I \times r_I \times \frac{x'_{max} - x'_{min}}{y'_{max} - y'_{min}}. \tag{A2}$$

Annealed positional and hash encoding. The projection offset employs Fourier positional encoding [48] or multi-resolution hash encoding [32] to capture high-frequency information. Given an input vector $\mathbf{x} \in \mathbb{R}^3$, the corresponding encoding can be defined as follows:

- The positional encoding is defined as $\gamma_{pe}(\cdot) : \mathbb{R}^3 \rightarrow \mathbb{R}^{3\times(1+2K)}$ to encode 3-dimensional vector $\mathbf{x}$ up to K frequencies as $\gamma_{pe}(\mathbf{x}) = [\mathbf{x}, F_1(\mathbf{x}), ..., F_K(\mathbf{x})]$. For the k-$th$ frequency of positional encoding, we have the encoding function $F_k(\mathbf{x}) = [\sin(2^k\mathbf{x}), \cos(2^k\mathbf{x})] \in \mathbb{R}^{2\times 3}$.
- The hash encoding is defined as $\gamma_h(\cdot) : \mathbb{R}^3 \rightarrow \mathbb{R}^{3+DK}$ to encode the vector $\mathbf{x}$ by a K-resolution hash grid with D-dimensional feature per layer as $\gamma_h(\mathbf{x}) = [\mathbf{x}, H_1(\mathbf{x}), ..., H_K(\mathbf{x})]$. For the k-th resolution hash grid with D-dimensional feature at each layer, we have the encoding function $H_k(\mathbf{x}) \in \mathbb{R}^D$.

Motivated by Nerfies [38], the positional or hash encoding can incorporate an optional annealing learning strategy. Specifically, we introduce a weight factor $w_k^n = \frac{1}{2}(1 - \cos(\alpha_k^n \pi))$ for some encoded frequency F_k^n or H_k^n at some training step n, such that we have $F_k^n(\cdot) = w_k^n F_k(\cdot)$ or $H_k^n(\cdot) = w_k^n H_k(\cdot)$ and

$$\alpha_k^n = \min(\max(\frac{n - N_s}{N_e - N_s} K - k, 0.0), 1.0), \tag{A3}$$

where N_s and N_e denote the start and end steps for anneal encoding, respectively. The strategy aims to facilitate the learning of low-frequency details and gradually incorporate high-frequency bands as the training progresses.

For all the experiments, the encoding γ_d of direction $\mathbf{d}$ specifically employs positional encoding γ_{pe}, where we set $K = 4$ with the optional annealing learning strategy off. Concerning the encoding γ_p of position $\mathbf{p}_{xyz}$, we choose to use positional encoding γ_{pe} for PE models and hash encoding γ_h for hash models, where we set $K = 8$ with the annealed learning starting at training step $N_s = 4000$ and ending at $N_e = 8000$ for PE models, and we set $D = 2$ and $K = 16$ without the optional annealed learning strategy for hash models.