4D-Editor: Interactive Object-level Editing in Dynamic Neural Radiance Fields via 4D Semantic Segmentation

Dadong Jiang* Zhihui Ke* Xiaobo Zhou†
Xidong Shi
College of Intelligence and Computing, Tianjin University
{patrickdd, kezhihui, xiaobo.zhou, suif}@tju.edu.cn

Figure 1. 4D-Editor can interactively edit objects in a dynamic NeRF. For example, with strokes drawn on a reference frame by users, 4D-Editor can remove the human, recolor and shift the balloon within the dynamic NeRF. After the novel view synthesis, After the novel view synthesis, the women disappears, the balloon is changed into purple and shifted in both spatial and temporal dimensions.

Abstract

This paper targets interactive object-level editing(e.g., deletion, recoloring, transformation, composition) in dynamic scenes. Recently, some methods aiming for flexible editing static scenes represented by neural radiance field (NeRF) have shown impressive synthesis quality, while similar capabilities in time-variant dynamic scenes remain limited. To solve this problem, we propose 4D-Editor, an interactive semantic-driven editing framework, allowing editing multiple objects in dynamic NeRF based on user strokes on a single frame. Our dynamic scene representation is built upon hybrid semantic feature fields so that the spatial-temporal consistency can be maintained after editing. In addition, we design recursive selection refinement that significantly boosts segmentation accuracy in a dynamic NeRF to aid the editing process. Moreover, we develop multi-view reprojection inpainting to fill holes caused by incomplete scene capture after editing. Extensive experiments and editing examples on real-world demonstrate that 4D-Editor achieves photo-realistic dynamic NeRF editing. Project page: https://patrickddj.github.io/4D-Editor

∗ Equal Contribution
† Corresponding Author

1. Introduction

With the popularity of Neural Radiance Field (NeRF [27]) and its extensions [6,9,12,16,25,31,33,39,41] enabling observation on dynamic real-world scenes in a free-viewpoint way, there is a critical demand for editing NeRF to support downstream applications, e.g., VR/AR, computer animation, and education. Semantic-NeRF [54] directly use existing semantic labels to supervise NeRF for object-level scene editing, while it requires mutual labels. Therefore, some recent studies [13, 17, 42] adopt semantic distillation to learn 3D semantic features from large pre-trained models like DINO [5] or LSeg [20], which can generate open-vocabulary scene semantic labels that are viewed as priors. Despite these methods can achieve fine editing results in 3D static NeRFs, they are constrained to edit one 4D dynamic NeRF due to its spatial-temporal variant.

Our goal is to interactively edit target objects within a dynamic NeRF. The closest work to ours is NeuPhysics [35], which allows for partially editing dynamic scenes. It simply decomposes a scene into the dynamic foreground and the static background, which means that the foreground and background are viewed as a whole, respectively, resulting in only editing the entire foreground or background, and cannot edit an object in the scene (*e.g.*, recolor a street sign in a

static background or remove one specific car when existing multiple moving cars). Moreover, NeuPhysics relies on a mesh proxy for editing and does not provide any editing interface, which is user-friendliness. The differences between our work and existing works are listed in Table 1.

In this paper, we propose an interactive object-level editing framework for dynamic NeRFs vis semantic distillation, named 4D-Editor, which allow users to select different objects by strokes in a image of the dynamic scene and edit them, then the editing operation will spread to the entire dynamic NeRF. However, it is a challenge to maintain multi-view consistency and time-variant consistency, after we apply the editing of a observation view to the entire dynamic NeRF. Therefore, 4D-Editor employs Hybrid Semantic Feature Distillation to include semantic information at both static field and dynamic field from a pre-trained DINO [5] and support interactive object-level editing operations through matching 2D user strokes and 4D semantic features. Nevertheless, due to few user strokes and limited observation views, it is difficult to precisely segment edited objects in dynamic NeRFs and fill *holes* with spatial-temporal consistency after editing. To solve above challenges, we propose Recursive Selection Refinement method to ensure accurate matching of all target areas while preserving unrelated areas. Moreover, Multi-view Reprojection Inpainting strategy is proposed to complete the holes through reprojection of known views from multiple perspectives views and inpainting of invisible parts with spatial-temporal consistency by performing inpainting operations. The editing process takes only about 2 seconds to generate a novel modified view.

We summarize our contributions as follows:

- We propose 4D-Editor, to our knowledge, the first interactive editing framework which is used to edit multiple objects within dynamic NeRFs by user strokes on 2D images, while maintaining spatial-temporal consistancy. 4D-Editor enables diverse editing operations, e.g., remove, recolor, and transform.
- We propose hybrid semantic feature field which incorporates DINO semantic features through hybrid semantic feature distillation for maintaining spatial-temporal consistency.
- We also propose recursive selection refinement, an accurate object segmentation method, which provide a rapid but precise selection of edited objects in dynamic NeRF in a recursive manner. Our experimental results demonstrate its accuracy and efficiency.
- multi-view reprojection inpainting is developed to fill holes after editing, in invisible areas due to incomplete scene capture caused by sparse views.

Method	Dynamic	Interactive	Object-Level
Ours	✓	✓	✓
NeuPhysics [35]	✓	✗	✗
N3F [42]	✓	✗	✓
ISRF [13]	✗	✓	✓

Table 1. Comparison with prior methods.

2. Related work

Dynamic Scene Representations. In the past few years, dynamic scene representations [4, 8, 11, 16, 19, 21, 23, 25, 41, 45, 46, 49, 51] have experienced great development both in reconstruction quality and training speed. DyNeRF [21] is an early work in this area, where expressive time-variant latent code is introduced into implicit volume representation to reconstruct a dynamic scene. However, the training speed of DyNeRF is extremely slow, needs about 7 days. Thus, recent explicit volume representation methods like K-Planes [8] and D-TensoRF [16] use multiple low-dimension planes or vectors to represent the 4D dynamic scene, thereby greatly accelerating training and rendering speed. Different from directly learning dynamic scene representation, DynamicNeRF [51], RobustNeRF [25], and MixVoxels [45] divide the scene into static and dynamic parts and model them separately, hence they are called hybrid representation.

NeRF Editing. The direct way to edit NeRF is to bake NeRF into textured mesh [24, 47, 48, 52], thereby we can use existing 3D editing tools(e.g, Blender [3]) to edit textures. However, it is challenge to bake large scenes with high quality and is limited to reconstruct static scenes. With the development of language-guided pre-trained models, researchers utilize CLIP model [36] or diffusion model [15, 26, 30, 32, 37, 53] to edit NeRF by text prompt and image patch. However, this method hard to maintain multi-view consistency due to these pre-trained models are training by 2D images. Besides, researchers also proposed blending editing methods [2, 14], which blend the output of an auxiliary editing field and original NeRF. And using inpainting model [22, 28, 29, 44, 50] to fill holes after object removal. However, all above methods can only edit static NeRF. Regarding editing of dynamic NeRF, NeuPhysics [35] combines time-invariant signed distance function(SDF) with a deformation field to reconstruct dynamic scenes, but only support edit foreground dynamic objects. Recently, N3F [42] and ISRF [13] treat pre-trained semantic models as a teacher and distill 2D semantic features from a teacher model into NeRF, thereby enabling 3D object segmentation and editing in 3D space. However, N3F not only does not support interactive object-level editing, but also the editing performance is poor in a dynamic NeRF due to imprecise object segmentation. While our proposed 4D-Editor can interactively edit multi-objects in dynamic NeRF with spatial-temporal consistency.

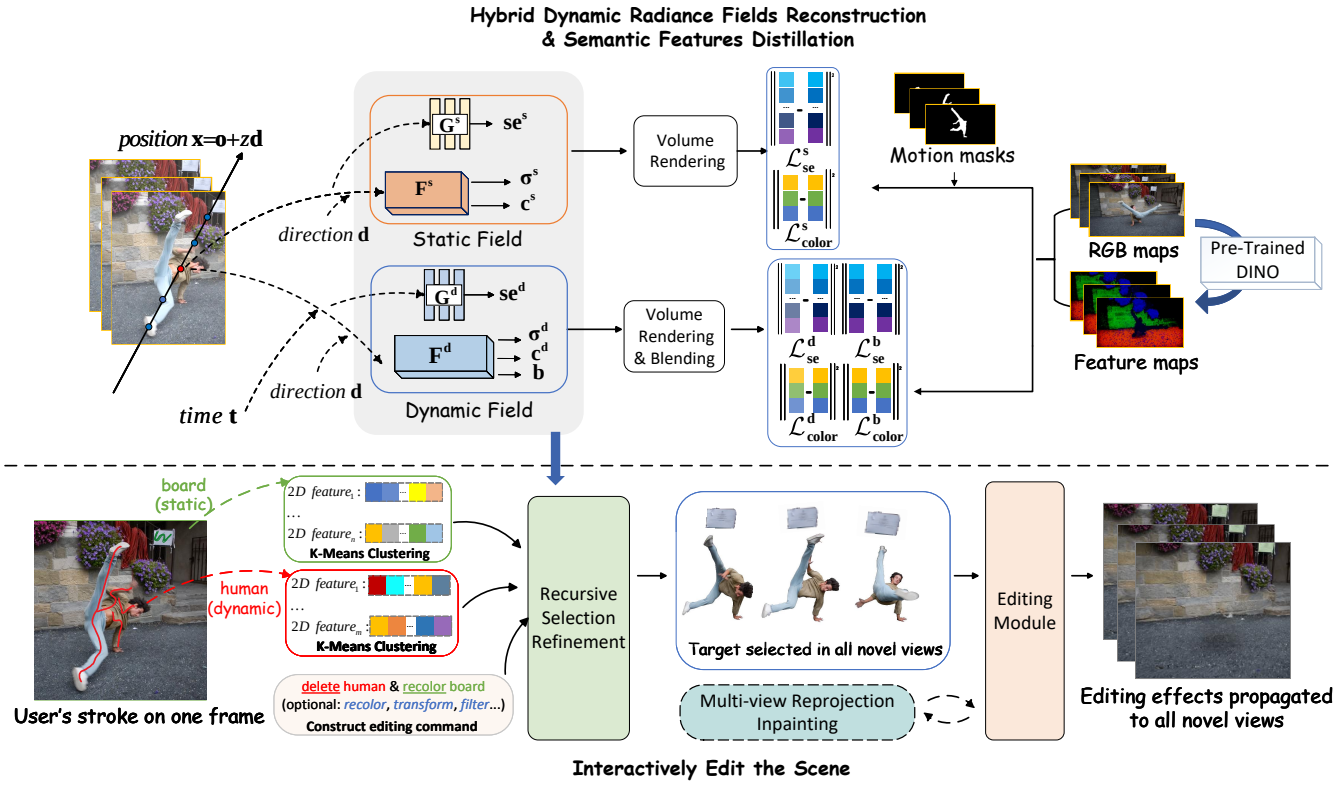


Figure 2. **4D-Editor framework overview.** The dynamic NeRF is represented by hybrid radiance fields F^s, F^d and corresponding feature fields G^s and G^d which are distilled from DINO teacher model. A user can choose a reference view, mark on desired edited objects and assign desirable operations. For example, **delete** dynamic human(red[∼] stroke) and **recolor** static board(green[∼] stroke). Then, two groups of 2D semantic features are collected and clustered by K-Means to recursively match the corresponding 4D features in dynamic NeRF to achieve precise object segmentation. Finally, desired editing operations on these objects are applied and the effect of editing is spread to the whole NeRF(the human disappeared and the board turned green).

NeRF with Semantics. Recent methods introduce semantic features into NeRF for spatial semantic segmentation [7, 10, 18, 43, 54]. Semantic-NeRF [54] encodes scene semantic information with appearance and geometry and achieve interactive segmentation using semantic label propagation. PNF [18] optimizes an object-aware neural scene representation that decomposes a scene into a set of objects and background using pseudo-supervision from predicted semantic segmentation. However, these methods require manual annotated labels. In contrast to above supervision way, our method distill semantic features from a pre-trained object segmentation model into Dynamic NeRF in a self-supervision manner.

3. Method

In this section, we first briefly introduce the background of hybrid representation of dynamic neural radiance fields. Subsequently, we propose our hybrid semantic features distillation method which serves as a guidance for editing one dynamic NeRF. We then detail the interactive editing pipeline including K-means clustering, 2D-4D feature matching, recursive selection refinement, and editing module. Finally, we demonstrate the Multi-view Reprojection Inpainting method for hole-filling in invisible areas after editing. The proposed framework is shown in Fig. 2.

3.1. Preliminary: Hybrid Representation of Dynamic Neural Radiance Field

Dynamic NeRFs (e.g, DynNeRF [12] and RoDynNeRF [25]) typically employ a hybrid representation composed of a static radiance field F^s and a dynamic radiance field F^d, to model dynamic scenes. In this hybrid representation, given a 3D position $\mathbf{x} \in \mathbb{R}^3$ with its normalized viewing direction $\mathbf{d} \in \mathbb{R}^3$ and time $t \in \mathbb{R}$, F^s maps radiance values as time-invariant density σ^s and RGB color $\mathbf{c}^s$ for background:

$$(\sigma^s, \mathbf{c}^s) = F^s(\mathbf{x}, \mathbf{d}) \tag{1}$$

While F^d maps radiance values as time-variant density σ^d and RGB color $\mathbf{c}^d$ for foreground. Furthermore, F^d also predicts blending weight b, forward scene flow s_f, and backward scene flow s_b, which are utilized to regularize the consistency of the scene flow to address the ambiguity in-

herent in monocular videos:

$$(\sigma^d, \mathbf{c}^d, s_f, s_b, b) = F^d(\mathbf{x}, \mathbf{d}, t) \tag{2}$$

The calculated density and color are then used in volume rendering alone the ray $\mathbf{r}$ emitted from the camera to obtain corresponding pixel color:

$$\begin{aligned}\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T(u_i) \cdot (\alpha(\sigma^s(u_i)\delta_i) \cdot \mathbf{c}^s(u_i) \cdot b \\ + \alpha(\sigma^d(u_i)\delta_i) \cdot \mathbf{c}^d(u_i) \cdot (1-b))\end{aligned} \tag{3}$$

$$T(u_i) = exp\left(\sum_{j=1}^{i-1} \sigma^s(u_j)\delta_j b + \sigma^d(u_j)\delta_j(1-b)\right) \tag{4}$$

where$\alpha(z) = 1 - exp(-z)$, $\delta_i = u_{i+1} - u_i$ is the distance between two neighbor sampled points along the ray, the N points $\{u_i\}_{i=1}^N$ are uniformly sampled between near plane and far plane [27], and $T(u_i)$ indicates the accumulated transmittance.

3.2. Hybrid Semantic Features Distillation

In order to maintain spatial-temporal consistency during object-level editing, we use two feature fields to store semantic features of static and dynamic parts of a scene respectively, which are denoted as G^s and G^d. We employ large pre-trained teacher models (e.g., DINO [5]) to distill semantic features into two feature fields, which serves as guidance for editing. Therefore, the static field is represented by F^s and G^s. Similarity, the dynamic field is represented by F^d and G^d, as shown in Fig. 2. Moreover, G^s stores time-invariant semantic features for multi-view consistency, whereas G^d stores time-variant features for both time-variant and multi-view consistency.

We obtain the static semantic feature $\mathbf{se}^s \in \mathbb{R}^C$ and dynamic semantic feature $\mathbf{se}^d \in \mathbb{R}^C$ of a sampled point according to follows:

$$\mathbf{se}^s = G^s(\mathbf{x}), \quad \mathbf{se}^d = G^d(\mathbf{x}, t) \tag{5}$$

Note that we disregard view direction $\mathbf{d}$ due to the direction-agnostic nature of scene semantics. Specifically, given a set of N consecutive frames $\mathcal{I} : \{I_i\}_{i=1}^N \in \mathbb{R}^{H\times W\times 3}$, we utilize the DINO ViT-b8 model to generate corresponding semantic feature maps $\in \mathbb{R}^{H/8\times W/8\times C}$. Then, we upsample these low-resolution feature maps through an upsampling layer to output the final feature maps $\{Se_i\}_{i=1}^N \in \mathbb{R}^{H\times W\times C}$. Next, we employ volume rendering to obtain the pixel-aligned semantic features $\hat{Se}^s(\mathbf{r})$ and $\hat{Se}^d(\mathbf{r})$, which represent the accumulated static and dynamic semantic feature along a ray $\mathbf{r}$:

$$\begin{aligned}\hat{Se}^s(\mathbf{r}) &= \sum_{i=1}^{N} T^s(u_i)\alpha(\sigma^s(u_i)\delta_i)\mathbf{se}^s(u_i), \\ \hat{Se}^d(\mathbf{r}) &= \sum_{i=1}^{N} T^d(u_i)\alpha(\sigma^d(u_i)\delta_i)\mathbf{se}^d(u_i)\end{aligned} \tag{6}$$

where $T^s(u_i) = exp(\sum_{j=1}^{i-1}\sigma^s(u_j)\delta_j)$ and $T^d(u_i) = exp(\sum_{j=1}^{i-1}\sigma^d(u_j)\delta_j)$.

Finally, we calculate the final semantic feature by blending static and dynamic semantic features, similar to Equation 3:

$$\begin{aligned}\hat{Se}^b(\mathbf{r}) = \sum_{i=1}^{K} T^b(u_i)(\alpha(\sigma^s(u_i)\delta_i)\mathbf{se}^s(u_i)b \\ + \alpha(\sigma^d(u_i)\delta_i)\mathbf{se}^d(u_i)(1-b))\end{aligned} \tag{7}$$

We add three new losses to train the static semantic field and dynamic semantic field: $\mathcal{L}_{se}^s$ for static part of a scene, $\mathcal{L}_{se}^d$ and $\mathcal{L}_{se}^b$ for all pixels. We treat the output of the teacher model as the ground truth. By minimizing the difference between predicted features and the ground truth, two semantic fields can learn scene semantics.

$$\begin{aligned}\mathcal{L}_{se}^s &= \sum_{\mathbf{r}\in\mathcal{R}^s} \|Se(\mathbf{r}) - \hat{Se}^s(\mathbf{r})\|_2^2, \\ \mathcal{L}_{se}^d &= \sum_{\mathbf{r}\in\mathcal{R}^s+\mathcal{R}^d} \|Se(\mathbf{r}) - \hat{Se}^d(\mathbf{r})\|_2^2, \\ \mathcal{L}_{se}^b &= \sum_{\mathbf{r}\in\mathcal{R}^s+\mathcal{R}^d} \|Se(\mathbf{r}) - \hat{Se}^b(\mathbf{r})\|_2^2\end{aligned} \tag{8}$$

where $\mathcal{R}^s$, $\mathcal{R}^d$ are sampled rays from static and dynamic part of a scene, respectively. The total semantic loss is

$$\mathcal{L}_{se} = \mathcal{L}_{se}^s + \lambda_1\mathcal{L}_{se}^d + \lambda_2\mathcal{L}_{se}^b \tag{9}$$

3.3. Recursive Selection Refinement

K-Means Query for Multi-Objects Selection. In our 4D-Editor framework, users can mark desired editing objects with a brush on a reference frame. Based on user's strokes, 4D-Editor extracts target 2D semantic features from corresponding feature maps generated by DINO. These semantic features are utilized to construct different queries for matching multiple objects in feature fields, i.e., G^s and G^d. Then, we can use queries to match 2D-4D features to segment editing objects in dynamic NeRF.

However, user provided strokes are sparse, which leads to naturally inadequate and inexpressive semantic features, where simply query such as average feature can cause incorrect 2D-4D feature matches. Therefore, inspired by ISRF

$u_i \in \mathcal{V}$	$d(\gamma, u_i) \leq \alpha$
$u_i \in \mathcal{P}$	$\alpha < d(\gamma, u_i) \leq \alpha + \beta$
$u_i \in \mathcal{Q}$	$d(\gamma, u_i) > \alpha + \beta$

Table 2. Ranges of feature distance among $\mathcal{V}, \mathcal{P}, \mathcal{Q}$.

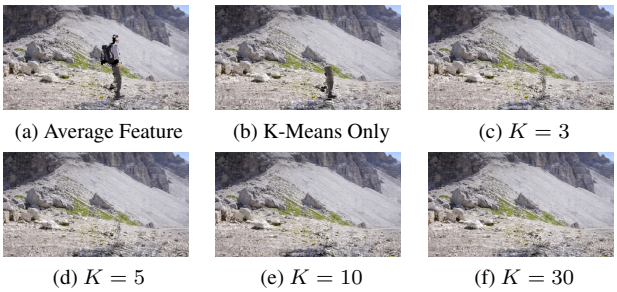
[13], we utilize K-Means to cluster a set of most significant and related features for individual object to improve the accuracy of feature matching. Actually, 4D-Editor generates multiple sets for multi-objects.

Recursive Refinement. Despite employing K-Means for effective 2D-4D feature matching, achieving accurate object segmentation still remains challenging: Owing to 8x up-sampled feature maps, unsupervised semantic feature distillation inherently leads to imprecise semantic segmentation in dynamic NeRF, particularly for the confusion between object edges and background.

Given a M K-means clustered query $\gamma \in \mathbb{R}^{M \times C}$, a set of sampled points $\mathcal{U}$, and corresponding indexes $\mathcal{A}$, we can obtain the feature distance $d(\gamma, u_i)$ between one sampled point $u_i \in \mathcal{U}$ and query γ according to $d(\gamma, u_i) = \min_M(\|\gamma - se(u_i)\|_2)$. However, it is difficult to directly set the threshold α of feature distance to select potential points where $d(\gamma, u_i) < \alpha$, due to the semantic ambiguity between object edges and background. Here, a slightly higher threshold may result in selecting unexpected areas, while a lower threshold cannot ensure that the entire object is selected, leading to obvious "artifacts" in rendered views. Another native solution is to use neighbour points(e.g., within a unit sphere) to indirectly judge the validity of the current point. However, the radius is still a threshold that is infeasible to set due to the ambiguousness of object edges.

To sovle this problem, we propose recursive selection refinement(RSR) algorithm that is inspired by the ray-tracing process [38], which estimate unbiased indirect illumination through recursively computing the intersection of rays and surfaces. RSR algorithm recursively refine the object selection instead of relying on a fixed threshold. Thus, we not only avoid heavy manual threshold setting but also achieve precise object selection, especially object edges. Specifically, we add exploration range β and divide all sampled points into three sets: (1) Valid point set $\mathcal{V}$, which contains the main part of the object. (2) Possible point set $\mathcal{P}$, which contains points are likely on the object surface. (3) Impossible point set $\mathcal{Q}$, which contains points far away from the object. Table 2 shows the differences.

Then, we need to distinguish these sampling points belonging to object in possible point set $\mathcal{P}$. We record their original indexes and apply random offsets to points in $\mathcal{P}$ to nudge them into valid point set and impossible point set, thereby forming new valid point set $\mathcal{V}'$, possible point set $\mathcal{P}'$, and impossible point set $\mathcal{Q}'$ as shown in Algorithm 1. We repeat this recursive refinement process until reaching the maximum recursion number K or the possible point set is empty. Similar to ray-tracing, we also can obtain the approximate unbiased estimation on the object segmentation (or object selection) as demonstrated in Fig. 3 and Table 3.



(a) Average Feature (b) K-Means Only (c) $K = 3$

(d) $K = 5$ (e) $K = 10$ (f) $K = 30$

Figure 3. **Refinement with different recursive number.** Compared to **(a)** and **(b)**, our method **(c)-(f)** achieve great improvement in object removal. With larger recursion number, artifacts are eliminated, while maintaining spatial-temporal consistency in other areas. Similar to the convergence of ray-tracing, our method also achieve high quality when $K = 10$. The benefits generated by a larger K are very small.

Algorithm 1: Recursive Selection Refinement

Input: $\mathcal{U}, \mathcal{A}, K, \alpha, \beta, \gamma, s, k = 0$
Output: valid point index set $\mathcal{W}$
Def RSR $(\mathcal{U}, \mathcal{A}, k, \alpha, \beta, \gamma, s)$ **:**
  $\mathcal{W} \leftarrow [\ ]$
  if $k = K$ or $\mathcal{U} = \varnothing$: **return** $\mathcal{W}$
  // Calculate feature distances
  $\mathcal{D} \leftarrow \{d(\gamma, u_i), u_i \in \mathcal{U}\}$
  // Store valid points & indexes
  $\mathcal{V} \leftarrow \{d(\gamma, u_i) \leq \alpha, d(\gamma, u_i) \in \mathcal{D}\}$
  $\mathcal{W}$.append(index of $\mathcal{V}$ from $\mathcal{A}$)
  // Dismiss impossible points
  $\mathcal{Q} \leftarrow \{d(\gamma, u_i) > \alpha + \beta, d(\gamma, u_i) \in \mathcal{D}\}$
  // Get possible points and indexes
  $\mathcal{P} \leftarrow \{\alpha < d(\gamma, u_i) \leq \alpha + \beta, d(\gamma, u_i) \in \mathcal{D}\}$
  $\mathcal{A}' \leftarrow$ index of $\mathcal{P}$ from $\mathcal{A}$
  // Apply random offsets
  $\mathcal{P}' \leftarrow$ offset$(\mathcal{P}, \text{randn}(0, s))$
  $\mathcal{V}' \leftarrow$ RSR $(\mathcal{P}', \mathcal{A}', k + 1, \alpha, \beta, \gamma, s)$
  $\mathcal{W}$.append(index of $\mathcal{V}'$ from $\mathcal{A}$)
  return $\mathcal{W}$

3.4. Editing Module

After the object segmentation, for each editing object, we can obtain two sampling point sets: $\mathcal{T}$ inside the object and $\mathcal{S}$ outside. Then, we can edit specific object as follows:

Remove. Removing one object in a dynamic NeRF, actually means that we need to treat the object as transparent in order to expose the background behind the object. For sampling point $t_i \in \mathcal{T}$ of a static object, we set the density $\sigma^s(t_i) = 0$, so that the point will be ignored during volume rendering. As for the dynamic object, we not only set density $\sigma^d(t_i) = 0$ but also set blending weight $b(t_i) = 1$. This

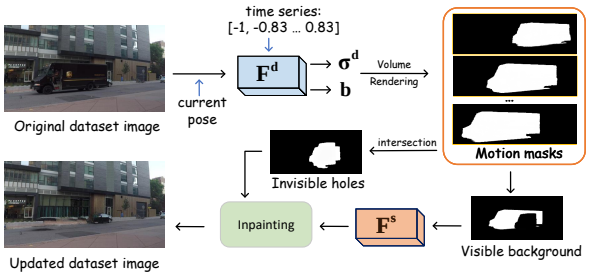


Figure 4. **Example of fill holes in a image via multi-view reprojection inpainting.**

is because we expect the dynamic object removal operation not to affect the static field.

Filter. To filter an object from a dynamic NeRF, we set $\sigma(s_i) = 0$ where $s_i \in \mathcal{S}$, which means making all points outside the object invisible.

Composite. We can composite objects filtered from other scenes(represented by $\mathcal{Z}$) into current scene by setting $\sigma(s_i) = \sigma(z_i), c(s_i) = c(z_i)$ where $z_i \in \mathcal{Z}$.

Recolor. For appearance modification, we find editing color $\mathbf{c}(s_i)$ in 4D space is the similar to that on 2D images: we exchange RGB channels to change hue parameter, improve the corresponding RGB channel to change RGB saturation parameter, and add all RGB channels to change lightness parameter. This discovery can make it more convenient and controllable for us to recolor. The recoloring results are demonstrated in Fig. 7.

Transform. We allow users apply various transforming operations to the object, such as translating, scaling, mirroring or duplicating. Users only need to define a transforming function $mapping(x) : x \rightarrow x'$ to set the spatial mapping relation of the target object. For translating, we . Then, we set $\sigma(mapping(s_i)) = \sigma(s_i)$, $\mathbf{c}(mapping(s_i)) = \mathbf{c}(s_i)$ for volume rendering. Note that users can also remove the original object first, just set $\sigma(s_i)$ to zero and use our proposed inpainting method (introduced in Section 3.5) to fill *holes*, then apply transformed density $\sigma(mapping(s_i))$ and color $\mathbf{c}(mapping(s_i))$ for transforming(e.g., $mirror(x, y, z) : (-x, -y, z)$, $scale(x, y, z) : (x, y, z) \times 0.5$). The detailed experimental results can be seen in Fig. 8.

3.5. Multi-view Reprojection Inpainting

Since the limited observations of the scene, the removing operation may cause "holes" in the novel views, leading to obvious artifacts. A feasible solution is to use an inpainting model [40] to fill these holes in edited novel rendered images, and then use these inpainted images as training sets to retrain the dynamic NeRF model, similar to the approach employed by SPIn-NeRF [29]. However, the inpainting model treats each image as an independent individual, ignoring the multi-view consistency between images, resulting in artifacts and 3D inconsistency. Therefore, we propose multi-view reprojection inpainting method. As Fig. 4 shows, we divide the training set $\mathcal{J}$ into two parts: $\mathcal{J}_{vis}$ and $\mathcal{J}_{inv}$. The holes in images in the former can be seen in other views. While the holes in images in the latter are invisible across all views. These two parts need to be filled separately. This belief stems from the notion that, given a specific perspective, if a occluded 3D point is present in some other views, its geometry and appearance information are inherently captured during the NeRF's reconstruction process, thereby allowing inherent inpainting by NeRF. Conversely, if the occluded points lack observation in other views, then it is necessary to use other models for inpainting.

Thus, the key objective is to identify which holes are visible and which are not. To accomplish this, we calculate time-variant motion masks for each individual image in the original dataset. These masks are generated by performing volume rendering on the blending weight b across the entire time series on corresponding camera poses. The overlapping area of these masks represents $\mathcal{J}_{inv}$, while the remaining regions constitute $\mathcal{J}_{vis}$. This process is applied to all original images in the dataset. For inpainting $\mathcal{J}_{vis}$, we eschew the traditional pixel reprojection method due to its drawbacks such as potential loss of fine details and sensitivity to geometric inconsistencies or occlusions between views. Instead, we leverage NeRF's inherent multi-view information to accomplish the inpainting of $\mathcal{J}_{vis}$ in all training images. In this way, these holes are filled with 3D consistency. Subsequently, lama model [40] is utilized for inpainting on the remaining invisible parts, $\mathcal{J}_{inv}$. By prefilling the background, we narrow down areas that need to be generated, which strengthens the reliability of inpainting results. Static field F^s is then retrained based on these inpainted images. As Fig. 9 shows, our proposed inpainting method demonstrates better inpainting results in comparison with only use lama model to fill holes.

4. Experiments

4.1. Experimental setup

Datasets. We experiment on three datasets: Dynamic View Synthesis [51], DAVIS [34] and NeuPhysics [35].

Implements Details. We implement 4D-Editor with Pytorch, using Adam optimizer to update learnable parameters on one NVIDIA A6000 GPU. We train original NeRF and semantic parts separately, where training DynNeRF or RobustNeRF can take 6-8 hours, while time for training additional distillation of hybrid semantic features is only 10-15 minutes, based on a complete spatial knowledge. Editing one frame can take 2 seconds based on RobustNeRF and 10 seconds based on DynNeRF.

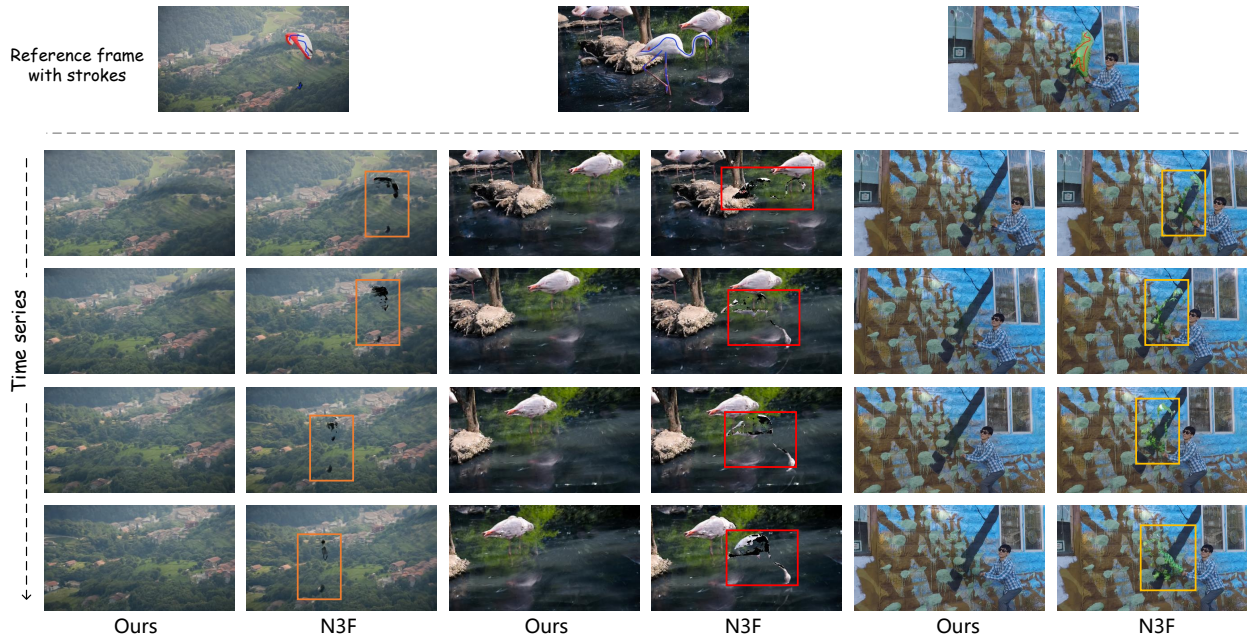


Figure 5. **A Qualitative Comparison of 4D-Editor's Segmentation in 4D space against N3F.** Using a constructed feature query based on marked regions in a reference frame, we search for similar regions in the semantic feature fields across the entire time series. Subsequently, the deletion operation affects all novel views accordingly. Our results demonstrate that 4D-Editor achieves superior segmentation accuracy compared to N3F, as evidenced by cleaner deletion without artifacts.

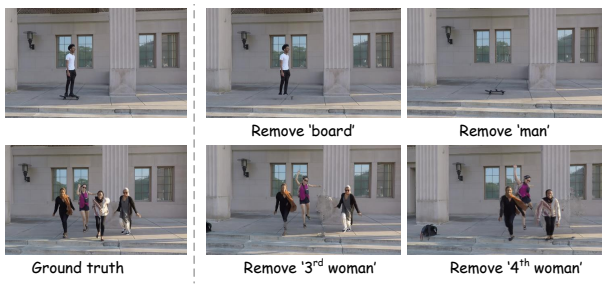
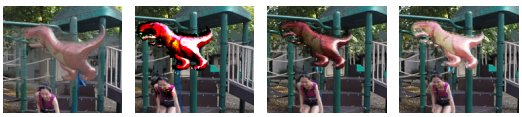
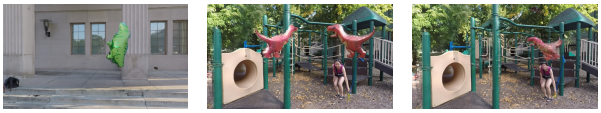


Figure 6. **Object-Level Editing.** We individually remove different objects within the same scene, and only the user-defined target area is affected, with rest regions remaining unaffected.



(a) saturation ↓ (b) saturation ↑ (c) lightness ↓ (d) lightness ↑

Figure 7. **Recolor the ballon.**



(a) composition (b) mirror (c) scale

Figure 8. **Composition & Transformation.** DOES supports composition across different scenes and flexible transformations.

4.2. Interactive Object Removal using strokes

Our method enables users to perform interactive object editing using strokes. As Fig. 5 shows, the user simply annotates the desired editing object on a reference frame of the original videos(Row 1). After set a editing operation (e.g., removal), this operation can be propagated to the whole scenes. Compared with N3F [42], our method can achieve clean and continuous removal effects, whereas N3F leaves obvious artifacts.

Our proposed 4D-Editor can remove arbitrary objects in dynamic NeRF by using different queries to match with semantic feature fields, while maintaining spatial-temporal consistency in rest regions as shown in Fig. 6. Moreover, 4D-Editor also enables users to edit multiple objects, such as deleting the women, recoloring, and shifting the balloon simultaneously, as demonstrated in Fig. 1, which is achieved by constructing multiple queries and editing commands at one time.

4.3. Refined and Diverse Editing

4D-Editor facilitates accurate segmentation to achieve controllable modifications in appearance color, akin to the HSL (Hue, Saturation, and Lightness) editing in PhotoShop [1]. As illustrated in Fig. 7, we begin by selecting the balloon, which is initially red in color, and modify its hue, saturation, and lightness individually. This is accomplished by increasing or decreasing values in all color channels.

Additionally, 4D-Editor provides users with the flexibility to define their own transformation functions to apply affine transformations on individual selected objects.

K	$new\ \mathcal{V}$	$\mathcal{P}$	$all\ \mathcal{V}$	K	$new\ \mathcal{V}$	$\mathcal{P}$	$all\ \mathcal{V}$
1	2.84 M	828.13 k	2.84 M	10	3.68 k	35.58 k	3.22 M
2	225.93 k	373.77 k	3.06 M	20	426	6.55 k	3.23 M
3	63.97 k	239.31 k	3.13 M	30	114	1.64 k	3.24 M
5	20.69 k	120.73 k	3.18 M	50	8	169	3.24 M
8	6.73 k	54.89 k	3.21 M	60	1	56	3.24 M

Table 3. **Points in each recursion iteration.** When the recursive number reaches 20, the number of new valid points decreases significantly, which converges to 3.24M despite further recursion.

K	$mIoU$	$mAcc$	K	$mIoU$	$mAcc$
1	72.79 %	79.81 %	30	78.63 %	93.45 %
5	76.53 %	85.39 %	40	80.11 %	93.24 %
20	80.85 %	92.70 %	50	79.21 %	93.88 %

Table 4. **Effects of maximum recursion number K.**

Fig. 8 illustrates the effects of object-level composition across multiple scenes and a variety of transformations. The green balloon is actually filtered from another scene and inserted into the playground in Fig. 8a. As for different transformation operations, we can still keep the correct spatial information: The mirrored balloon is partially traversed in Fig. 8b while previously obscured background become visible after scaling in Fig. 8c.

4.4. Ablation Studies

Recursive Selection Refinement. For accurate object segmentation in dynamic NeRF, it is essential to utilize a 2D-4D feature matching technique that matches marked features with the distilled semantic features in volumetric space. As depicted in Fig. 3, we evaluate three different methods: (1) Average feature, (2) K-Means clustered features [13], and (3) Recursive selection refinement (our method). Fig. 3a illustrates ineffective feature matching resulting from the limited capability of average features to extract meaningful features. K-Means clustered features only remove part of object as shown in Fig. 3b, leading to artifacts(e.g., remaining legs). However, our proposed recursive selection refinement improves the precise of feature matching and achieves near perfect removal, as demonstrated in Fig. 3c-3f.

Table 3 presents the number of newly added valid points and the total number of possible points in each recursion. We can see that when the recursive number reaches 20, almost all points belonging to the target is selected and ultimately converged to $3.24M$. We also studied the effect of different recursive number on mIoU and mAcc in Table 4. We can find that as the number of recursive number K increases, the object segment accuracy also improves, ultimately converging to around 93%. (See more analysis in supplementary material).

Multi-view Reprojection Inpainting. By employing multi-view reprojection, we restore backgrounds using information captured from multiple perspectives (refer to Fig. 9). In contrast to the direct inpainting method, which results in the loss of significant spatial information (e.g., the disappearance of the black car in Fig. 9b) and produces independent multi-view outcomes causing blurry artifacts after re-training F^s, our method preserves the original spatial information in the scene. Furthermore, after pre-filling in the 'visible' parts of the backgrounds by reprojection, we narrow down the inpainting areas to alleviate the failure caused by inpainting models, obtaining more reliable and detailed outcomes (e.g., walls with clear textures in Fig. 9c). Fig. 10) demonstrates that our proposed multi-view reprojection inpainting method can achieve better multi-view consistency compared with only using inpainting models to fill holes. Moreover, the inpainting performance can be further improved by employing more powerful inpainting models.

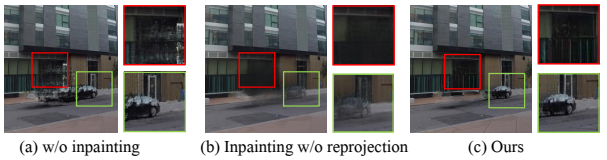


Figure 9. **Comparison of different inpainting methods.** To fill holes in **(a)**, we pre-fill backgrounds through reprojection and narrow the inpainting region for reliable results(e.g., the black car remains in **(c)** while disappears in **(b)**).

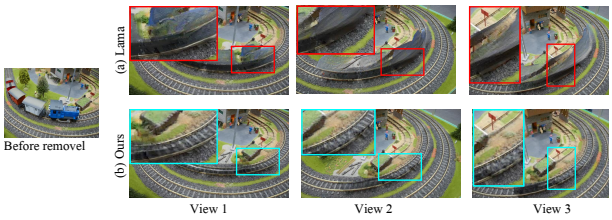


Figure 10. **Visulaizations of multi-view consistency.** The top raw demonstrates that direct inpainting causes view-inconsistent with obvious artifacts, while our method (the bottom raw) exhibits better multi-view consistency.

5. Conclusion

We propose a novel interactive editing framework for dynamic NeRFs that enables object-level editing operations through user-provided strokes on a single reference frame, and delivers spatial-temporal consistency across the entire time series. We present several excellent results from multiple challenging scenes. However, our method has limitations in removing shadows of moving objects. Additionally, while we can handle invisible background completion, in some times, the scene inpainting may still have spatial-temporal inconsistencies. We will investigate ways to address the problem in future works.

References

[1] Adobe. Adobe Photoshop. https://www.adobe.com/products/photoshop.html. Accessed on August 30, 2023. 7

[2] Chong Bao, Yinda Zhang, Bangbang Yang, Tianxing Fan, Zesong Yang, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Sine: Semantic-driven image-based nerf editing with prior-guided editing field. In *The IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2023. 2

[3] Blender Foundation. Blender. https://www.blender.org/, 2023. Version 3.6. 2

[4] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 2

[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 the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 1, 2, 4

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields, 2022. 1

[7] Yuedong Chen, Qianyi Wu, Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Sem2nerf: Converting single-view semantic masks to neural radiance fields. In *European Conference on Computer Vision*, pages 730–748. Springer, 2022. 3

[8] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *CVPR*, 2023. 2

[9] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 1

[10] Xiao Fu, Shangzhan Zhang, Tianrun Chen, Yichong Lu, Lanyun Zhu, Xiaowei Zhou, Andreas Geiger, and Yiyi Liao. Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation. In *2022 International Conference on 3D Vision (3DV)*, pages 1–11. IEEE, 2022. 3

[11] Wanshui Gan, Hongbin Xu, Yi Huang, Shifeng Chen, and Naoto Yokoya. V4d: Voxel for 4d novel view synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 2

[12] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021. 1, 3

[13] Rahul Goel, Dhawal Sirikonda, Saurabh Saini, and PJ Narayanan. Interactive segmentation of radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4201–4211, 2023. 1, 2, 5, 8

[14] Ori Gordon, Omri Avrahami, and Dani Lischinski. Blended-nerf: Zero-shot object generation and blending in existing neural radiance fields. *arXiv preprint arXiv:2306.12760*, 2023. 2

[15] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 2

[16] Hankyu Jang and Daeyoung Kim. D-tensorf: Tensorial radiance fields for dynamic scenes, 2022. 1, 2

[17] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *Advances in Neural Information Processing Systems*, 35:23311–23330, 2022. 1

[18] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12871–12881, 2022. 3

[19] Sacha Lewin, Maxime Vandegar, Thomas Hoyoux, Olivier Barnich, and Gilles Louppe. Dynamic nerfs for soccer scenes. *arXiv preprint arXiv:2309.06802*, 2023. 2

[20] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *International Conference on Learning Representations*, 2022. 1

[21] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video, 2022. 2

[22] Hao-Kang Liu, I-Chao Shen, and Bing-Yu Chen. Nerf-in: Free-form nerf inpainting with rgb-d priors, 2022. 2

[23] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Devrf: Fast deformable voxel radiance fields for dynamic scenes. *Advances in Neural Information Processing Systems*, 35:36762–36775, 2022. 2

[24] Ruiyang Liu, Jinxu Xiang, Bowen Zhao, Ran Zhang, Jingyi Yu, and Changxi Zheng. Neural impostor: Editing neural radiance fields with explicit shape manipulation, 2023. 2

[25] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13–23, 2023. 1, 2, 3

[26] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12663–12673, 2023. 2

[27] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 4

[28] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Marcus A. Brubaker, Jonathan Kelly, Alex Levinshtein, Konstantinos G.

Derpanis, and Igor Gilitschenski. Reference-guided controllable inpainting of neural radiance fields, 2023. 2

[29] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G Derpanis, Jonathan Kelly, Marcus A Brubaker, Igor Gilitschenski, and Alex Levinshtein. Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20669–20679, 2023. 2, 6

[30] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4328–4338, 2023. 2

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, jul 2022. 1

[32] Jangho Park, Gihyun Kwon, and Jong Chul Ye. Ed-nerf: Efficient text-guided editing of 3d scene using latent space nerf, 2023. 2

[33] 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. *ACM Trans. Graph.*, 40(6), dec 2021. 1

[34] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6

[35] Yi-Ling Qiao, Alexander Gao, and Ming Lin. Neuphysics: Editable neural geometry and physics from monocular videos. *Advances in Neural Information Processing Systems*, 35:12841–12854, 2022. 1

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2

[37] Hoigi Seo, Hayeon Kim, Gwanghyun Kim, and Se Young Chun. Ditto-nerf: Diffusion-based iterative text to omni-directional 3d model, 2023. 2

[38] Peter Shirley. *Realistic Ray Tracing*. A K Peters/CRC Press, 2nd edition, 2003. 5

[39] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023. 1

[40] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2149–2159, 2022. 6

[41] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021. 1, 2

[42] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural feature fusion fields: 3d distillation of self-supervised 2d image representations. In *2022 International Conference on 3D Vision (3DV)*, pages 443–453. IEEE, 2022. 1, 2, 7

[43] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi SM Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *arXiv preprint arXiv:2111.13260*, 2021. 3

[44] Dongqing Wang, Tong Zhang, Alaa Abboud, and Sabine Süsstrunk. Inpaintnerf360: Text-guided 3d inpainting on unbounded neural radiance fields, 2023. 2

[45] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, Yafei Song, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19706–19716, 2023. 2

[46] Liao Wang, Qiang Hu, Qihan He, Ziyu Wang, Jingyi Yu, Tinne Tuytelaars, Lan Xu, and Minye Wu. Neural residual radiance fields for streamably free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 76–87, 2023. 2

[47] 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 *European Conference on Computer Vision*, pages 597–614. Springer, 2022. 2

[48] Zheyuan Yang, Yibo Liu, Guile Wu, Tongtong Cao, Yuan Ren, Yang Liu, and Bingbing Liu. Learning effective nerfs and sdfs representations with 3d generative adversarial networks for 3d object generation: Technical report for iccv 2023 omniobject3d challenge, 2023. 2

[49] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 2

[50] Youtan Yin, Zhoujie Fu, Fan Yang, and Guosheng Lin. Or-nerf: Object removing from 3d scenes guided by multiview segmentation with neural radiance fields, 2023. 2

[51] Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5336–5345, 2020. 2, 6

[52] 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 the IEEE/CVF Con-*

ference on Computer Vision and Pattern Recognition, pages 18353–18364, 2022. 2

[53] Jingbo Zhang, Xiaoyu Li, Ziyu Wan, Can Wang, and Jing Liao. Text2nerf: Text-driven 3d scene generation with neural radiance fields, 2023. 2

[54] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15838–15847, 2021. 1, 3