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

Supplementary Material

1. Implementation Details

We use Principal Component Analysis (PCA) method to extract 64 most important semantic features from DINO [1] ViT-b8 model. We use 64 sampled points on each ray in volume rendering.

During the editing process, we set different threshold α and number of K-Means clusters N_k according to different sizes of target objects: $\alpha = 0.4 \pm 0.1$, $N_k = 3 \pm 2$ for small objects and $\alpha = 0.7 \pm 0.1$, $N_k = 20 \pm 15$ for small ones. The value of β should be approximately 1/3 of α. Actually, in our experiments, all these values can be set loosely within Recursive Selection Refinement.

2. Interactive GUI

We design a user-friendly graphical user interface (GUI) that facilitates interactive editing of dynamic scenes. The editing procedure involves three steps: 1) Selecting a reference frame, 2) Marking target objects with strokes in distinct colors, and 3) Configuring editing parameters, including editing operations, threshold α, exploration range β and recursion depth K. These steps are depicted in Fig. **??**.

3. Model Structure

3.1. Structures of Hybrid Radiance Fields

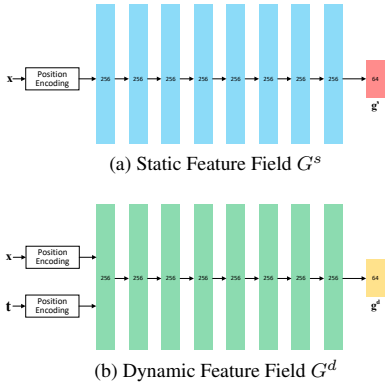
We directly adopt RobustNeRF [2] and DynamicNeRF [4] to represent a dynmaic scene, their methods are displayed in Fig. 9.

3.2. Structures of Hybrid Semantic Fields

We design semantic fields G^s and G^d, to represent semantic information of the static and dynamic components in the scene, respectively. Both semantic fields are modeled using an 8-layer multi-layer perceptron (MLP). As illustrated in Fig. 1, G^s takes the position $\mathbf{x}$ as input after position encoding, while G^d incorporates both the position $\mathbf{x}$ and the time t.

3.3. Volume Rendering on Semantic Features

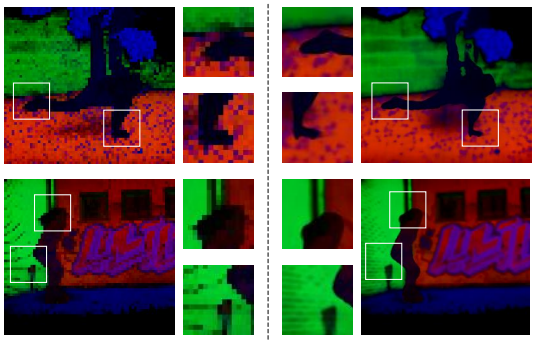
The spatial semantic features themselves are represented by a 64-dimensional vector. We apply volume rendering to these features and select the first 3 dimensions to generate RGB visualizations. As shown in Fig. 2, upon reconstructing the spatial semantic information using semantic fields, our approach demonstrates superior preservation of both multi-view and spatio-temporal consistency of semantic information in the 4D space, compared to the relatively coarse-grained feature map generated by the DINO model [1]. Notably, on object edges, the transition of semantic information appears remarkably smooth.



(a) Static Feature Field G^s

(b) Dynamic Feature Field G^d

Figure 1. **Structures of Semantic Fields.**



(a) DINO features (b) Ours

Figure 2. **Visualization of Semantic Features.** After semantic features distillation, we obtain enhanced semantic features in 4D space (For additional comparisons, refer to the demo videos in the supplementary material).

4. Visualization of Recursive Selection Refinement

Figure 3 shows the flow of semantic feature matching and target selection(semantic segmentation). Given sampled points $p_1 \dots p_8$, we categorize them based on feature distances: valid points p_4 p_5, impossible points p_1 p_2 p_8, possible points p_3 p_6 p_7. Then we introduce a random offset on

K	$new\ \mathcal{V}$	$\mathcal{P}$	$all\ \mathcal{V}$	IoU	Acc
1	2.84M	828.13k	2.84M	72.79	79.81
5	20.69k	120.73k	3.18M	76.53	85.39
20	426	6.55k	3.23M	80.85	92.70
30	114	1.64k	3.24M	78.63	93.45
40	8	169	3.24M	80.11	93.24
50	1	56	3.24M	79.21	93.88

Table 1. **Effects of maximum recursion number K.**

possible points and recalculate feature distances, repeating until these points are judged as valid or impossible. Here, p_3 p_6 are considered as valid points after 1 or 2 iterations respectively, while p_7 excluded after 2 iterations. Ultimately, the selected points are p_3, p_4, p_5, p_6. We will repeat this process for all sampling points on each ray during the rendering process, as shown in Algorithm 4. As for the same batch of sampling points, we can apply different random offset step s at one time (*e.g.*, $s = (1/50, 1/100, 1/150)$) to control different ranges of selection on target.

In RSR, we initially get N possible points near the edges of the object and M valid points. Given the feature points around the object's edges, we determine the probabilities of a point with an offset, being classified as ***valid*** with a possibility of μ or ***possible*** with a possibility of λ. Note that $\mu + \lambda < 1$ due to the presence of invalid points, where λ also indicates the probability of progressing to the next round. For each point, the expected probability of being valid is $\lambda + \lambda\mu + \lambda^2\mu + ...$ This is a geometric progression, the sum of which is $\lambda/(1-\mu)$. Finally, the expected number of points that will be selected is $M + N\lambda/(1-\mu)$.

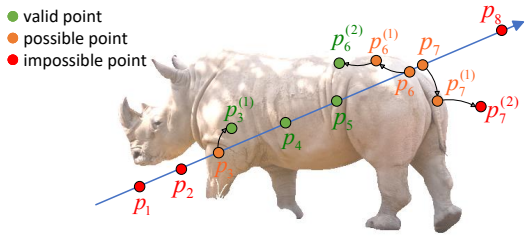


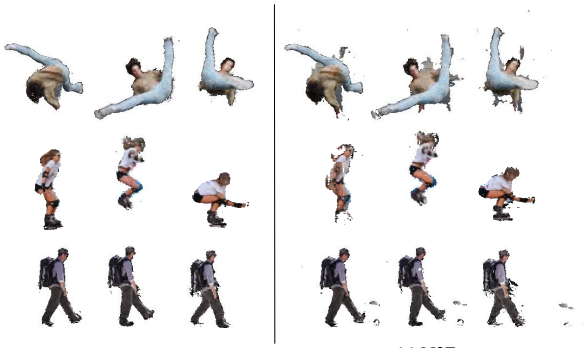
Figure 3. **Process of Recursive Selection Refinement.** Although DINO model generates coarse semantic feature maps which are used as the training set, our reconstruction and volume rendering of the 4D spatial semantic information results in finer semantic feature maps.

5. Analysis on Object Segmentation

In this section, we begin by conducting a qualitative and quantitative comparison between N3F [3] and our method in terms of object segmentation performance. Subsequently,

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($\mathcal{P}$, 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 $\mathcal{W}$

Figure 4. Recursive Selection Refinement



(a) Ours (a) N3F

Figure 5. **Results of Object Segmentation.** Compared with N3F, we achieve finer object segmentation, which contains fewer artifacts especially for edge areas.

we evaluate the impact of the hyperparameter β (exploration range) on Recursive Selection Refinement.

5.1. Evaluation on Segmentation Accuracy

We fine-tune the threshold of N3F for object segmentation tasks and dispaly its best results in the right column of Fig. 5. Since our method utilizes Recursive Selection Refinement to achieve more precise selection of target objects, surpassing N3F in both qualitative and quantitative assessments (indicated by higher Acc and IoU values in Table. 2). In our experiments, we observe that our method performs especially well in scenes containing significant object movement (*e.g.*, a girl starting from the left side and jumping to the right side in the *Rollerblade* dataset). Conversely, N3F produces numerous broken artifacts in such scenes. Fig. 6 presents additional results of object segmentation in 4D space.

5.2. Analysis on exploration range β

The selection of the exploration range parameter β can be loose and flexible. Fig. 3 demonstrates that the segmentation quality remains consistent despite using different val-

Table 2. **Quantitative Analysis on Object Segmentation.** We choose several scenes from DAVIS dataset and use its true motion masks as ground truth.

Scene	Metric	N3F	Ours
Breakdance Flare	**Acc** ↑	89.96	**94.12**
	IoU ↑	**84.97**	83.09
Rollerblade	**Acc** ↑	83.38	**93.10**
	IoU ↑	71.55	**81.56**
Hike	**Acc** ↑	94.80	**95.16**
	IoU ↑	85.73	**89.88**

(a) balloon (b) woman (c) balloon

Figure 6. **Object Segmentation in 4D Space.**

Table 3. **Effects of exploration range β.** We set the parameters as follows: $K = 3$, $N_k = 8$, and $\alpha = 0.6$. Additionally, we experiment with different values of β to demonstrate that it has no impact on the segmentation quality. Consequently, β can be loosely set during editing operations.

β	0.05	0.08	0.1	0.12	0.15	0.18	0.2
Acc ↑	93.10	93.42	93.81	93.66	93.94	94.02	93.94
IoU ↑	81.56	81.26	80.90	80.77	80.21	79.65	79.26

ues of β. This finding supports the notion that it is the recursion depth parameter K that significantly enhances segmentation accuracy, while β merely serves as an auxiliary factor.

6. Scene Editing

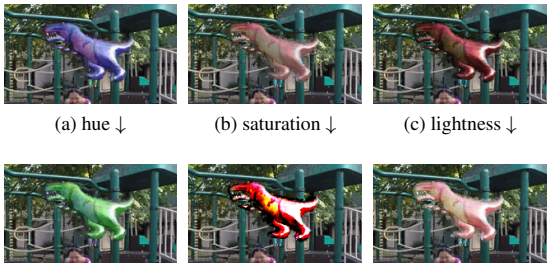
This section focuses on recoloring and two complex editing operations: transformation and composition(Fig. 8).

6.1. Recoloring

Recoloring on HSL is showed in Fig. 7 .

6.2. Transformation Details

Table 4 shows three types of geometric transformation operations (*Shift*, *Scale*, and *Mirror*) and a time-variant operation (*Reverse*). The time-variant transformation allows editing in the temporal dimension. For example, we can reverse the trajectory of moving objects in a time series while maintaining the others (*e.g.*, Fig. 8b shows two boys: the boy in the red box follows the real trajectory to the right



(a) hue ↓ (b) saturation ↓ (c) lightness ↓

(d) hue ↑ (e) saturation ↑ (f) lightness ↑

Figure 7. **Recolor the balloon.**

Table 4. **Transformation Functions.**

	Mapping Function
Shift	$(x, y, z) : (x, y, z) + \delta$
Scale	$(x, y, z) : (x, y, z) \times scale$
Mirror	$(x, yz) : (-x, -y, z)$
Reverse	$(x, y, z, t) : (x, y, z, -t)$

Table 5. **Final σ, c after Transformation.**

$u_i \in \mathcal{T}'$	$(\sigma, c) \leftarrow (\sigma^t c^t)$
$u_i \in \mathcal{T}$, $u_i \notin \mathcal{T}'$	$(\sigma, c) \leftarrow (\sigma^o c^o)$ if reserved $\sigma \leftarrow 0$ if not reserved
$u_i \notin \mathcal{T}$, $u_i \notin \mathcal{T}'$	$(\sigma, c) \leftarrow (\sigma^o c^o)$

side, but the boy in the blue box moves towards the opposite direction and reverses to the left side).

In dynamic scenes, we lack explicit knowledge regarding the distribution of different objects in both the current space and the space after transformation. We can only judge the selected target region by calculating the feature distance. Therefore, we perform two rounds of calculations for all sampled points. In the first round, we use initial positions of all sampled points, dividing them into $\mathcal{T}$ (inside the object) and $\mathcal{S}$ (outside) according to Section 3.3. In the second round, we use the positions after transformation, dividing edited space into $\mathcal{T}'$ and $\mathcal{S}'$. Additionally, for each sampled point u_i, we obtain its original attributes c^o and σ^o, as well as c^t and σ^t after transformation.

To preserve the original object, it is kept intact. Alternatively, if removal is desired, we set $\sigma = 0$. In cases where there are overlapping areas between the transformed object and the original object, we consistently apply c^t and σ^t to assign the properties to these points. The settings for the final density and color can be found in Table 5 provided below.

6.3. Composition Details

Assuming the presence of N objects derived from distinct dynamic scenes, which have been filtered from the original NeRF and are represented using masks in 4D space. Trans-

(a) Composition

(b) Time reversal

Figure 8. **Composition & Time Reversal.** We maintain spatial-temporal consistency across the entire time series when combining different scenes or applying time-variant transformations to individual objects within the same scene.

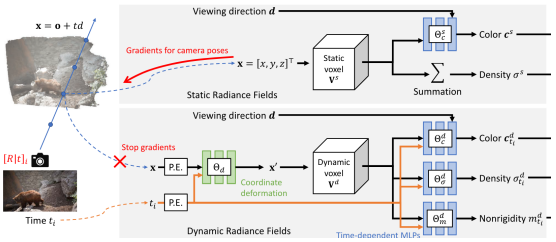
formation can also be applied here to prevent object overlap during composition. The composition is performed at each sample point for all segmented parts collectively:

$$(\sigma', \mathbf{c}') = (\sum_{i=1}^{N} \sigma_i \; mask_i, \sum_{i=1}^{N} c_i \; mask_i) \tag{1}$$

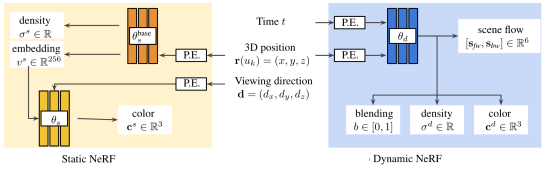
where $mask_i$ denotes the membership of the current point to the i^{th} object. Additionally, blending weights for dynamic objects must be recalculated:

$$b' = \sum_{i=1}^{N} b_i \; mask_i \tag{2}$$

In Fig. 8a, we create a novel scene by separately using moving objects from the *Balloon2* dataset and backgrounds from the *Playground* dataset, and then compositing them together. Subsequently, the rendering formula can be employed to compute the pixel color. The computational complexity of the combination increases in proportion to the number of objects.



(a) Robust NeRF [2]



(b) Dynamic NeRF [4]

Figure 9. **Hybrid Radiance Field Reconstruction of Dynamic Scenes.** The structures of these two methods are directly adopted from their respective theses.

References

[1] 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] 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, 4

[3] 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. 2

[4] 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. 1, 4