

SADG: Segment Any Dynamic Gaussian Without Object Trackers

Yun-Jin Li

Technical University of Munich

yunjin.li@tum.de

Yan Xia

Technical University of Munich

Munich Center for Machine Learning

Yan.Xia@tum.de

Mariia Gladkova

Technical University of Munich

Munich Center for Machine Learning

Mariia.Gladkova@tum.de

Daniel Cremers

Technical University of Munich

Munich Center for Machine Learning

cremers@tum.de

Abstract

Understanding dynamic 3D scenes is fundamental for various applications, including extended reality (XR) and autonomous driving. Effectively integrating semantic information into 3D reconstruction enables holistic representation that opens opportunities for immersive and interactive applications. We introduce SADG, Segment Any Dynamic Gaussian Without Object Trackers, a novel approach that combines dynamic Gaussian Splatting representation and semantic information without reliance on object IDs. In contrast to existing works, we do not rely on supervision based on object identities to enable consistent segmentation of dynamic 3D objects. To this end, we propose to learn semantically-aware features by leveraging masks generated from the Segment Anything Model (SAM) and utilizing our novel contrastive learning objective based on hard pixel mining. The learned Gaussian features can be effectively clustered without further post-processing. This enables fast computation for further object-level editing, such as object removal, composition, and style transfer by manipulating the Gaussians in the scene. We further extend several dynamic novel-view datasets with segmentation benchmarks to enable testing of learned feature fields from unseen viewpoints. We evaluate SADG on proposed benchmarks and demonstrate the superior performance of our approach in segmenting objects within dynamic scenes along with its effectiveness for further downstream editing tasks. Our project page is available at: <https://yunjinli.github.io/project-sadg/>.

1. Introduction

We live in a dynamic, three-dimensional world. Most of the time, we describe our environment at a high level of

objects. As we move around and observe the scene from different viewpoints, we complete the geometry and perceive the colors contributing to our object representation. At the same time, we infer dynamic properties with object motion and deformations. Despite pose, color, and shape changes, the underlying semantic meaning stays constant. This intuition can serve as a strong constraint for neural 3D representations that have gained recent popularity and promote innovation in augmented reality, gaming, and autonomous driving interactive applications. In these technologies, seamlessly integrating semantic information into underlying representation ensures an immersive experience and gives users advanced control over the dynamic scene contents.

Neural Radiance Fields (NeRFs) have revolutionized the field of 3D reconstruction and demonstrated excellent performance in synthesizing photorealistic novel views of a scene from any angle. The limitations of vanilla NeRFs promoted many follow-up works, including those that encompass semantic information by lifting 2D segmentation masks [7, 19, 24] or distilling features [18, 21] and those that faithfully represent dynamic scenes [27, 35, 41]. Due to continuous implicit representation, these methods are computationally intensive as they require re-rendering from multiple views to ensure edit consistency, which hinders their applicability for interactive downstream tasks.

Building on the strengths of NeRFs, Gaussian splatting (3DGS) [17] has introduced a powerful new approach that prioritizes efficiency and speed. Dynamic versions of 3DGS [30, 56] have offered faithful reconstruction of scenes with motion. These models can serve as a geometric prior for learning semantic features that are consistent in both space and time and open new possibilities for real-time interactive applications like scene editing and dynamic interactions. Towards this, a line of works has extended ge-

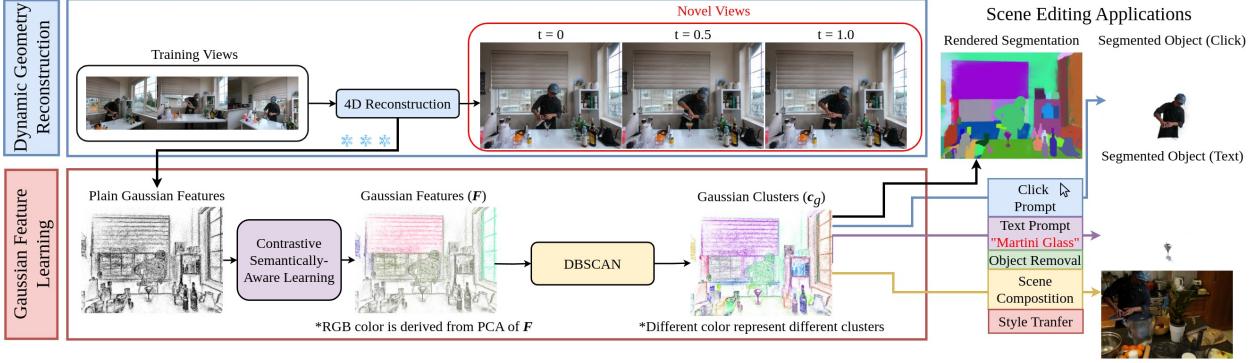


Figure 1. Our pipeline. SADG consists of two main components: dynamic geometry reconstruction (Sec. 3.1) and Gaussian feature learning (Sec. 3.2). We adopt the approach from [56] to effectively learn dynamic 3D reconstruction for both multi-view and single-view sequences. Given 4D reconstruction, we proceed to learn Gaussian features $\mathbf{F} \in \mathbb{R}^{N \times 32}$ using our contrastive semantically-aware learning based on SAM [20] masks. Once the features are properly learned, clustering with DBSCAN [12] is performed directly on the learned Gaussian features. We demonstrate the applicability of our representation on various scene-editing applications. Some of them include segmentation of a target object by click or a text prompt in our GUI, object removal or scene composition, and others.

ometric properties of colored and explicit Gaussians with semantic information [6, 57], producing consistent novel view synthesis and segmentation. Nonetheless, these works are limited to static scenes and cannot handle objects in motion. There are only a few concurrent works [16, 25], which unify semantics and dynamics to comprehensively understand real-world environments. However, SA4D [16] relies on the supervision of object trackers and suffers from semantic identity conflicts in multi-view scenes. Moreover, their method is sensitive to mask noise and requires post-processing to remove degenerate Gaussians. At the same time, DGD [25] has a higher memory requirement and longer training time as they are learning 4D feature fields based on 512-dimensional CLIP [42] or 384-dimensional DINOv2 [34]. Our work aims to tackle both single- and multi-view representations with fast rendering times, minimal post-processing, and low memory footprint.

To this end, we introduce our novel framework, Segment Any Dynamic Gaussian Without Object Trackers (SADG), which effectively combines dynamic Gaussian splatting backbone [56] and semantic information without any reliance on trackers that maintain consistent object identities. To learn the spatio-temporal semantic field, we define a new contrastive objective based on hard positive and negative mining with the Segment Anything Masks (SAM) [20]. We inherit the strengths of zero-shot 2D scene understanding of the foundation model to learn the semantically-aware feature field and produce consistent spatio-temporal segmentation. Our 32-dimensional features are compact and do not depend on features of a 2D segmentation model. Rendering time and storage requirements are almost unaffected, enabling real-time interaction and editing. The learned feature field can be effectively clustered without any further post-processing and utilized in a range of downstream editing tasks, including object style transfer [14], re-coloring,

composition, and removal. Given an early stage of development and benchmarks for semantic segmentation for dynamic scenes, we propose to extend existing novel-view synthesis datasets with semantic benchmarks. The proposed evaluation protocol includes single- and multi-view scenes with challenging motions and diverse contents, allowing extensive testing of semantically-aware latent representations from unseen viewpoints.

Our contributions can be summarized as follows:

- We propose Segment Any Dynamic Gaussian Without Object Trackers (SADG), a new approach that achieves multi-view consistent segmentation of dynamic scenes without any tracking supervision.
- We leverage information from 2D masks to learn semantically-aware latent representation for dynamic scenes using a novel contrastive learning objective.
- We extensively evaluate our method on single- and multi-view scenes across five dynamic novel-view benchmarks and demonstrate state-of-the-art segmentation performance.
- We demonstrate the general nature of our feature space and apply it to several downstream tasks, including object removal, style transfer, and scene composition.
- We offer an interactive framework that allows to edit the scene with a simple mouse click or a text prompt. Our tool does not depend on input IDs or 2D strokes and operates in real-time.

2. Related Works

2.1. Semantic Feature Learning

With the emergence of various vision foundation models, such as Segment Anything Model (SAM) [20], DINO [5], and DINOv2 [34], researchers start to integrate the semantics from these foundation models into their reconstruc-

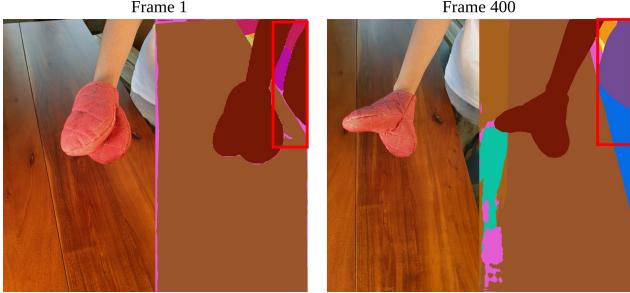


Figure 2. Example of a failure case from the video tracker DEVA [8]. Different colors refer to various object IDs associated by the model. We can observe that due to inconsistent presence of the human torso in the video, DEVA fails to provide reliable and consistent object mask for supervision.

tion. SA3D [7] is the pioneering method that extends 2D SAM masks across images into 3D consistent object masks. GARField [19] introduces hierarchical grouping, leveraging the physical scales of 2D masks to group objects of different sizes and represent scenes at different granularity scales. Gaussian Grouping [57] is a pioneering Gaussian-based approach that utilizes consistent object IDs generated by video tracker DEVA [8] to render object IDs into camera views. Concurrent work such as SAGA [6] utilizes the masks generated from SAM, combined with scales obtained from 3D data inspired by GARField, to segment objects of various sizes. Notably, these methods are designed for static scenes. To date, only two concurrent work, SA4D [16] and DGD [25], addresses the segmentation task in dynamic 3DGS. DGD extends the rasterization pipeline of 3DGS to be able to render semantic feature of each Gaussian. The rendered Gaussian feature map is then trained with the supervision of DINoV2 [34] or CLIP [42] to mimic the output from their feature extractors. The biggest drawback of such supervision is anticipated longer training times as the dimensions of the features generated by DINoV2 and CLIP are quite large (384 and 512). Similar to Gaussian Grouping and its derivatives [11, 31], SA4D relies on video trackers to associate masks across different views to produce consistent object IDs. These approaches face significant limitations, as the mask association often fails to offer consistent object IDs in dynamic scenes, such as the oven-mitts sequence from HyperNeRF [36] as shown in Fig. 2.

2.2. Scene Editing Applications

Learning semantically-aware latent space opens opportunities for interactive downstream applications, allowing users to manipulate scenes on an object level. Several works proposed editing techniques for radiance field representations [13, 49], which are limited to a single object. On the other hand, scene-based methods [24, 53] learn a separate radiance field per object and render the full scene by combining learned opacities along the viewing rays. While the

structure is modular, its editing application is limited to a pre-defined number of objects. Our method is guided by user interaction and is capable of operating on any object in the scene. Another line of work distills features such as DINO [34] into volumetric representation and shows editing capabilities on learned continuous feature field [48]. Due to the implicit representation, scene editing requires a re-assessment of the final consistency by rendering from novel scenes, which impedes the model’s applicability to real-time interactive applications.

At the same time, Gaussian splatting methods mitigate the runtime limitation of NeRFs with their explicit point-based representation. SAGA [57] proposes Local Gaussian Editing scheme, which localize the Gaussians and manipulate them according to the editing task. Feature3DGS [60] distills features from 2D foundation models such as SAM [43] and LSeg [26] enabling promptable and language-driven 3D edits. Nonetheless, these methods operate only on static scenes, while our method is capable of editing dynamic objects and ensuring temporal consistency of the edits. DGD [25] shows semantic texture editing performance by utilizing a pre-trained diffusion model and fine-tuning dynamic 3DGS representation with the reconstruction loss. On the other hand, we demonstrate a variety of editing tasks, including style transfer, object removal, and composition. Despite offering several scene editing opportunities, SA4D [16] has limited user interactivity and requires prior knowledge of object identities. Our framework contains a user-friendly graphical interface that allows operation with a simple text prompt or a mouse click.

3. Method

In this section, we introduce our novel framework SADG. As illustrated in Fig. 1 the pipeline comprises two main components: dynamic geometry reconstruction (Sec. 3.1) and Gaussian feature learning (Sec. 3.2). We adopt the Deformable-3DGS [56] pipeline to reconstruct the 4D scene. Once the 4D reconstruction is learned, we freeze this representation and proceed with Gaussian feature learning using SAM [20] masks and our novel contrastive learning objective in the rendered feature space.

3.1. Dynamic Geometry Reconstruction

We follow the approach proposed in [56] and describe it briefly for completeness. Unlike Dynamic3DGS [30] model, where the reconstruction is stored per frame, resulting in high memory consumption, an MLP learns per-Gaussian deformation ($\delta\mathbf{x}_i, \delta\mathbf{r}_i, \delta\mathbf{s}_i$) with respect to the static canonical space \mathcal{G}_c , which is defined as

$$\begin{aligned} \mathcal{G}_c = \{ & \mathbf{x}_i \in \mathbb{R}^3, \mathbf{r}_i \in \mathbb{R}^4, \mathbf{s}_i \in \mathbb{R}^3, \alpha_i \in \mathbb{R}, \\ & \mathbf{sh}_i \in \mathbb{R}^{3 \times (D_{max}+1)^2} \}_{i=1, \dots, N}. \end{aligned} \quad (1)$$

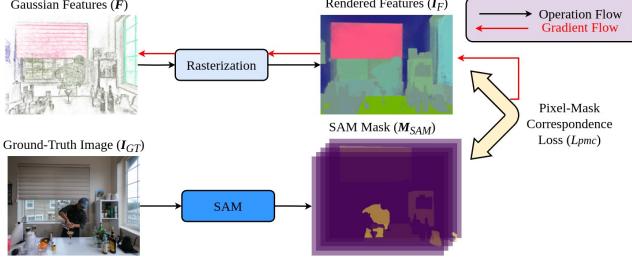


Figure 3. Illustration of rendering Gaussian features and contrastive semantically-aware learning with \mathcal{L}_{pmc} . The Gaussian features \mathbf{F} can be rendered to obtain rendered features \mathbf{I}_F . The rendered features are used to compute the pixel-mask correspondence loss \mathcal{L}_{pmc} with SAM masks \mathcal{M}_{SAM} .

\mathbf{x}_i is the center position of the i -th Gaussian, \mathbf{r}_i is the quaternion representing the rotation, \mathbf{s}_i is the scaling vector, α_i refers to the opacity, and \mathbf{sh}_i is the Spherical Harmonic Coefficients encoding the color information ($D_{\max} = 3$) as per [56].

The resulting Gaussians \mathcal{G}_t at timestamp t are defined as

$$\mathcal{G}_t = \{\mathbf{x}_i + \delta\mathbf{x}_i, \mathbf{r}_i + \delta\mathbf{r}_i, \mathbf{s}_i + \delta\mathbf{s}_i, \alpha_i, \mathbf{sh}_i\}_{i=1,\dots,N}. \quad (2)$$

Once \mathcal{G}_t is obtained, we proceed with the standard rasterization pipeline [17] to render the image \mathbf{I}_r . The color loss $\mathcal{L}_{\text{color}}$ is computed with the ground truth image \mathbf{I}_{GT} as

$$\mathcal{L}_{\text{color}} = (1 - \lambda)\mathcal{L}_1(\mathbf{I}_{\text{GT}}, \mathbf{I}_r) + \lambda\mathcal{L}_{\text{D-SSIM}}(\mathbf{I}_{\text{GT}}, \mathbf{I}_r), \quad (3)$$

where λ refers to the weighting parameter for the structural similarity loss term.

3.2. Gaussian Feature Learning

Problem statement Most of the existing works [16, 57] utilize a video tracker to provide consistent object mask IDs across views and render the segmentation field directly by predicting the per-pixel probability of different object classes. However, such approaches have a significant bottleneck, namely their reliance on perfect object mask association across views. When mask inconsistencies of the same object occur between views, the entire optimization pipeline can break down, leading to sub-optimal results or failure.

Therefore, we propose a method that aims to learn the 3D semantics of objects and leverage them directly for downstream tasks across different views. In other words, instead of predicting per-pixel probabilities during rendering, we focus on integrating semantic information into Gaussians and leverage semantic constancy for learning multi-view consistent spatio-temporal feature field.

Representation Given 4D reconstruction, we incorporate semantic information into the scene. Specifically, we extend its i -th Gaussian with a 32-dimensional semantic feature vector $\mathbf{f}_i \in \mathbb{R}^{32}$, the so-called ‘‘Gaussian feature’’. The

semantically-aware feature field of a scene represented with N Gaussians can be expressed as

$$\mathbf{F} = \begin{bmatrix} \mathbf{f}_1^T \\ \vdots \\ \mathbf{f}_N^T \end{bmatrix} \in \mathbb{R}^{N \times 32}. \quad (4)$$

Rendering Gaussian Features The learned Gaussian features \mathbf{F} can be rendered similarly to [17, 60] with point-based α -rendering [22, 23]. Given a sorted subset of Gaussians \mathcal{G}_p that are related to pixel p , the resulting rendered features $\mathbf{I}_F(p)$ can be computed as follows:

$$\mathbf{I}_F(p) = \sum_{i \in \mathcal{G}_p} \mathbf{f}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \quad (5)$$

Existing methods [6] use smooth Gaussian features \mathbf{f}_i^s instead, as they help to filter out noisy 3D Gaussian features with high similarity scores corresponding to unrelated objects. The smooth Gaussian feature for the i -th Gaussian is computed as per Eq. (6), where $\text{KNN}(i)$ denotes a set of K nearest neighbors [9] of the i -th Gaussian based on its 3D position. We adopt this convention in our work and utilize smooth feature vectors \mathbf{f}_i^s in the rendering process (Eq. (5)).

$$\mathbf{f}_i^s = \frac{1}{K} \sum_{j \in \text{KNN}(i)} \mathbf{f}_j \quad (6)$$

Our contrastive semantically-aware learning objective
The generated set of SAM masks \mathcal{M}_{SAM} comprise M bi-

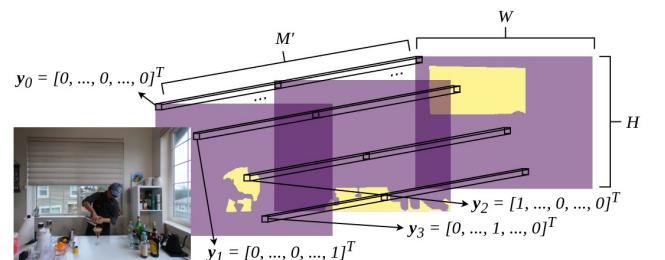


Figure 4. The illustration of pixel-mask correspondence vector \mathbf{y}_i . The pixel-mask correspondence vector is constructed from a subset of SAM masks $\mathcal{M}'_{\text{SAM}}$ for a given image \mathbf{I}_{GT} .

nary masks of a given image \mathbf{I}_{GT} , where each mask M_i corresponds to a different object. In practice, we do not consider all masks from SAM to alleviate memory and computational requirements. Instead, we randomly sample $M' \ll M$ masks for further usage.

We define the pixel-mask correspondence vector \mathbf{y}_i in Eq. (7), which captures mask information for a certain pixel coordinate (u_i, v_i) across all binary masks in $\mathcal{M}'_{\text{SAM}}$. This vector is similar to a one-hot encoding technique; however,

there is no guarantee that a pixel belongs to a single mask due to 2D segmentation inaccuracies. We visually demonstrate the formulation of \mathbf{y}_i in Fig. 4.

$$\mathbf{y}_i = \begin{bmatrix} \mathbf{M}_1(u_i, v_i) \\ \mathbf{M}_2(u_i, v_i) \\ \vdots \\ \mathbf{M}_{M'}(u_i, v_i) \end{bmatrix} \in \{0, 1\}^{M'}, \quad (7)$$

where $u_i \in \{1, 2, \dots, W\}$ and $v_i \in \{1, 2, \dots, H\}$.

In practice, we only sample N_p pixels from an image such that $N_p \ll W \times H$. Mathematically, we define the sampled pixel coordinates as a set $\mathcal{P} = \{(u_i, v_i)\}_{i=1,2,\dots,N_p}$.

The pairwise similarities of the sampled N_p pixel-mask correspondence vectors can be retrieved by computing the Gram matrix of their stacked matrix $\mathbf{Y}^{\mathcal{P}} \in \{0, 1\}^{N_p \times M'}$. All entries of the resulting Gram matrix are then bounded to either 0 (negative pair) or 1 (positive pair) to form a mask-based correspondence matrix $\mathbf{C} \in \{0, 1\}^{N_p \times N_p}$ among sampled pixels \mathcal{P} , formally defined as

$$\mathbf{C}(i, j) = \begin{cases} 1, & \text{if } \mathbf{y}_i(u_i, v_i)^T \cdot \mathbf{y}_j(u_j, v_j) > 0 \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

Additionally, we formulate a correspondence matrix \mathbf{C}_F for the sampled rendered features $\mathbf{I}_F^{\mathcal{P}}$. Analogous to the mask-based correspondence matrix \mathbf{C} , we compute a feature-based Gram matrix, capturing the pairwise similarities between the features as follows:

$$\mathbf{C}_F = \mathbf{I}_F^{\mathcal{P}} (\mathbf{I}_F^{\mathcal{P}})^T \in \mathbb{R}^{N_p \times N_p}, \quad (9)$$

where $\mathbf{I}_F^{\mathcal{P}}$ is defined as follows:

$$\mathbf{I}_F^{\mathcal{P}} = \begin{bmatrix} \mathbf{I}_F(u_1, v_1)^T \\ \vdots \\ \mathbf{I}_F(u_{N_p}, v_{N_p})^T \end{bmatrix} \in \mathbb{R}^{N_p \times 32}. \quad (10)$$

Finally, we have all the necessary ingredients to formulate the contrastive learning objective based on a hard positive and negative mining scheme. Guided by the mask-based correspondence matrix, we enforce features to be close for pixels from the same object mask and far apart for the distinct objects. Furthermore, we concentrate on hard pixel pairs. Specifically, we sample K_p positive samples with low similarity scores and K_n negative samples with high similarity scores. With this, we formally define the

optimization objective as

$$\begin{aligned} \mathcal{L}_{\text{pmc}} = & \frac{1}{K_n} \sum_{i=0}^{N_p-1} \sum_{j=i+1}^{N_p-1} \mathbb{I}_0[\mathbf{C}(i, j)] \cdot \mathbb{I}_{\mathcal{H}\mathcal{N}}[\mathbf{C}_F(i, j)] \cdot \mathbf{C}_F(i, j) - \\ & \frac{1}{K_p} \sum_{i=0}^{N_p-1} \sum_{j=i+1}^{N_p-1} \mathbb{I}_1[\mathbf{C}(i, j)] \cdot \mathbb{I}_{\mathcal{H}\mathcal{P}}[\mathbf{C}_F(i, j)] \cdot \mathbf{C}_F(i, j), \end{aligned} \quad (11)$$

Note that $\mathbb{I}_A[x]$ is an indicator function. It returns 1 when $x \in A$, and 0 otherwise. $\mathcal{H}\mathcal{P}$ stands for hard positive which is defined as $\mathcal{H}\mathcal{P} = \{x | x < T_p\}$ and $\mathcal{H}\mathcal{N}$ refers to a hard negative which is defined as $\mathcal{H}\mathcal{N} = \{x | x > T_n\}$. Here, $T_p = 0.75$ and $T_n = 0.5$ are the positive and negative thresholds for pairwise similarity respectively. As both \mathbf{C}_F and \mathbf{C} are symmetric matrices following the property of the Gram matrix, we only consider entries in the upper triangular part and exclude the diagonal to avoid repeated computation. We show rendered Gaussian features $\mathbf{F} \in \mathbb{R}^{N \times 32}$ learned with our contrastive scheme in Fig. 1. For demonstration, they are compressed into $\mathbf{F}' \in \mathbb{R}^{N \times 3}$ via PCA [33], allowing each rendered feature to be represented with RGB color.

Gaussian Feature Clustering To enable interactive applications with the learned 4D representation, we deploy the DBSCAN algorithm [12] to cluster the Gaussian features into several groups and generate the so-called Gaussian Clusters c_g as defined in Eq. (12). $c_g(i)$ provides the corresponding cluster ID for i -th Gaussian feature:

$$c_g \in \{1, 2, \dots, N_c\}^N. \quad (12)$$

Note that N_c refers to the total number of clusters. We demonstrate them visually in Fig. 1. As can be observed, they represent distinct objects in 4D space.

Since a typical scene contains a large number of Gaussians (more than 300k), we adopt the sub-sampling scheme proposed by [6], where only 2% of the Gaussian features are used for clustering. We propagate the cluster IDs to the remaining 98% of the Gaussians by computing the feature similarity between cluster-representative Gaussians. In the supplementary material, we demonstrate that the performance is not affected by a significant reduction in the number of Gaussians contributing to the generation of clusters.

4. Experimental Results

We evaluate the performance of SADG across several well-known datasets to validate its effectiveness. We begin by introducing the datasets in Sec. 4.1. Since there are no established benchmarks for validating the segmentation performance on novel views in dynamic scenes, we provide

details on how we extend these different datasets with segmentation benchmarks (Sec. 4.2). Next, we explore the implementation details and various configurations used to train and test methods on these benchmarks (Sec. 4.3). We present segmentation results of our model, along with both quantitative and qualitative comparisons to other related works [6, 25, 57] in Sec. 4.5. Last but not least, we demonstrate capabilities of our learned semantic representation on a number of downstream tasks including scene editing and selection in our a graphical user interface (Sec. 4.6).

4.1. Datasets

NeRF-DS dataset [52] contains scenes with challenging transparent and shiny moving objects recorded by two monocular cameras. Five sequences are used for reconstruction and evaluation including *as_novel_view*, *basin_novel_view*, *cup_novel_view*, *press_novel_view*, and *plate_novel_view*.

HyperNeRF dataset [36] is recorded using a single monocular camera . The camera moves while capturing the dynamic scenes involving different human-object interactions. In our experiments, we use ten sequences: *americano*, *split-cookie*, *oven-mitts*, *espresso*, *chickchicken*, *hand1-dense-v2*, *torchocolate*, *slice-banana*, *keyboard*, and *cut-lemon1*.

Neu3D dataset [27] contains videos captured by a multi-view rig with 18 to 21 cameras each at 30 FPS. The scenes involve a person performing cooking tasks in the kitchen. In total, five scenes are used in our experiments: *coffee_martini*, *cook_spinach*, *cut_roasted_beef*, *flame_steak*, and *sear_steak*.

Google Immersive dataset [2] consists of videos captured by a 46-camera rig at 30 FPS. The cameras are fish-eye cameras and are distributed on the surface of a hemispherical, 92cm diameter dome. Four scenes are used in our experiments: *01_Welder*, *02_Flames*, *10_Alexa_1*, and *11_Alexa_2*. **Technicolor Light Field dataset** [44] captures scenes using a 4×4 camera array at 30 FPS. We use the undistorted images provided by the authors, and four scenes are used in our experiments: *Birthday*, *Fabien*, *Painter*, and *Theater*.

4.2. Segmentation Benchmarks

To date, while numerous benchmarks exist for dynamic video segmentation, such as DAVIS [3, 4, 38, 40] and VOS [51, 54, 55], there are still no well-established segmentation benchmarks for dynamic novel view synthesis.

Such benchmarks are crucial for evaluating a comprehensive understanding of scenes, both geometrically and semantically. Existing datasets, such as Replica [47], ScanNet [10], and ScanNet++ [58], include ground-truth labels for evaluating panoptic segmentation performance, but they do not cover dynamic sequences. To address this gap, we manually annotate dynamic sequences in our experiments and propose the *NeRF-DS-Mask*, *HyperNeRF*-

Mask, *Neu3D-Mask*, *Immersive-Mask*, and *Technicolor-Mask* benchmarks. Utilizing the powerful SAM2 model [43], which provides consistent object masks across video sequences, we manually define the objects of interest for each sequence to evaluate their segmentation performance in our model. Examples of these mask objects can be seen in the supplementary.

4.3. Implementation Details

We employ SAM [20] to generate the anything-masks for Gaussian feature learning. However, the resulting masks are memory-inefficient because they are stored in a dense format ($M \times W \times H$) with a lot of pixels with zero values. To optimize memory usage, we save the masks as a bit array.

Given a permuted tensor version of an RGB image $I \in \mathbb{R}^{3 \times W \times H}$, SAM generates masks $\mathcal{M} \in \mathbb{R}^{M \times W \times H}$. We first flatten \mathcal{M} , then convert flattened ($M \times W \times H$)-dimensional array into a bit-array. The final output is stored in a dictionary with the required information to re-create the original mask tensor: a number of masks M , image dimensions (W, H), and the compressed bit array, i.e., M, W, H , bitarray. This approach significantly reduces the storage used and is particularly useful for multi-view datasets, where numerous masks from different views are generated. For instance, a raw mask generated from an image in *coffee_martini* occupies around **80 MB**, but the dictionary representation is significantly reduced to approximately **1 MB**. Due to space limits, we provide training details for different datasets in the supplementary.

4.4. Evaluation Protocol and Baselines

Given ground truth masks we use Mean Intersection over Union (mIoU) and Mean Pixel Accuracy (mAcc) to quantitatively evaluate segmentation performance of the models. We compare our model with SAGA [6], Gaussian Grouping [57] and DGD [25]. Note that both SAGA and Gaussian Grouping are designed for static scenes and do not include a module to encode temporal information. To address this limitation, we integrate our 4D reconstruction backbone into their pipeline, extending their approach to effectively handle dynamic scenes. Moreover, Gaussian Grouping needs the precomputed object IDs from DEVA [8], whose labels are inconsistent for multi-view camera setup. Thus, we train their semantic module on a monocular stream from the camera closest to the test view. Furthermore, all reported baselines utilize the same pre-trained 4D reconstruction, with semantics learned according to each pipeline. This allows us to isolate and evaluate the performance of each model based on its semantic learning capabilities. After training, we manually perform click prompts in the rendered novel views of each scene, which are trained by different approaches to select the objects of interest defined in the benchmark. Apart from the quantitative results, we

Method	NeRF-DS-Mask		HyperNeRF-Mask		Neu3D-Mask		Immersive-Mask		Technicolor-Mask	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
Gaussian Grouping [57]	0.8351	0.9729	0.8341	0.9771	0.8864*	0.9932*	0.7673*	0.9776*	0.7979*	0.9800*
SAGA [6]	0.8072	0.9644	0.7660	0.9579	0.6941	0.9516	0.7395	0.9747	0.7913	0.9792
DGD [25]	0.7125	0.9551	0.8297	0.9777	0.7721	0.9851	0.7981	0.9850	0.7791	0.9728
SADG (Ours)	0.8719	0.9826	0.8663	0.9845	0.9022	0.9945	0.9234	0.9945	0.9308	0.9917

Table 1. Quantitative comparison of our method’s performance against SOTA approaches on multi-view dynamic segmentation. SADG outperforms the baselines on average mIoU and mAcc. The details for each sequence can be found in the supplementary. *: Gaussian Grouping needs the precomputed object IDs from DEVA, whose supervision is inconsistent for multi-view camera setup. Thus, we train their semantic module on monocular stream from the camera that is closest to the test view.

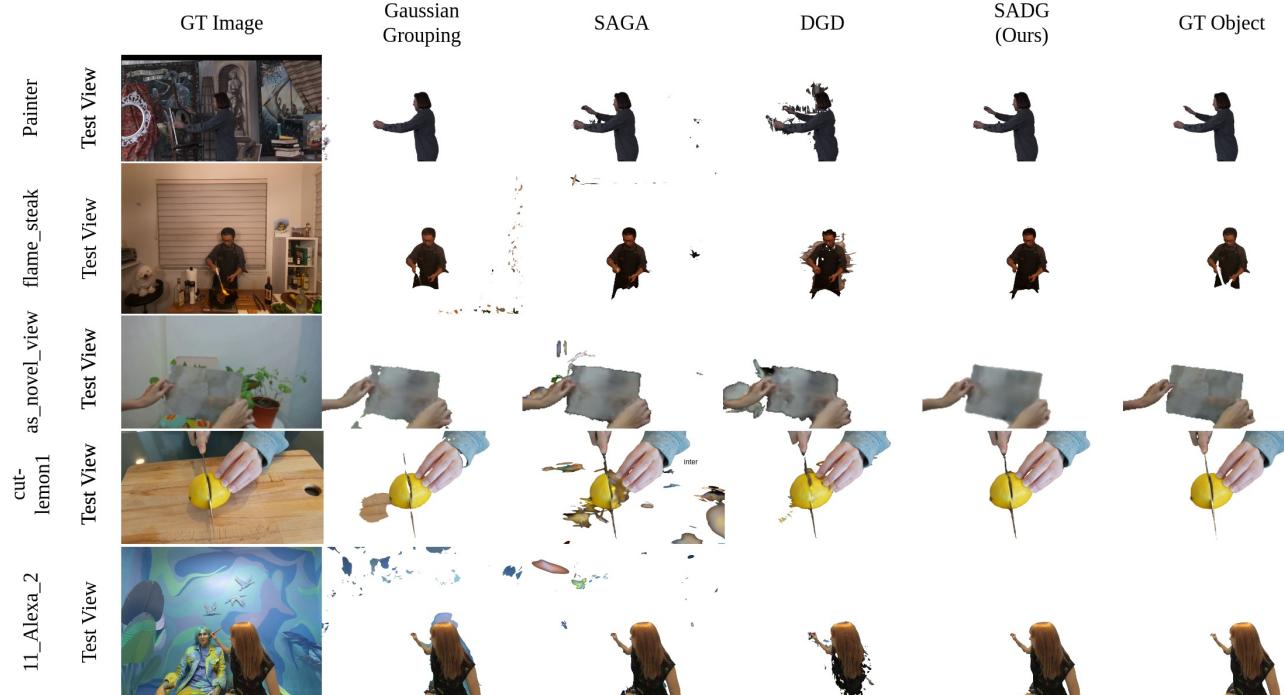


Figure 5. Segmentation qualitative results. We compare our performance on the proposed segmentation benchmark against dynamic 3D segmentation baselines, such as Gaussian Grouping [57], SAGA [6], DGD [25]. Our model consistently demonstrates superior segmentation quality and crisp masks without reliance on tracking supervision or post-processing.

also present qualitative results from various benchmarks, as certain insights cannot be fully captured through numerical evaluations alone. Since the official source code for SA4D [16] is not available, we report the comparison based on unofficial implementation in the supplementary.

4.5. Segmentation Results

As shown in Tab. 1, SADG consistently outperforms all other methods across all benchmarks in terms of average mIoU and mAcc. We also select some sequences to visually illustrate the quality of the segmentation of each model in Fig. 5. These comparisons reveal that SADG qualitatively surpasses existing models and exhibits multi-view consistent segmentation performance. Notably, SADG maintains strong generalization to test camera views in multi-view sequences such as *flame_steak*, *Painter*, and *11_Alexa_2*. This robust performance can be attributed to the compact Gaus-

sian features learned through the proposed Gaussian feature training. In contrast, DGD segments objects by comparing cosine similarity scores between features retrieved via click prompts. However, determining a suitable threshold that generalizes across various scenes is neither straightforward nor intuitive.

SADG employs clustering based on the Gaussian features to group the Gaussians into several segments. As this operation is done in 3D space, SADG achieves cross-view consistency and does not need to associate objects from different views. This advantage is evident in *cut-lemon1* example in Fig. 5, where wrongly associated object IDs can cause segmentation failure for Gaussian Grouping.

4.6. Scene Editing Applications

In addition to accurate segmentation capabilities and multi-view consistent rendered masks, we demonstrate the effec-



Figure 6. Versatile scene editing applications. (a) The object of interest can be selected by click prompts or text prompts. (b) Scene composition can be done by manipulating the selected Gaussians in another scene. (c) Style transfer of the segmented objects to obtain different textures. (d) Object removal for the selected object.

tiveness of our learned semantic feature field in a range of downstream tasks. Specifically, we focus on interactive editing applications and develop a graphical user interface (GUI) to grant full user control over the representation. Thanks its explicit nature and our semantic abstraction to Gaussian clusters, we achieve real-time operation capability in handling many editing tasks. We provide snapshots of the GUI in the supplementary.

Besides simple click prompts in the novel view of dynamic scenes for object selection, we also support text-based prompts to make object selection more intuitive. To enhance 3D content editing in dynamic scenes, we integrate style transfer capabilities for the selected objects. By segmenting objects directly in 3D, we facilitate scene composition and object removal through straightforward manipulation of the selected Gaussians. Due to space limit, we shall provide more qualitative results in the supplementary. In the following we describe the implementation details of editing tasks based on our semantically-aware feature field.

Text Prompt. Inspired by Gaussian Grouping [57], we employ Grounding DINO [28] to generate 2D mask based on the text prompt in the novel view. The 2D mask is re-projected into the 3D space with the rendered depth information. Finally, we associate each reprojected point with its nearest Gaussian in the 3D scene and retrieve its corresponding cluster. If the number of associated Gaussian of a given cluster is more than a threshold, the cluster would be selected. The example of text prompts on different sequences is shown in Fig. 6 (a).

Scene Composition. We also perform qualitative results on scene composition. As the segmentation is performed in 3D, we can simply put the selected Gaussians from the dynamic scene into another static or dynamic scene. The scales and coordinate transformation need to be adapted.

We show some examples on scene composition in Fig. 6.

Object Style Transfer. We adopt the static Gaussian style transfer from StyleSplat [14] for dynamic scenes. Given a rendered image I_r and a style image I_s , we extract their feature maps (F_r, F_s respectively) from VGG16 [46] and optimize only the SHs of the Gaussians of the selected cluster with the nearest-neighbor feature matching (NNFM) loss [59]. Fig. 6 (c) illustrates an example of style transfer on the segmented object.

Object Removal. We can also remove object by simply deleting Gaussians belonging to the specific cluster as illustrated in Fig. 6 (d). It is worth noting that no inpainting is required due to the underlying 3D representation, where exposed parts are learned by multi-view observations.

5. Conclusion

We introduced a novel framework for dynamic scene understanding, which enables multi-view consistent segmentation without any object tracking supervision. Our SADG effectively combines dynamic 3D Gaussian Splatting 3DGS [56] and 2D SAM [20] masks in a contrastive learning objective, which lifts semantic information into 3D space and learns expressive Gaussian features based on hard pixel cases. This results in cross-view consistency when rendering segmented objects, enhancing the quality and coherence of object segmentation across different views. Evaluated on various novel-view datasets, SADG shows superior performance both quantitatively and qualitatively. We further demonstrated the effectiveness of the learned feature field on downstream editing tasks such as point and text prompts, style transfer, object removal, and scene composition. Our approach sets a strong foundation for further research into dynamic scene understanding and scene editing, especially in complex multiview scenarios.

References

- [1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. *CVPR*, 2022. 8
- [2] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew DuVall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. 39(4):86:1–86:15, 2020. 6
- [3] Sergi Caelles, Alberto Montes, Kevis-Kokitsi Maninis, Yuhua Chen, Luc Van Gool, Federico Perazzi, and Jordi Pont-Tuset. The 2018 davis challenge on video object segmentation. *arXiv:1803.00557*, 2018. 6
- [4] Sergi Caelles, Jordi Pont-Tuset, Federico Perazzi, Alberto Montes, Kevis-Kokitsi Maninis, and Luc Van Gool. The 2019 davis challenge on vos: Unsupervised multi-object segmentation. *arXiv:1905.00737*, 2019. 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 the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 2
- [6] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment any 3d gaussians. *arXiv preprint arXiv:2312.00860*, 2023. 2, 3, 4, 5, 6, 7, 10
- [7] Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, et al. Segment anything in 3d with nerfs. *Advances in Neural Information Processing Systems*, 36:25971–25990, 2023. 1, 3, 5
- [8] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. Tracking anything with decoupled video segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1316–1326, 2023. 3, 6
- [9] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967. 4
- [10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 6
- [11] Bin Dou, Tianyu Zhang, Yongjia Ma, Zhaohui Wang, and Zejian Yuan. Cosseggaussians: Compact and swift scene segmenting 3d gaussians. *arXiv preprint arXiv:2401.05925*, 2024. 3
- [12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–231, 1996. 2, 5
- [13] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-centric neural scene rendering. *arXiv preprint arXiv:2012.08503*, 2020. 3
- [14] Sahil Jain, Avik Kuthiala, Prabhdeep Singh Sethi, and Prakanshul Saxena. Stylesplat: 3d object style transfer with gaussian splatting, 2024. 2, 8
- [15] RC Jancey. Multidimensional group analysis. 1966. 3
- [16] Shengxiang Ji, Guanjun Wu, Jiemin Fang, Jiazhong Cen, Taoran Yi, Wenyu Liu, Qi Tian, and Xinggang Wang. Segment any 4d gaussians. *arXiv preprint arXiv:2407.04504*, 2024. 2, 3, 4, 7, 5, 8, 9
- [17] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. 1, 4
- [18] Justin* Kerr, Chung Min* Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023. 1
- [19] Chung Min Kim, Mingxuan Wu, Justin Kerr, Ken Goldberg, Matthew Tancik, and Angjoo Kanazawa. Garfield: Group anything with radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21530–21539, 2024. 1, 3
- [20] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, Piotr Dollar, and Ross B Girshick. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 2, 3, 6, 8, 1
- [21] 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
- [22] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-based neural rendering with per-view optimization. In *Computer Graphics Forum*, pages 29–43. Wiley Online Library, 2021. 4
- [23] Georgios Kopanas, Thomas Leimkühler, Gilles Rainer, Clément Jambon, and George Drettakis. Neural point catacaustics for novel-view synthesis of reflections. *ACM Transactions on Graphics (TOG)*, 41(6):1–15, 2022. 4
- [24] 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. 1, 3
- [25] Isaac Labe, Noam Issachar, Itai Lang, and Sagie Benaim. Dgd: Dynamic 3d gaussians distillation, 2024. 2, 3, 6, 7, 4, 5, 10
- [26] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and René Ranftl. Language-driven semantic segmentation. *arXiv preprint arXiv:2201.03546*, 2022. 3
- [27] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022. 1, 6

- [28] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 8
- [29] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 3
- [30] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 1, 3
- [31] Weijie Lyu, Xuetong Li, Abhijit Kundu, Yi-Hsuan Tsai, and Ming-Hsuan Yang. Gaga: Group any gaussians via 3d-aware memory bank. *arXiv preprint arXiv:2404.07977*, 2024. 3
- [32] J Macqueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*, 1967. 3
- [33] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers and Geosciences*, 19(3):303–342, 1993. 5
- [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, 2023. 2, 3
- [35] 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 the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 1
- [36] 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. *arXiv preprint arXiv:2106.13228*, 2021. 3, 6, 1
- [37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1
- [38] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016. 6
- [39] Joaquín Pérez-Ortega, N Nely Almanza-Ortega, Andrea Vega-Villalobos, Rodolfo Pazos-Rangel, Crispín Zavala-Díaz, and Alicia Martínez-Rebollar. The k-means algorithm evolution. *Introduction to data science and machine learning*, pages 69–90, 2019. 3
- [40] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbelaez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 6
- [41] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 1
- [42] 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, 3
- [43] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 3, 6, 5
- [44] Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbiriou, Frederic Babon, Matthieu Hog, Remy Gendrot, Tristan Langlois, Olivier Bureller, Arno Schubert, et al. Dataset and pipeline for multi-view light-field video. In *Proceedings of the IEEE conference on computer vision and pattern recognition Workshops*, pages 30–40, 2017. 6
- [45] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 5
- [46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 8
- [47] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 6
- [48] 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. 3
- [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 the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 3
- [50] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024. 5
- [51] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas S. Huang. Youtube-vos: A large-scale video object segmentation benchmark. *CoRR*, abs/1809.03327, 2018. 6

- [52] Zhiwen Yan, Chen Li, and Gim Hee Lee. Nerf-ds: Neural radiance fields for dynamic specular objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8285–8295, 2023. [6](#), [3](#), [5](#)
- [53] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13779–13788, 2021. [3](#)
- [54] Linjie Yang, Yuchen Fan, and Ning Xu. The 2nd large-scale video object segmentation challenge - video object segmentation track, 2019. [6](#)
- [55] Linjie Yang, Yuchen Fan, and Ning Xu. The 4th large-scale video object segmentation challenge - video object segmentation track, 2022. [6](#)
- [56] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024. [1](#), [2](#), [3](#), [4](#), [8](#)
- [57] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. 2024. [2](#), [3](#), [4](#), [6](#), [7](#), [8](#), [1](#), [10](#)
- [58] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2023. [6](#)
- [59] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *European Conference on Computer Vision*, pages 717–733. Springer, 2022. [8](#)
- [60] Shijie Zhou, Haoran Chang, Sicheng Jiang, Zhiwen Fan, Zehao Zhu, Dejia Xu, Pradyumna Chari, Suya You, Zhangyang Wang, and Achuta Kadambi. Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21676–21685, 2024. [3](#), [4](#)

SADG: Segment Any Dynamic Gaussian Without Object Trackers

Supplementary Material

In the supplementary, we provide further implementation details in Sec. 6 and Sec. 7. A range of ablation studies on different components of our pipeline is given in Sec. 8. We compare our SADG with the concurrent work on 4D segmentation and editing, Segment Any 4D Gaussians (SA4D), in Sec. 9. Several examples from our segmentation benchmark are given in Sec. 11. We present detailed quantitative results on all benchmarks in Sec. 10. We finally provide qualitative results of our method on segmentation (Sec. 12, Sec. 14) and editing (Sec. 13).

6. Training Implementation Details

We implement the whole pipeline in PyTorch [37]. We reuse the codebase from Deformable-3DGS [56] and extend the differential Gaussian rasterization pipeline to render the Gaussian features similar to [57]. In the following, we introduce the detailed training setup for different datasets.

6.1. NeRF-DS and HyperNeRF

For the NeRF-DS sequences, we use the left camera for training and the right camera for testing. In the HyperNeRF sequences, training frames are sampled every 4 frames, starting from the first frame, while testing frames are selected every 4 frames, starting from the third frame. This follows the interp-sequence data-loading pipeline from the original HyperNeRF paper [36], ensuring that all testing frames are unseen during training. Note that we use $2\times$ downsized images for all experiments.

Dynamic Geometry Reconstruction. We adopt the default hyper-parameters from Deformable-3DGS [56]. Initially, the Gaussians are warmed up without the deformation MLP for the first 3k iterations to better capture the positions and shape stably as suggested in [56], after which the MLP optimization begins, and the 3D Gaussian and the deformation MLP are trained jointly. The densification of Gaussians for adaptive density control introduced in 3DGS [17] is set to run until 15k iterations. The dynamic geometry reconstruction is trained for a total of 20k iterations.

Gaussian Feature Learning. After the dynamic reconstruction is completed, we freeze model’s weights and begin Gaussian feature learning for an additional 10k iterations. For each image, we randomly sample 25 masks ($M' = 25$) generated by SAM [20] and 5k pixels ($N_p = 5000$), with $K = 16$ to get smooth Gaussian features.

6.2. Neu3D

Each video contains 300 frames, and we begin by downscaling the images by a factor of 2, resulting in a resolution

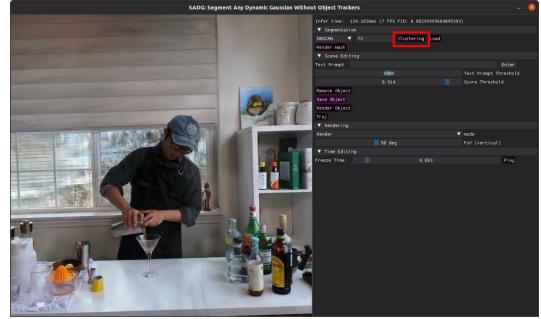


Figure 7. Example of our GUI operation. After opening the GUI, click “Clustering” to perform Gaussian feature clustering.

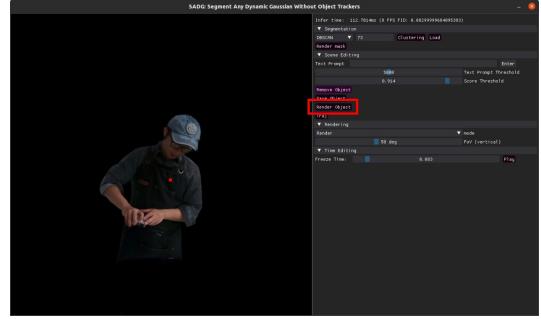


Figure 8. Example of our GUI operation. Hold the key “A” and click (■) on the objects of interest in the novel view to segment the target objects.

of 1352×1014 . The center camera (camera_00) is used for testing, while the remaining cameras are for training. Initially, we run SfM [45] across all the first frames, excluding camera_00, to get the initial point cloud.

Dynamic Geometry Reconstruction. We follow the same pipeline for dynamic geometry reconstruction as described in Sec. 6.1, except the densification of Gaussians is set to 8k iterations.

Gaussian Feature Learning. After the dynamic reconstruction is completed, we freeze the model’s weights and begin Gaussian feature learning for an additional 10k iterations. For each image, we randomly sample 50 masks ($M' = 50$) generated by SAM [20] and 10k pixels ($N_p = 10000$), with $K = 16$ to get the smooth Gaussian feature.

6.3. Google Immersive

Each video contains frames captured by fish-eye cameras with 300 frames. We first undistort the images and align the principal points to the center, then downscale them to a resolution of 1280×960 .

Similar to Neu3D, the center camera (camera_0001) is used for testing, while the other cameras are used for train-

Method	NeRF-DS-Mask		HyperNeRF-Mask		Neu3D-Mask		Immersive-Mask		Technicolor-Mask	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
SADG (w/o filtering)	0.8719	0.9826	0.8368	0.9790	0.8738	0.9923	0.9154	0.9942	0.9258	0.9904
Ours (w/ filtering)	0.8719	0.9826	0.8663	0.9845	0.9022	0.9945	0.9234	0.9945	0.9308	0.9917

Table 2. Comparison of average mIoU and mAcc per benchmark of our method with (w/) filtering and without (w/o) filtering. Filtering improves the performance and validates our approach.

Method	NeRF-DS-Mask		HyperNeRF-Mask		Neu3D-Mask		Immersive-Mask		Technicolor-Mask	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
SADG (K=0)	0.8228	0.9706	0.8360	0.9787	0.8661	0.9920	0.8725	0.9909	0.9259	0.9908
Ours (K=16)	0.8719	0.9826	0.8368	0.9790	0.8738	0.9923	0.9154	0.9942	0.9258	0.9904
SADG (K=32)	0.8754	0.9828	0.8331	0.9780	0.8702	0.9922	0.9217	0.9944	0.9255	0.9904

Table 3. Comparison of average mIoU and mAcc per benchmark with K=0 (no smooth Gaussian features), K=16, and K=32. Our approach strikes a balance between performance and runtime requirements.

ing. We generate the initial point cloud using SfM across all the first frames, excluding the test camera, and apply the same hyper-parameters as in Neu3D in Sec. 6.2.

6.4. Technicolor Light Field

We use the undistorted images provided by the authors, aligning the principal points to the center and resizing the images to a resolution of 1024×544 . Similar to Neu3D and Google Immersive dataset, the center camera (camera_0000) is used for testing, while the other cameras are used for training. We generate the initial point cloud using SfM across all the first frames excluding the test camera and apply the same hyper-parameters as in Neu3D in Sec. 6.2.

7. Testing Implementation Details

After the dynamic geometry reconstruction and the Gaussian features are properly trained, we proceed to our GUI to select the object of interest and evaluate the performance of the model by mIoU and mAcc as discussed in Sec. 4.4.

Given a clicked point $[u, v]^T$ on the novel view, we first use the known camera intrinsics to compute the inverse intrinsic matrix \mathbf{K}^{-1} . With the depth information d rendered by the dynamic reconstruction of the Gaussians \mathcal{G}_t at time t , we can reproject the 2D clicked point to 3D coordinates in the camera frame. Finally, using the known camera extrinsics (rotation matrix \mathbf{R} and translation vector \mathbf{T}) for the world-to-camera transformation, we convert the points back into the world coordinate frame. We can then use this point coordinate to query its nearest neighbor Gaussian using KNN. The cluster ID corresponding to its nearest neighbor Gaussian is subsequently retrieved.

We provide interactive illustrations in our GUI to segment target objects as follows:

1. Open the GUI and run DBSCAN (Click button "Clustering") to group Gaussians into different clusters based on the learned Gaussian features as illustrated in Fig. 7.

2. Click the objects of interest in the novel view to segment the target objects as shown in Fig. 8.
3. Click the button "Render Object" to render the selected object to test views for evaluating the performance.

8. Ablation Studies

8.1. Filtering

We noticed that the raw segmented object from the selected Gaussian clusters sometimes has some artifacts near the outline of the objects as illustrated in Fig. 9. Therefore, we adopt a simple yet effective filtering step to filter out those artifacts. The procedure of the filtering is as follows:

- In each selected Gaussian cluster c , we compute the mean Gaussian features \mathbf{f}_m^c in the selected cluster c . $\mathbf{f}_m^c = \frac{1}{N_c} \sum_{\mathbf{f}_g^c \in c} \mathbf{f}_i$
- We compute the per-Gaussian similarity score between the Gaussian features inside the selected cluster c and mean Gaussian features \mathbf{f}_m^c of the selected cluster with cosine similarity.
- Delete Gaussians that have lower similarity scores than the threshold T .

As shown in Tab. 2, the proposed filtering step can effectively improve the average mIoU and mAcc for each benchmark.

8.2. Smooth Gaussian Features

In Sec. 3.2 we introduced smooth Gaussian features \mathbf{f}_i^s (Eq. (6)). We provide a detailed analysis of the selection of the number of nearest neighbors for the KNN algorithm in Tab. 3. Overall, when smooth Gaussian features are enabled ($K = 16$ and $K = 32$), the performance is better than when smooth Gaussian features are disabled ($K = 0$). Since there is no significant improvement between $K = 16$ and $K = 32$, we eventually chose $K = 16$ for our pipeline to avoid additional overheads with the KNN sampling.

NeRF-DS-Mask			HyperNeRF-Mask			Neu3D-Mask			Immersive-Mask			Technicolor-Mask			
Avg. Num. Gaussians	186455.4		565614.7				678566		985713				457677.75		
Method	mIoU	mAcc	time↓	mIoU	mAcc	time↓	mIoU	mAcc	time↓	mIoU	mAcc	time↓	mIoU	mAcc	time↓
Ours (2%)	0.8719	0.9826	0.4068	0.8368	0.9790	2.3117	0.8738	0.9923	2.2506	0.9154	0.9942	9.6083	0.9258	0.9904	1.6118
SADG (10%)	0.8707	0.9828	1.5782	0.8384	0.9794	11.8430	0.8792	0.9930	6.1878	0.9160	0.9939	71.2429	0.9283	0.9908	6.8211
SADG (25%)	0.8762	0.9832	3.9425	0.8461	0.9809	43.2682	0.8886	0.9935	18.8976	0.9180	0.9943	221.7642	0.9297	0.9907	19.0767

Table 4. Comparison of average mIoU, mAcc, and clustering time per benchmark with different sub-sampling of Gaussians 2%, 10%, and 25%.

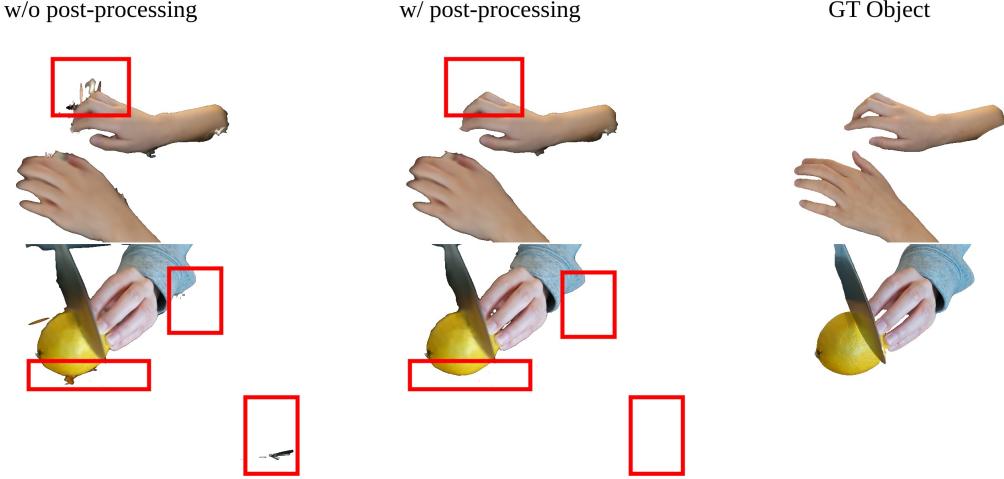


Figure 9. Visualization of ground-truth object of interest, segmented object without (w/o) filtering, and segmented object with (w/) filtering.

8.3. Gaussian Feature Clustering

In Sec. 3.2 we additionally introduced Gaussian Feature Clustering and the sub-sampling strategy to run the clustering in real-time with scenes that normally contain more than 300k Gaussians. We perform time analysis with different sub-sampling numbers of Gaussians (2%, 10%, and 25%) on RTX3080 and Intel i7-12700. For each scene, we compute the time for clustering using Python *timeit* package and take the average clustering time for five runs per scene. We report the average clustering time across all scenes in each benchmark and the corresponding mIoU and mAcc for our object of interest in Tab. 4. Overall, with different sub-sampling of Gaussians (2%, 10%, and 25%), the mIoU and mAcc stay almost unchanged. However, as increasing the number Gaussians results in a longer clustering time, we believe that 2% is the optimal sub-sampling number for clustering.

8.4. DBSCAN vs. K-Means

There are various clustering techniques and K-means [15, 29, 32, 39] is one of the most widely used. K-means partitions data into k clusters, where each feature belongs to the cluster with the nearest centroid (mean of the cluster). The value k must be predefined, and the algorithm iteratively adjusts the centroids to minimize the sum of squared distances between features and their corresponding cluster

centers. Therefore, the performance of the algorithm heavily depends on the optimal value of k .

To investigate this, we conducted an additional ablation study comparing the rendered segmentation produced by K-means with different values of k and DBSCAN. As shown in Fig. 10, the choice of k significantly impacts the result of rendered segmentation. Since different sequences contain varying numbers of objects, we believe that employing a method with an adaptive number of clusters, such as DBSCAN, validates our design choices with additional flexibility and no loss on the segmentation accuracy.

8.5. Time and Storage Analysis

We performed a comprehensive time analysis of SADG using an RTX 3080 GPU and an Intel i7-12700 CPU on the NeRF-DS [52] dataset. To evaluate its performance, we compared it with DGD [25], the only publicly available model addressing segmentation in dynamic novel views. The results are summarized in Tab. 6.

Most existing approaches for novel view segmentation [6, 16, 57] require precomputing SAM masks or object masks prior to training. In contrast, DGD dynamically computes DINOv2 feature maps for supervision during training. For a fair comparison, we precomputed the DINOv2 feature maps for all training images in DGD and stored them beforehand on an SSD, alongside our precomputed SAM

Method	HyperNeRF-Mask		Neu3D-Mask		Immersive-Mask		Technicolor-Mask	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
SA4D [16]	0.7767	0.9633	0.8832*	0.9391*	0.7987*	0.9835*	0.8271*	0.9734*
SADG (Ours)	0.8663	0.9845	0.9022	0.9945	0.9234	0.9945	0.9308	0.9917

Table 5. Average quantitative results for SA4D [16] and SADG on our *HyperNeRF-Mask*, *Neu3D-Mask*, *Immersive-Mask*, and *Technicolor-Mask* segmentation benchmark. SADG outperforms the baseline on average mIoU and mAcc. *: Similarly to Gaussian Grouping [57] SA4D needs the precomputed object IDs from DEVA, and for the multi-view sequences, we train their semantic module on the camera closest to the test view.

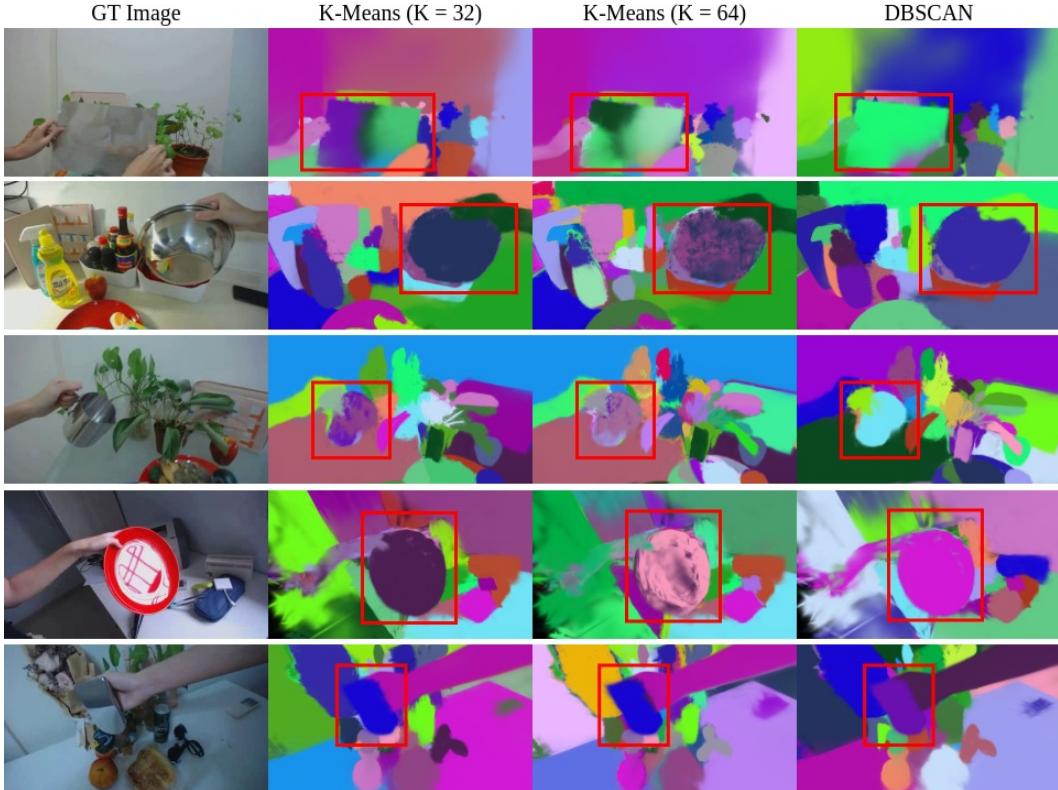


Figure 10. Comparison of rendered segmentation for K-Means ($K=32$), K-Means ($K=64$), and DBSCAN. K-Means-based segmentation results exhibit noise in the specular regions. Moreover, they require a pre-defined number of clusters, which is hard to provide for casual realistic videos. DBSCAN alleviates all of the above issues.

masks.

As shown in Tab. 6, DGD [25] spends little preprocessing time for DINOv2 features (approximately 5 seconds for all training images per sequence). Although our SADG requires preprocessing time to generate SAM masks, this process only needs to be done once before training. DGD assigns a 384-dimensional semantic feature to each Gaussian. Rendering these high-dimensional features into 2D is computationally inefficient, significantly slowing down the training process. SADG uses 32-dimensional Gaussian features, which can be rendered much more efficiently. As a result, SADG achieves over $3\times$ faster total time (preprocessing + training) than DGD while only requiring 4.5× less space in the storage.

In addition, we evaluated the processing time for click prompts in our GUI. For smaller scenes with less than 200k Gaussians such as *as_novel_view*, segmenting an object via a click prompt takes approximately 0.05 seconds. For larger scenes with more than 1,000k Gaussians captured by a multi-view camera setup, such as *O1_Welder*, the processing time per click prompt is around 0.4 seconds.

9. Comparison Against Segment Any 4D Gaussians Model

We provide additional quantitative and qualitative comparisons with the concurrent work on 4D segmentation and editing, SA4D [16]. The pipeline consists of three

	Avg. Preprocessing Time (min)	Avg. Training Time (min)	Total Time (min)		Avg. Size of Gaussians (MB)
DGD [25]	0.08	103	103		285
SADG (Ours)	16	16	32		60

Table 6. Comparison of the average size (MB) of the saved Gaussians of each scene and average total times (minute) for preprocessing and training of the semantic branch of DGD [25] and SADG on NeRF-DS [52] dataset (on average, 646 training images per scene). For both methods, we freeze the pre-trained Gaussians of the dynamic geometry reconstruction and train their semantic branches for an additional 10000 iterations. SADG demonstrated over $3\times$ faster training time compared to DGD while only requiring $4.5\times$ less memory in storage due to its compact Gaussian features.

stages: (1) dynamic 3D reconstruction based on 4D-GS model [50], (2) identity feature field learning, and (3) post-processing with outlier removal. Since the authors have not released the official code, we use the unclean version from <https://github.com/guanjunwu/sa4d> for training and testing the model on our benchmark. We do not provide the performance on *NeRF-DS-Mask* with specular and transparent moving objects in Tab. 5 since 4D-GS fails to converge on most of the sequences apart from two (*basin_novel_view*, *press_novel_view*). Similarly, we exclude two sequences from the HyperNeRF-Mask dataset, such as *hand1-dense-v2* and *espresso* due to 4D reconstruction failure.

In quantitative assessment, we select the top 30 segmentation results to more closely reflect the test set used in the original SA4D benchmark (not released). We also uniformly sub-sample point cloud (100K points) from SfM [45]. As seen in Tab. 5, our SADG significantly outperforms SA4D on all datasets. From the qualitative results in Fig. 12 and Fig. 13, we notice that rendered masks are larger than the underlying object and include a lot of unrelated regions even after the outlier removal in the post-processing step, which negatively affects objects' segmentation accuracy. We further demonstrate some editing comparisons in Fig. 17, Fig. 16, and Fig. 15. Due to inaccurate segmentation, composition, and removal, SA4D results contain many spurious regions, degrading the editing quality.

10. Quantitative Results per Sequence

For completeness to the average numbers per datasets (Tab. 1), we report the performance of our method individually on all benchmarks in Tab. 7, Tab. 8, Tab. 9, Tab. 10, and Tab. 11.

11. Segmentation Benchmarks

As discussed in Sec. 4.2, we employ SAM2 [43] to manually select objects of interest to form our segmentation benchmarks for dynamic scene novel view synthesis. The examples are illustrated in Fig. 11.

12. Additional Qualitative Results

We provide additional qualitative results in Fig. 12 and Fig. 13 to further showcase the effectiveness of SADG over other methods. While SAGA [7] suffers from degenerate Gaussians, our reconstruction backbone and effective post-processing mitigate the issue. At the same time, rendered masks from other methods, such as DGD [25] and SA4D [16], are inaccurate and introduce spurious regions. Our method demonstrates the best performance among all approaches.

13. Further Scene Editing Examples

We present further scene composite examples of SADG in Fig. 14. We also compare the quality of object removal and scene composition with SA4D [16] in Fig. 16 and Fig. 17. Overall, as discussed in Sec. 9, SADG outperforms SA4D both quantitatively and qualitatively.

14. Visualization of the Rendered View, Rendered Segmentation, Segmented Object

We also provide the visualizations of the rendered novel view, the rendered segmentation, and the segmented objects in Fig. 18, Fig. 19, and Fig. 20. The rendered segmentation of SADG is based on Gaussian clustering, which is performed in 3D space, resulting in consistent segmentation in the dynamic scene.

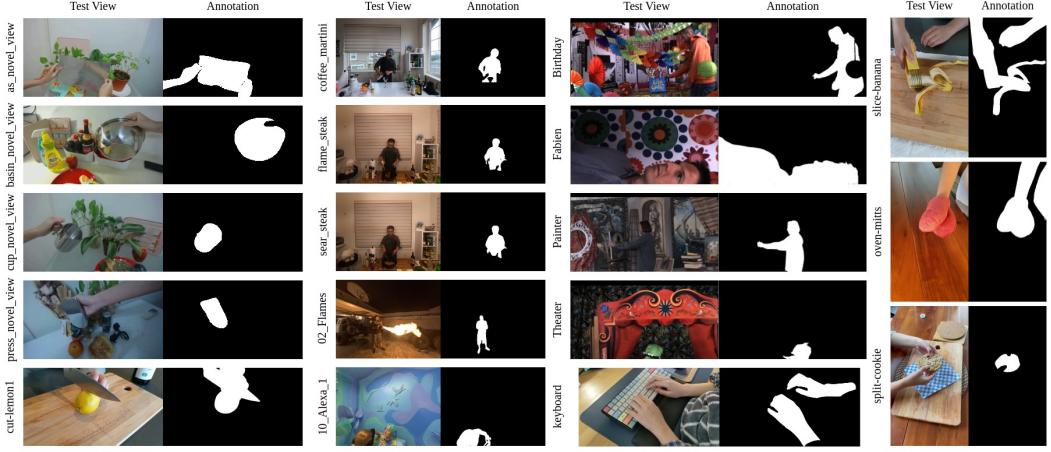


Figure 11. Image-object mask examples from our proposed extension to existing NVS datasets for dynamic 4D segmentation and downstream editing tasks. We ensure multi-view and temporal consistencies of the masks.

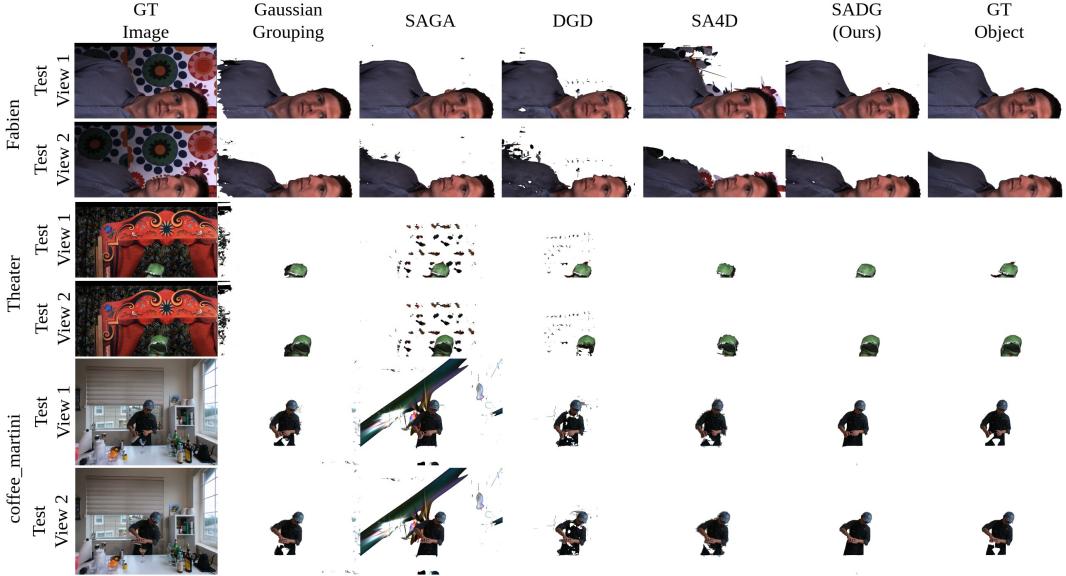


Figure 12. Additional qualitative results on *Immersive-Mask* and *Neu3D-Mask* datasets. Our method delivers the best temporally-consistent performance across all baselines.

Method	as_novel_view		basin_novel_view		cup_novel_view		press_novel_view		plate_novel_view		Average	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
Gaussian Grouping [57]	0.8140	0.9487	0.8083	0.9612	0.9020	0.9936	0.7803	0.9912	0.8710	0.9696	0.8351	0.9729
SAGA [6]	0.8235	0.9542	0.6437	0.9140	0.8833	0.9922	0.8119	0.9918	0.8736	0.9699	0.8072	0.9644
DGD [25]	0.7135	0.9266	0.6685	0.9291	0.694	0.9752	0.6457	0.9822	0.8406	0.9625	0.7125	0.9551
SADG (Ours)	0.9162	0.9799	0.8740	0.9785	0.8864	0.9924	0.8098	0.9916	0.8733	0.9707	0.8719	0.9826

Table 7. Quantitative results for Gaussian Grouping [57], SAGA [6], DGD [25] and SADG on our *NeRF-DS-Mask* segmentation benchmark. SADG outperforms the baselines on average.

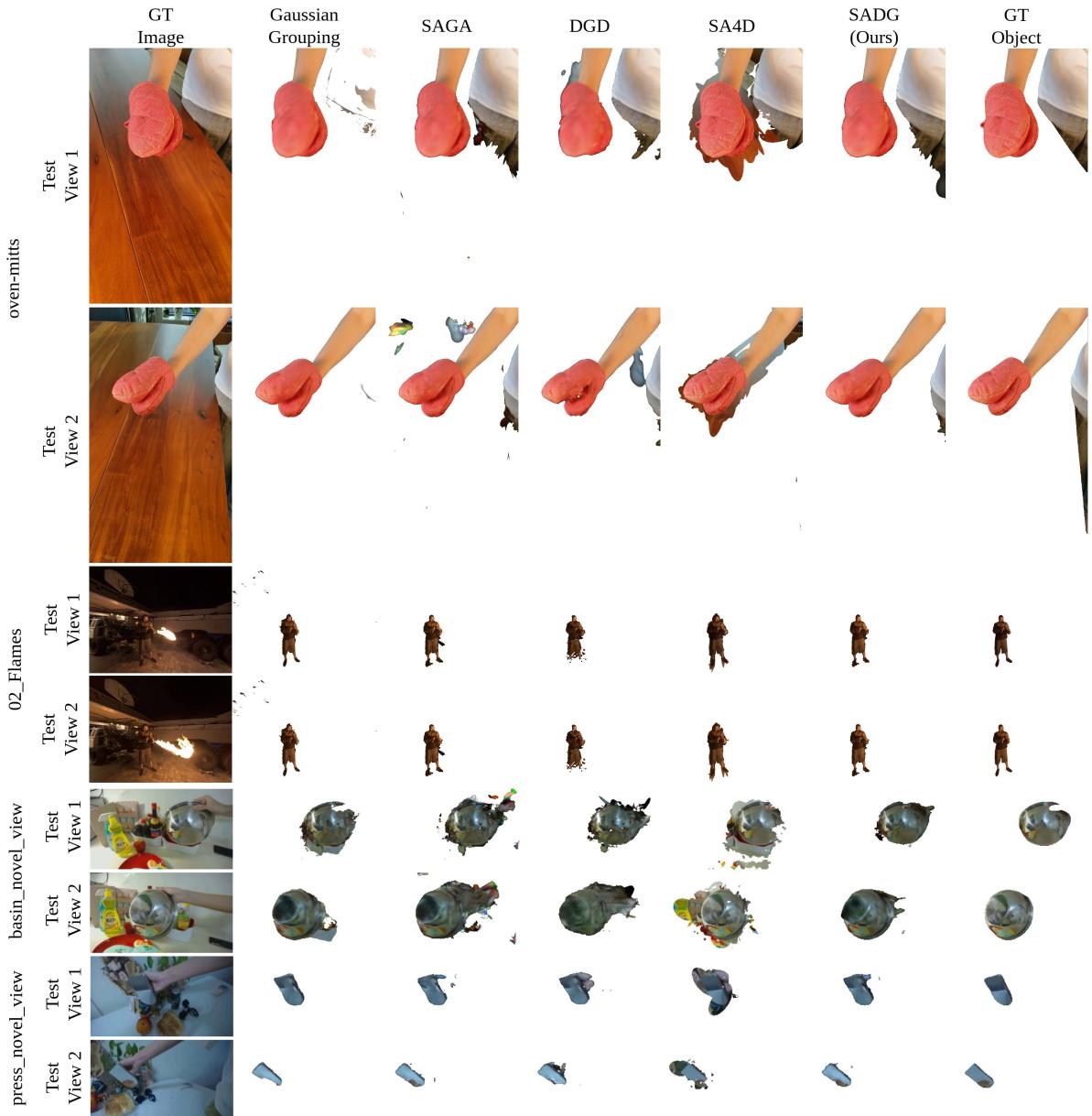


Figure 13. Additional qualitative results on *HyperNeRF-Mask*, *Immersive-Mask*, and *NeRF-DS-Mask* datasets. Our method delivers the best temporally-consistent performance across all baselines.

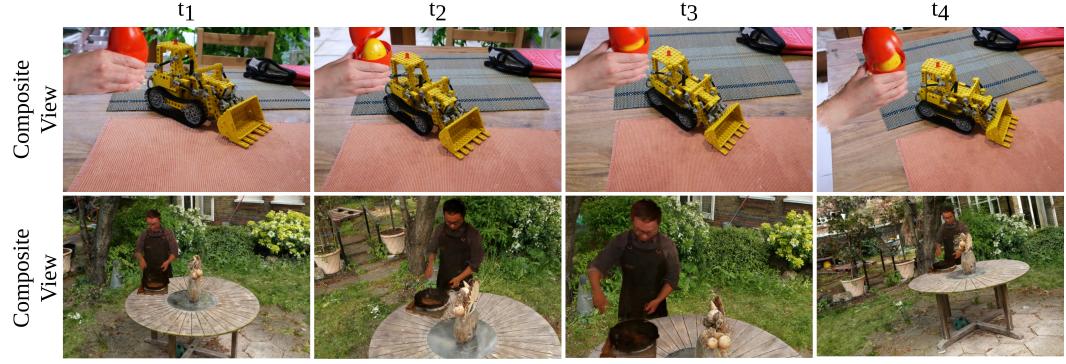


Figure 14. Scene composition with SADG. Top: *chickchicken* and Mip-NeRF 360 [1] kitchen. Bottom: *sear_steak* and Mip-NeRF 360 [1] garden. The scale and position of the object are adapted manually to fit the scene.

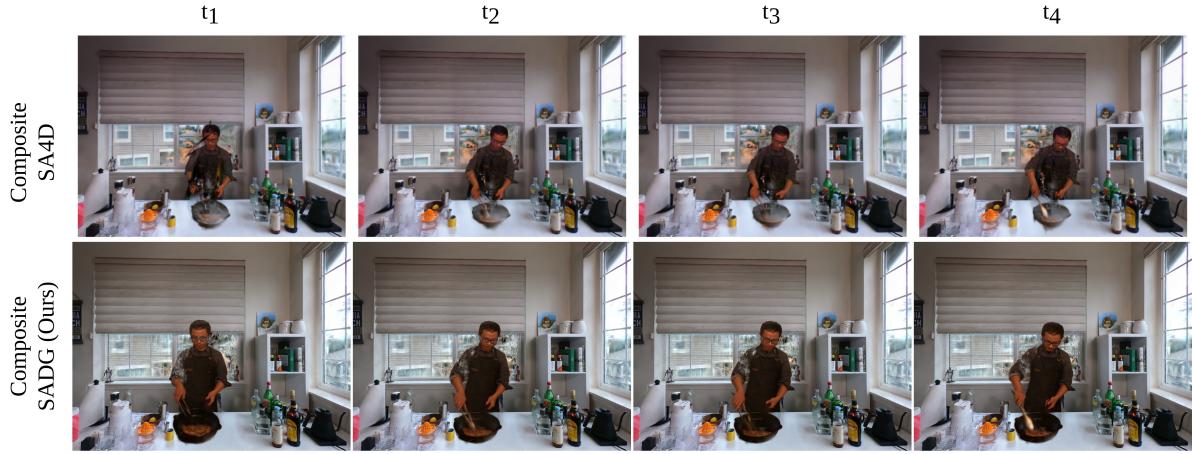


Figure 15. Scene composition with SA4D [16] and our SADG on Neu3D sequences. We segment the person from *sear_steak* and replace him in *coffee_martini* with it. Our method delivers fewer artifacts. The scale and position of the object are adapted manually to fit the scene.

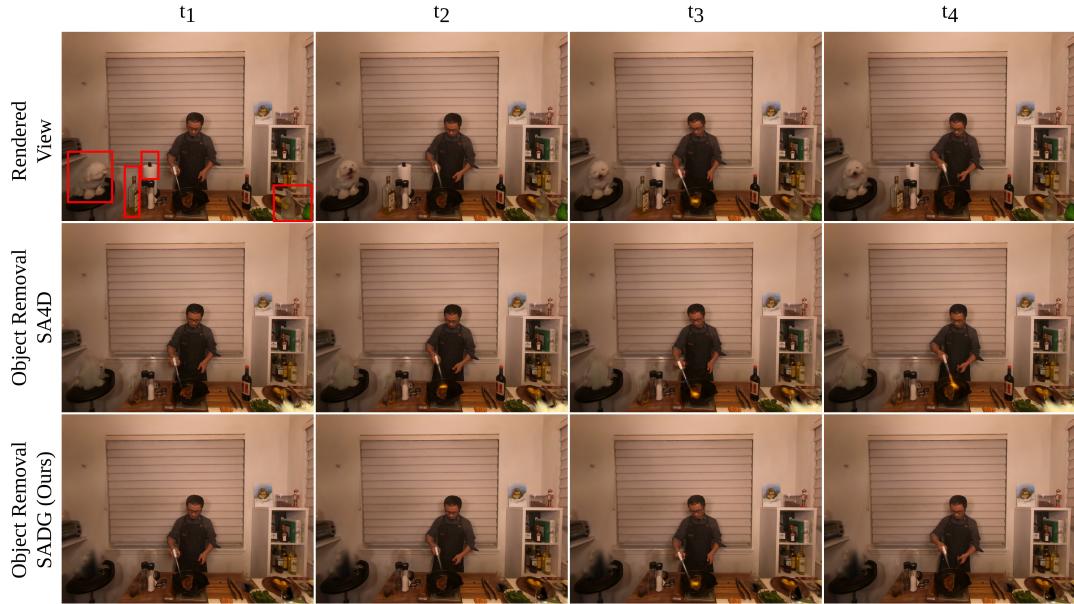


Figure 16. Object removal (identified with the red boxes) with SA4D [16] and our SADG in Neu3D *flame_steak* sequence. Our method delivers fewer artifacts and does not introduce ghostly effects in place of the removed object.

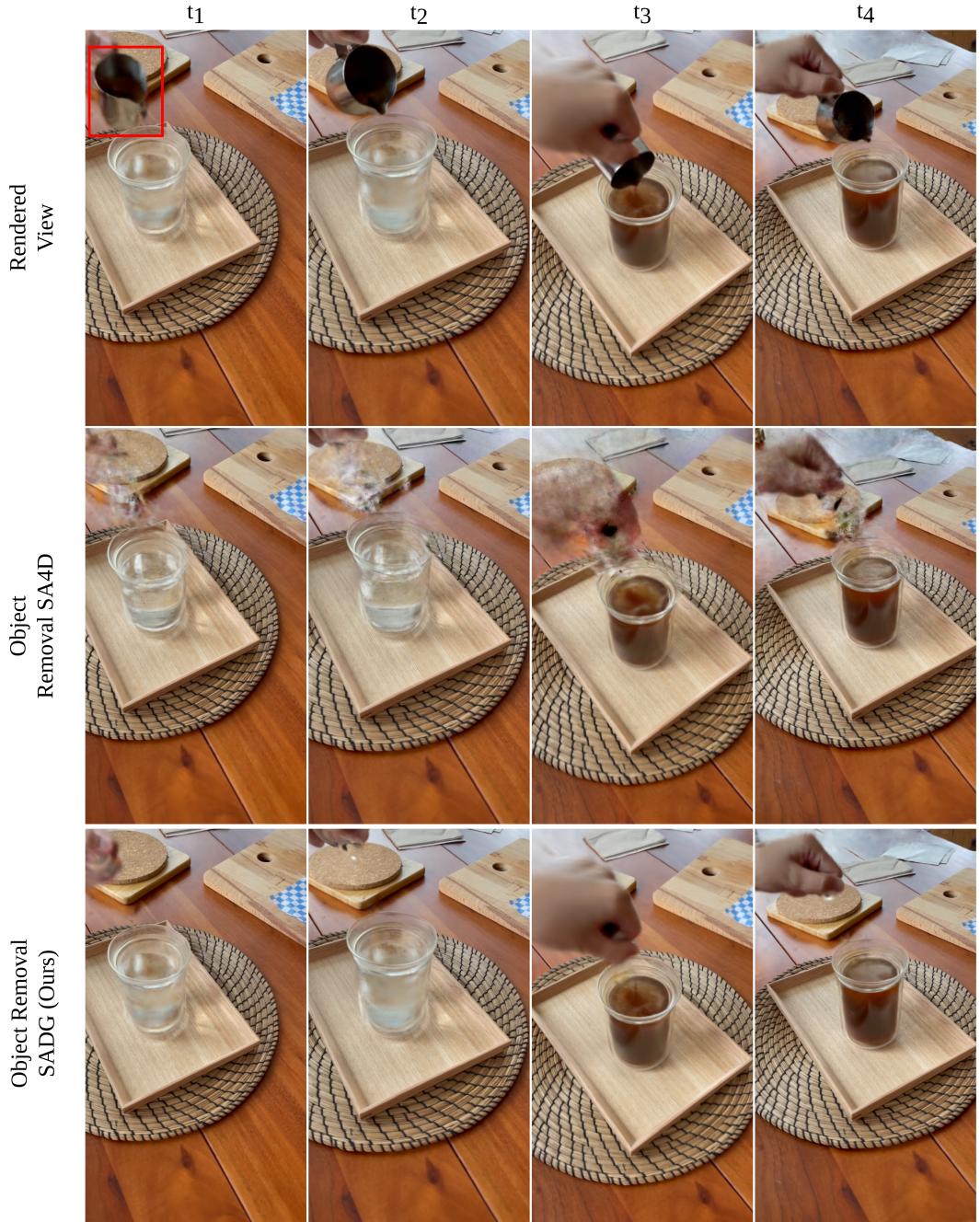


Figure 17. Object removal (saucer in the red box) with SA4D [16] and our SADG in HyperNeRF *americano* sequence. Our method delivers fewer artifacts and does not introduce ghostly effects in place of the removed object. Our method does not require additional inpainting as the background is complete due to multi-view observations.

Method	americano		chickchicken		cut-lemon1		espresso		hand		keyboard	
	mIoU	mAcc										
Gaussian Grouping [57]	0.8980	0.9969	0.9436	0.9867	0.7153	0.9394	0.4968	0.9642	0.8865	0.9814	0.9159	0.9861
SAGA [6]	0.7325	0.9873	0.8633	0.9648	0.5263	0.8398	0.6395	0.9798	0.9088	0.9859	0.8070	0.9645
DGD [25]	0.7748	0.9912	0.8317	0.9589	0.8583	0.9718	0.7240	0.9871	0.8936	0.9838	0.8451	0.9733
SADG (Ours)	0.8144	0.9922	0.9308	0.9835	0.8795	0.9769	0.7164	0.986	0.9006	0.9849	0.8952	0.9827
oven-mitts slice-banana split-cookie torchocolate Average												
Gaussian Grouping [57]	0.7420	0.9505	0.9095	0.9721	0.9134	0.9960	0.9196	0.9981	0.8341	0.9771		
SAGA [6]	0.8667	0.9766	0.7323	0.8960	0.8186	0.9906	0.7652	0.9933	0.7660	0.9579		
DGD [25]	0.8595	0.9755	0.8408	0.947	0.8467	0.9925	0.8223	0.9954	0.8297	0.9777		
SADG (Ours)	0.9295	0.9876	0.8782	0.9617	0.8581	0.9930	0.8599	0.9964	0.8663	0.9845		

Table 8. Quantitative results for Gaussian Grouping [57], SAGA [6], DGD [25] and SADG on our *HyperNeRF-Mask* segmentation benchmark. SADG outperforms the baselines on average.

Method	coffee_martini		cook_spinach		cut_roasted_beef		flame_steak		sear_steak		Average	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
Gaussian Grouping [57]	0.8295	0.9890	0.8974	0.9941	0.9512	0.9972	0.8279	0.9900	0.9261	0.9959	0.8864	0.9932
SAGA [6]	0.2201	0.8081	0.8125	0.9881	0.6982	0.9767	0.8439	0.9911	0.8959	0.9941	0.6941	0.9516
DGD [25]	0.7875	0.9865	0.815	0.9883	0.817	0.9877	0.6771	0.9776	0.7638	0.9854	0.7721	0.9851
SADG (Ours)	0.9120	0.9948	0.9129	0.9951	0.9103	0.9947	0.8716	0.9930	0.9044	0.9947	0.9022	0.9945

Table 9. Quantitative results for Gaussian Grouping [57], SAGA [6], DGD [25] and SADG on our *Neu3D-Mask* segmentation benchmark. SADG remains on par with Gaussian Grouping on average and outperforms SAGA by a large margin on average mIoU.

Method	01_Welder		02_Flames		10_Alexa_1		11_Alexa_2		Average	
	mIoU	mAcc								
Gaussian Grouping [57]	0.7920	0.9736	0.8364	0.9936	0.7391	0.9840	0.7018	0.9590	0.7673	0.9776
SAGA [6]	0.8062	0.9737	0.8463	0.9940	0.4794	0.9513	0.8260	0.9797	0.7395	0.9747
DGD [25]	0.873	0.9854	0.7775	0.9914	0.8107	0.9899	0.7312	0.9734	0.7981	0.9850
SADG (Ours)	0.8975	0.9877	0.8808	0.9955	0.9367	0.9970	0.9786	0.9979	0.9234	0.9945

Table 10. Quantitative results for Gaussian Grouping [57], SAGA [6], DGD [25] and SADG on our *Immersive-Mask* segmentation benchmark. SADG outperforms the baselines on average mIoU by a large margin.

Method	Birthday		Fabien		Painter		Theater		Average	
	mIoU	mAcc								
Gaussian Grouping [57]	0.9324	0.9908	0.9406	0.9758	0.8759	0.9888	0.4428	0.9646	0.7979	0.9800
SAGA [6]	0.8541	0.9779	0.9371	0.9943	0.9670	0.9867	0.4071	0.9579	0.7913	0.9792
DGD [25]	0.7203	0.9573	0.9177	0.9667	0.7959	0.9788	0.6826	0.9885	0.7791	0.9728
SADG (Ours)	0.8839	0.9833	0.9734	0.9889	0.9719	0.9976	0.8941	0.9969	0.9308	0.9917

Table 11. Quantitative results for Gaussian Grouping [57], SAGA [6], DGD [25], and SADG on our *Technicolor-Mask* segmentation benchmark. SADG outperforms the baselines on average mIoU by a large margin.

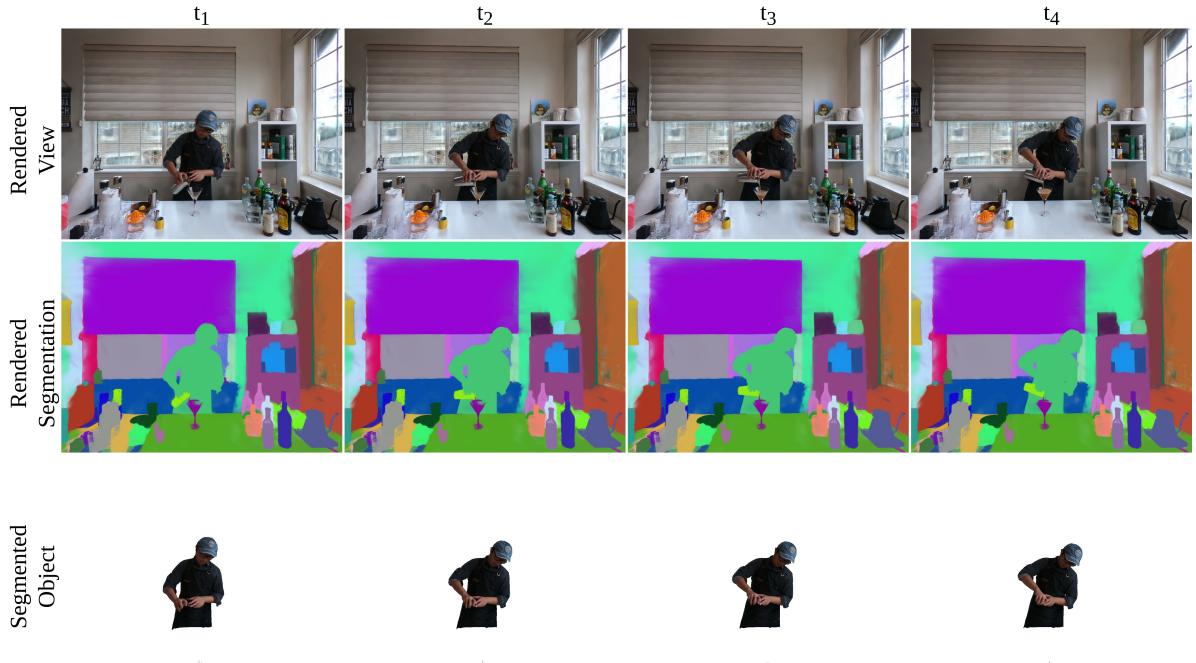


Figure 18. Visualization of rendered view, rendered segmentation, and segmented objects of the scene *coffee_martini* in four different timestamps in the novel views. The consistency of the semantic feature field across time opens out-of-box editing opportunities for dynamic scenes.

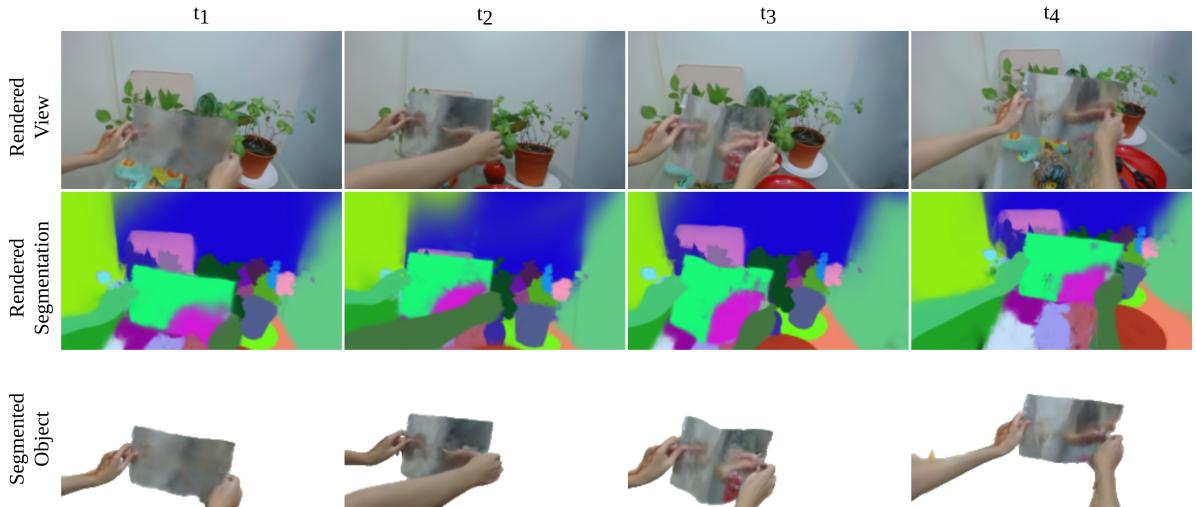


Figure 19. Visualization of rendered view, rendered segmentation, and segmented objects of the scene *as_novel_view* in four different timestamps in the novel views. The reflection in the foil remains temporally consistent, which enables post-processing editing effects.

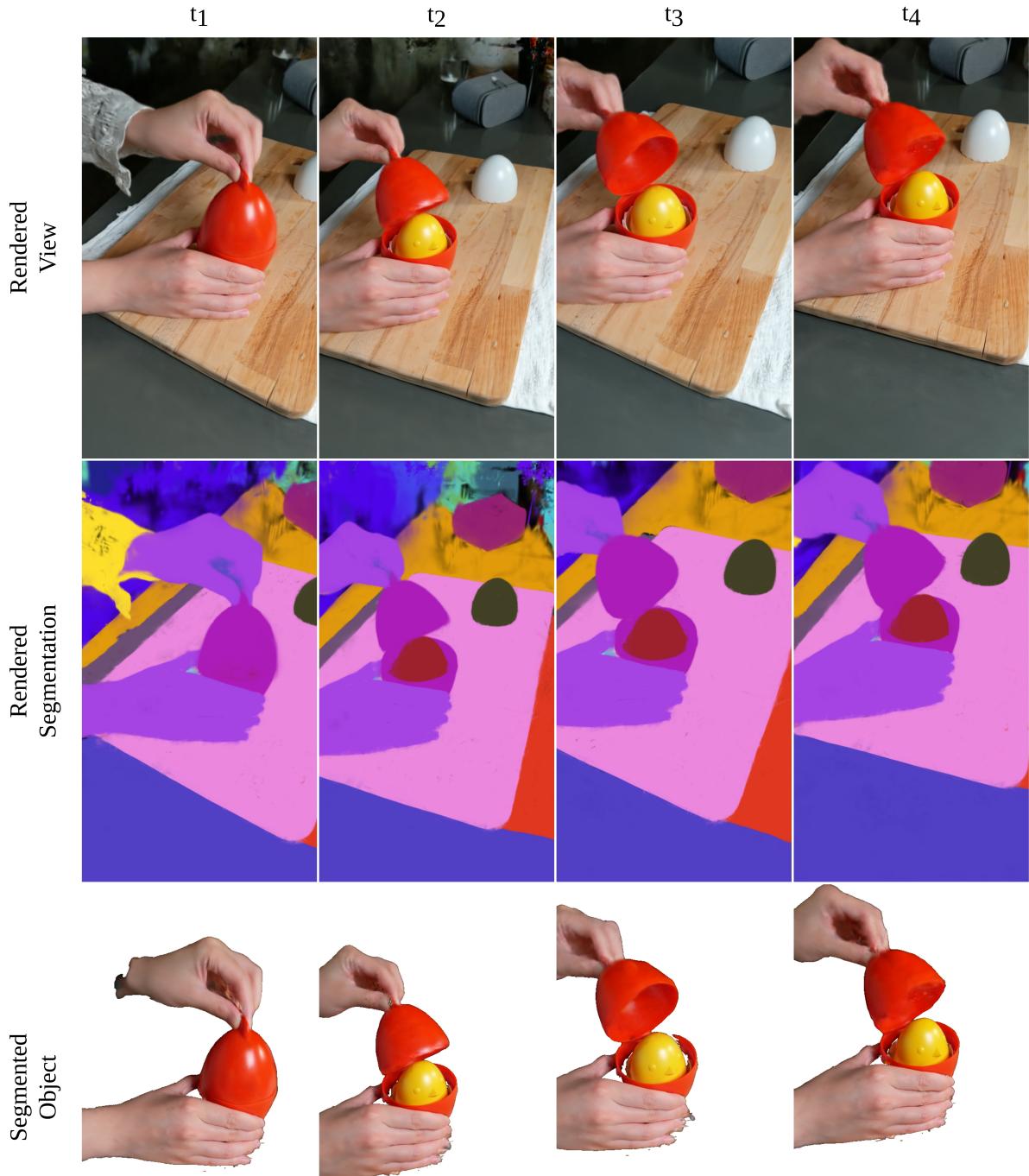


Figure 20. Visualization of rendered view, rendered segmentation, and segmented objects of the scene *chickchicken* in four different timestamps in the novel views. Despite severe occlusions, semantic segmentation is consistent over time.