TIBR4D: Tracing-Guided Iterative Boundary Refinement for Efficient 4D Gaussian Segmentation

He Wu
Zhejiang University of Technology
Hangzhou, Zhejiang, China

Xia Yan
Zhejiang University of Technology
Hangzhou, Zhejiang, China

Yanghui Xu
Zhejiang University of Technology
Hangzhou, Zhejiang, China

Liegang Xia
Zhejiang University of Technology
Hangzhou, Zhejiang, China

Jiazhou Chen✉
Zhejiang University of Technology,
Zhejiang Key Laboratory of Visual
Information Intelligent Processing
Hangzhou, Zhejiang, China
cjz@zjut.edu.cn

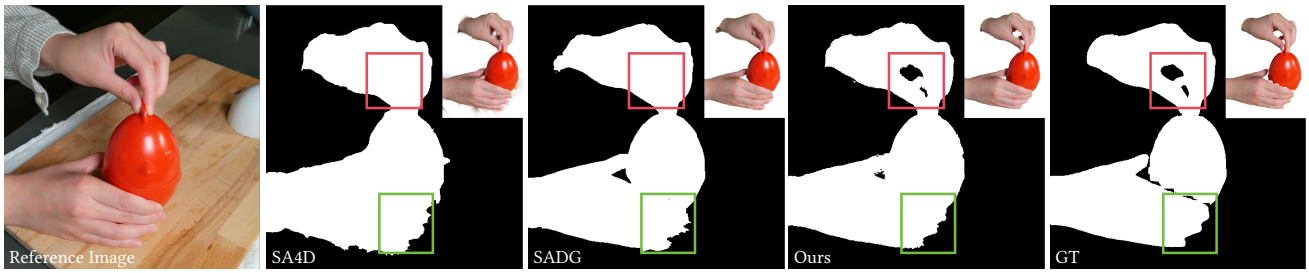

Figure 1: Comparison with prior works SA4D [10] and SADG [21], which suffer from inaccuracy at object boundaries. Benefiting from two training-free iterative refinement stages, our method enables efficient extraction of target objects while effectively suppressing floating Gaussians and boundary leakage, resulting in more accurate segmentation.

Abstract

Object-level segmentation in dynamic 4D Gaussian scenes remains challenging due to complex motion, occlusions, and ambiguous boundaries. In this paper, we present an efficient learning-free 4D Gaussian segmentation framework that lifts video segmentation masks to 4D spaces, whose core is a two-stage iterative boundary refinement. The first stage is Iterative Gaussian Instance Tracing (IGIT) at the temporal segment level. It progressively refines Gaussian-to-instance probabilities through iterative tracing and extracts corresponding Gaussian point clouds. Thus, it can handle occlusions and preserve the completeness of object structures. The second stage is frame-wise Gaussian Rendering Range Control (RRC), which suppresses highly uncertain Gaussians near object boundaries while retaining their core contributions for more accurate boundaries. Furthermore, a temporal segmentation strategy is proposed for IGIT to balance identity consistency and dynamic awareness. Correlations within each temporal segment enforce strong multi-frame constraints for stable identities, while independence across segments allows identity changes to be captured promptly. Experiments on HyperNeRF and Neu3D datasets demonstrate that our method produces clearer segmented Gaussian point clouds with accurate boundaries and achieves higher efficiency compared to SOTA methods. Source codes will be released soon after the paper acceptance.

CCS Concepts

• **Computing methodologies** → **Video segmentation**; *Rasterization.*

Keywords

4D Gaussian segmentation; Iterative Refinement; Instance Tracing

ACM Reference Format:
He Wu, Xia Yan, Yanghui Xu, Liegang Xia, and Jiazhou Chen. 2026. TIBR4D: Tracing-Guided Iterative Boundary Refinement for Efficient 4D Gaussian Segmentation. In *Proceedings of the 34th ACM International Conference on Multimedia (MM '26), November 10–14, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3767308.3835842

1 Introduction

This paper focuses on efficient and precise object-level segmentation in dynamic scenes. Real-world environments are inherently dynamic, with objects undergoing motion, deformation, and frequent interactions over time, making it particularly challenging to achieve efficient, reliable, and dynamic-aware object segmentation. On the other hand, such a capability is relevant to a wide

MM '26, Rio de Janeiro, Brazil.

ACM ISBN 979-8-4007-2213-4/2026/11
https://doi.org/10.1145/3767308.3835842

range of applications, including autonomous driving, robotics, and augmented or virtual reality.

The rapid advancement of novel view synthesis techniques [1, 12, 25, 40] has significantly improved scene modeling capabilities. Neural Radiance Fields (NeRF) [25] demonstrates remarkable expressive power for static scenes, and have been extended to incorporate semantic information [5, 13, 14, 16, 17] and support dynamic scene representations [8, 19, 27–29]. However, NeRF-based approaches rely on implicit scene representations, which incur substantial computational overhead during training and rendering, limiting their applicability to downstream tasks such as object segmentation and scene editing. In contrast, 3D Gaussian Splatting (3DGS) [12] uses a learnable explicit representation, achieving faster rendering and improved interpretability. Some approaches incorporate language features into 3DGS to support open-vocabulary querying and semantic interaction [11, 20, 30, 33, 37], while some focus on instance-level segmentation, aiming to decompose complex scenes into semantically meaningful objects [4, 7, 24, 32, 39]. Meanwhile, Gaussian-based representations have been extended to the dynamic scene [22, 34, 36, 38]. These advances lay a strong foundation for exploring 4D Gaussian segmentation; however, it is still challenging to achieve efficient, reliable, and dynamics-aware object segmentation in 4D Gaussian scenarios.

Representative challenges in 4D Gaussian segmentation cover three key aspects, including: 1) achieving *accurate instance boundaries* under occlusions and complex motion, 2) balancing *dynamic scene expressiveness* with segmentation stability, and 3) maintaining practical *efficiency*. Existing 4D Gaussian segmentation methods generally follow two distinct paradigms: (i) tracking-based approaches that lift 2D video segmentation labels into the 4D space, producing time-varying Gaussian identity features, and (ii) tracking-free approaches that focus on learning consistent and stable Gaussian identity features via contrastive learning.

Tracking-based methods, represented by SA4D [10] and CIF [35], predict the identity features per-frame for each Gaussian using MLPs. These methods typically require shorter feature learning time and exhibit stronger responsiveness to dynamic appearance and structural changes. Nevertheless, frequent temporal updates of Gaussian identities often lead to instability during object extraction, manifested as noticeable Gaussian flickering artifacts. Additionally, due to the complex occlusion relationships in dynamic scenes, they commonly suffer from floating points and boundary leakage, which significantly degrade the quality of object-level 4D segmentation.

Tracking-free methods, such as SADG [21], learn fixed identity features for deformable Gaussians via contrastive learning, which alleviates visual artifacts caused by label inconsistency in tracking–based supervision. Nevertheless, maintaining fixed Gaussian identities over time limits its ability to handle dynamic scenes where Gaussian identities may change. Split4D [9] further introduces linear motion modeling for Gaussian primitives and adopts a streaming sampling strategy to enable Gaussian reuse, partially mitigating the limitations of fixed identities. However, such motion-based Gaussians may be less suitable for scene reconstruction in monocular datasets due to the lack of reliable motion cues. Moreover, boundary leakage of segmented objects is still not fully addressed in these methods, and contrastive learning–based methods generally incur substantial feature learning overhead.

In this paper, we propose a learning-free 4D Gaussian segmentation framework that efficiently lifts 2D video segmentation labels into the 4D space by iteratively tracing the probabilities of Gaussians belonging to objects. To address challenges arising from floating points and boundary leakage, our framework consists of two iterative refinement stages: Iterative Gaussian Instance Tracing (IGIT) on the temporal segment level and frame-wise Gaussian Rendering Range Control (RRC). IGIT removes most floating points by iteratively refining instance assignments, enabling better handling of occluded Gaussians and preserving more complete object structures. RRC performs fine-grained, frame-wise intra-Gaussian control over Gaussian rendering ranges, suppressing the rendering ranges of highly uncertain Gaussians near object boundaries, while preserving their core contributions around the Gaussian centers, resulting in more accurate segmentation boundaries. It is worth noting that IGIT is conducted on temporal segments instead of the whole video or frame-wise, to balance identity consistency and dynamic awareness. Correlations within each temporal segment enforce strong multi-frame constraints for stable identities, while independence across segments allows identity changes to be captured promptly. Thus, we introduce a temporal segmentation merging strategy that progressively merges temporal segments during the iterative tracing process. We further propose a method to compute the similarity between the Gaussian point clouds of the target object in adjacent temporal segments, which measures whether significant changes in Gaussian identities occur between temporal segments. This similarity is used to guide the merging process: temporal segments with high similarity are merged, whereas others remain separate, thereby automatically determining an appropriate temporal granularity. In summary, our contributions are:

- We present a learning-free object-level 4D Gaussian segmentation framework with tracing-guided iterative boundary refinement. Gaussians are iteratively traced and segmented at the temporal segment level.
- We propose a frame-wise Gaussian rendering range control strategy for fine-grained intra-Gaussian segmentation. This strategy allows iterative refinement for precise object boundaries.
- Experiments on HyperNeRF [28] and Neu3D [19] show that our method achieves more accurate rendered masks for dynamic Gaussian segmentation compared to existing approaches.

2 Related Work

4D Scene Representation: Existing 4D scene reconstruction methods often build upon established 3D reconstruction techniques and extend them to dynamic scenes. The introduction of Neural Radiance Fields (NeRFs) [25] and its extensions [1, 2] has significantly advanced 3D reconstruction, enabling full-scene modeling via implicit neural representations and photorealistic image synthesis. Numerous works have further extended NeRFs [25] to dynamic scenes [8, 19, 27–29]. However, such implicit representations often incur high computational costs and limit applicability to downstream tasks. In contrast, 3D Gaussian Splatting (3DGS) [12] and its extensions [23, 40, 41] adopt explicit, learnable representations, enabling efficient real-time rendering. Many methods extend 3DGS [12] to dynamic reconstruction [22, 34, 36, 38], through deformation and

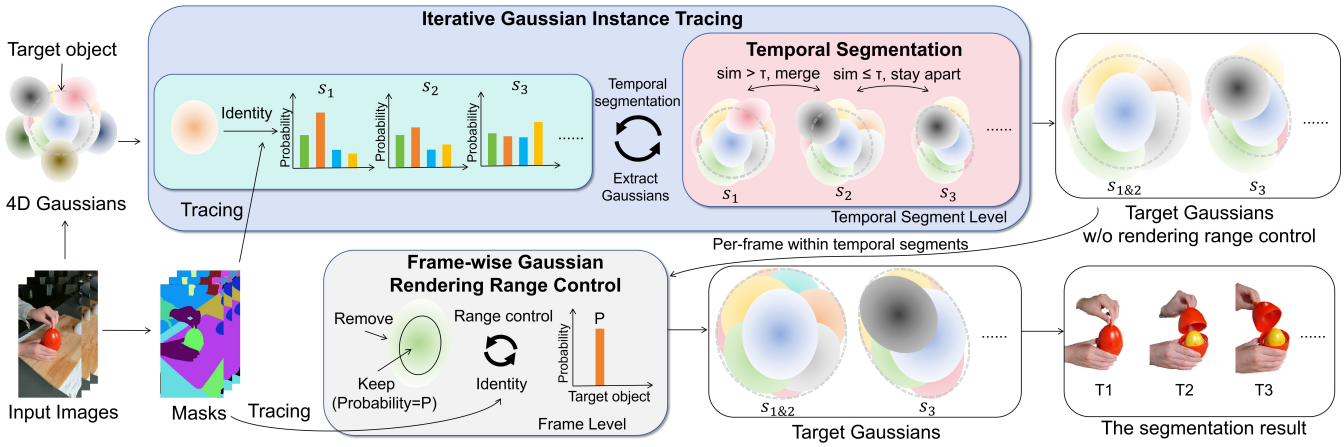

Figure 2: Overview. Our learning-free object segmentation framework of 4D Gaussian scenes consists of two stages: iterative Gaussian instance tracing (Sec. 3.1) with temporal segmentation (Sec. 3.2), and frame-wise Gaussian rendering range control (Sec. 3.3). In the first stage, we iteratively perform Gaussian Instance Tracing within temporal segments (denoted as $s_1, s_2, \ldots$) to remove erroneous Gaussians, and merge adjacent segments when their extracted Gaussians become consistent. In the second stage, we progressively truncate the outer regions of each projected 2D Gaussian and retain only its most reliable central contribution. After these two iterative stages, we obtain an accurate Gaussian point cloud for the target object.

trajectory modeling, facilitating downstream tasks such as dynamic scene understanding and interaction.

3D Segmentation: Recent advances in vision foundation models, such as SAM [15], DINO [3], DINOv2 [26], CLIP [31], and video trackers such as DEVA [6], have created new opportunities for semantic understanding in 3D and 4D scenes. Several NeRF-based approaches [5, 13, 14, 16, 17] lift semantic information from these models to the 3D scene level, enabling semantic-aware representations. With the emergence of 3D Gaussian Splatting (3DGS) [12], its explicit,learnable representation enables efficient rendering and improved interpretability, making it suitable for scene understanding. Consequently, many works integrate semantic information into Gaussian-based representations, some [11, 20, 30, 33, 37] focus on semantic segmentation by combining Gaussian representations with language features for open-vocabulary querying, while others target instance-level segmentation [4, 7, 24, 32, 39]. These instance-level approaches typically rely on 2D masks or video trackers for direct supervision, or employ contrastive learning to enforce cross-view feature consistency. Notably, Trace3D [32] introduces Gaussian Instance Tracing (GIT) to compute per-Gaussian weights, which are further used to guide adaptive density control. This design enables efficient lifting of 2D segmentation to 3D Gaussian scenes and improves boundary accuracy. However, most existing methods focus on static scenes and lack efficient mechanisms for robust instance segmentation in dynamic Gaussian scenes.

Dynamic Gaussian-based Instance Segmentation: In prior work on 4D Gaussian instance segmentation, DGD [18] extends the 3DGS [12] rasterization pipeline to render per-Gaussian semantic features supervised by DINOv2 [26] or CLIP [31]; however, the high dimensionality of these features (384/512) introduces non-negligible training overhead. SADG [21] adopts a contrastive learning strategy by assigning each Gaussian a set of learnable identity features that remain fixed over time, but struggles with identity changes. Split4D [9] addresses large-motion scenarios by endowing Gaussians with linear motion over time to improve temporal consistency; however, such motion-based Gaussians may be less suitable for monocular input, and boundary overflow remains challenging.

SA4D [10] and CIF [35] leverage a video tracker [6] to obtain temporally consistent instance labels and lift them into the 4D domain via feature fields. SA4D [10] employs an MLP to predict dynamically changing Gaussian features at each time step, but does not explicitly account for occlusions, and often results in noticeable artifacts or Gaussian flickering in the segmented objects. CIF [35] further estimates spatial occupancy and refines boundaries via Gaussian splitting, introducing higher learning complexity and increased instability in Gaussian identities.

In general, these methods rely on learned Gaussian features for 4D segmentation. Moreover, using these features for object segmentation requires empirically tuned thresholds to balance boundary leakage and object integrity. To address these challenges, we propose an efficient learning-free object-level 4D Gaussian segmentation framework with two tracing-based iterative refinement stages, achieving precise boundaries while preserving object integrity.

3 Methodology

Figure 2 shows the overall framework of the proposed 4D Gaussian segmentation method, which consists of two stages. In the first stage, Iterative Gaussian Instance Tracing (IGIT) is applied to obtain a relatively accurate Gaussian point cloud of target objects (Sec. 3.1). This process is performed over temporal segments, enabling identity changes of Gaussians to be captured promptly while maintaining multi-frame constraints. A temporal segmentation merging strategy is introduced to determine the appropriate temporal segmentation granularity (Sec. 3.2). In the second stage, the probability that each Gaussian belongs to the target object is leveraged as guidance for frame-wise Gaussian rendering range control. This process

is performed iteratively on each frame to progressively refine a reliable rendering range for each Gaussian distribution(Sec. 3.3).

3.1 Iterative Gaussian Instance Tracing (IGIT)

We adopt Gaussian Instance Tracing (GIT) [32] to lift 2D instance labels into 4D Gaussians and estimate the probability that each Gaussian belongs to different instances. Specifically, we first obtain 2D segmentation masks M_t for each frame t using DEVA [6]. For each training view v_t, we project Gaussians onto the image plane and compute a weight matrix based on the instance IDs of the covered pixels and each Gaussian's contribution to them. This matrix is denoted as $W_t \in \mathbb{R}^{N\times K}$, where N is the number of Gaussians and K is the number of unique instance IDs. The weight of a Gaussian G_i belonging to instance ID k (one of the K unique instance IDs) under view v_t is defined as:

$$W_t(G_i,k) = \sum_{(u,v)} \mathbb{I}\big[M_t(u,v)=k\big]\,\phi_{i,t}(u,v)\,T_i(u,v,t) \tag{1}$$

where $\phi_{i,t}(u,v) = o_i(t)\,G_i^{2D}(u,v,t)$ represents the contribution weight of the i-th Gaussian at pixel (u,v) and time t, $o_i(t)$ denotes the opacity of the i^{th} Gaussian at frame t, and $G_i^{2D}(u,v,t)$ represents the value of the 2D Gaussian distribution evaluated at pixel location (u,v,t). $T_i(u,v,t) = \prod_{j<i}\big(1-\phi_{j,t}(u,v)\big)$ represents the accumulated transmittance, $\mathbb{I}[\,\cdot\,]$ denotes the indicator function. $\mathbb{I}\big[M_t(u,v)=k\big]$ equals to 1 when $M_t(u,v)=k$ is true (i.e., the pixel (u,v,t) belongs to object k), otherwise it equals to 0. After summing the identity weight matrices of all Gaussians across all views and applying normalization, we obtain the probability matrix $P \in [0,1]^{N\times K}$, which is the probability of each Gaussian belonging to each instance identity over all time steps in the current scene:

$$P_{i,:} = \frac{\sum_t W_t(G_i,:)}{||\sum_t W_t(G_i,:)||} \tag{2}$$

For target instance k_0, we define a binary Gaussian mask $M_{k_0}^G \in \{0,1\}^N$ as $M_{k_0}^G = \mathbb{I}\left[k_0 = \arg\max_k P_{i,k}\right]$.

However, a single pass of GIT is insufficient to obtain an accurate Gaussian point cloud, as many Gaussians are partially or fully occluded in the original 4D scene. Although their visible regions are associated with the target object, occluded parts may become visible after segmentation, leading to changes in instance membership and resulting in incorrectly segmented floating Gaussians. Existing 4D Gaussian instance segmentation methods [9, 10, 21] typically compute identity features from the original scene, which are not well suited for the segmented point cloud where occlusion relationships have changed. Consequently, these methods rely on a single-step extraction with empirically tuned thresholds to suppress floating Gaussians. However, such thresholding is heuristic and lacks robustness, often requiring scene-specific tuning, and struggles to balance floating Gaussian removal with object completeness.

Benefiting from the aforementioned learning-free framework, we can efficiently perform iterative Gaussian identity estimation. Specifically, we re-project 2D instance labels onto the segmented Gaussians to update instance membership under new occlusion relationships, leading to refined identities:

$$W_t(G_i,k) = M_{k_0}^G \sum_{(u,v)} \mathbb{I}\big[M_t(u,v)=k\big]\,\phi_{i,t}(u,v)\,T_i(u,v,t;M_{k_0}^G) \tag{3}$$

where $T_i(u,v,t;M_{k_0}^G) = \prod_{j<i}\left(1-\phi_{j,t}(u,v)\,M_{k_0}^G\right)$. This formulation excludes Gaussians not belonging to the target object from subsequent identity and occlusion computations. By iterating this process, we progressively obtain a more accurate Gaussian point cloud.

In our experiments, each IGIT iteration corresponds to a full pass of instance tracing over all training views. This process is empirically observed to converge: in later iterations, the extracted Gaussian point cloud changes very marginally. Accordingly, the variation in the average number of remaining Gaussians per frame is continuously monitored across iterations. If the decrease in this quantity becomes negligible, IGIT is considered to have effectively converged, and the iteration is then terminated. In practice, let $\epsilon^{(l)}$ denote the reduction rate in the average per-frame Gaussian count at iteration l, and we terminate the iteration when $\epsilon^{(l)} < 0.005$ for two consecutive iterations. For most scenes, this typically occurs within 5–10 iterations. As an example, the evolution of ϵ in the *americano* is illustrated by the orange line in Figure 4.

3.2 Temporal Segmentation (TS)

In 4D Gaussian scenes, Gaussian identities may vary over time. Taking the *cut-lemon1* example in Figure 3, Gaussians at the cut part shrink, while those at the new location deform into the lemon slice. In such a case, methods with fixed Gaussian identities, such as SADG [21], tend to produce artifacts and content loss. Instead, approaches like [10, 35] predict Gaussian identities dynamically at each time step, and thus can capture identity variations. Nevertheless, per-frame prediction may introduce flickering or leave Gaussians unhandled, due to insufficient multi-view supervision.

To avoid the constancy of global-fixed identities and the instability of per-frame identity prediction, IGIT operates at the temporal segment level. Correlations within each temporal segment enforce strong multi-frame constraints for stable identities, while independence across segments allows identity changes to be captured promptly. An appropriate temporal segmentation strategy is therefore needed to determine at which time points the dynamic scene should be partitioned into temporal segments.

Assume that the sequence of a scene is partitioned into S temporal segments in an iteration, indexed by $s \in \{1,\dots,S\}$. For two adjacent segments s and $s+1$, let $M_{k_0}^{G,s}$ and $M_{k_0}^{G,s+1}$ denote the Gaussians extracted by IGIT (Sec. 3.1). Their Gaussians can be divided into two categories: (i) points that exist in both temporal segments, i.e., $M_{k_0}^{G,s} \cap M_{k_0}^{G,s+1}$, and (ii) points that exist only in one of them. The underlying assumption is that two adjacent temporal segments should be merged only when the first category dominates the rendering contributions in each segment. In this case, the target object is considered to have no significant change across the two temporal segments, and merging them allows IGIT to leverage a larger set of views, leading to more accurate segmentation results. Conversely, if the Gaussians that exist only in one temporal segment contribute significantly to its rendering, this indicates notable changes in the target object. Merging in such cases may hinder correct segmentation of these high-contribution Gaussians; therefore, the temporal segments should stay apart, and IGIT will be applied independently.

Based on the above intuition, we define the ratios $\alpha_{s,s+1}$ and $\alpha_{s+1,s}$ to measure the proportion of rendering contributions of the first cagetory in each segment, and the similarity between two

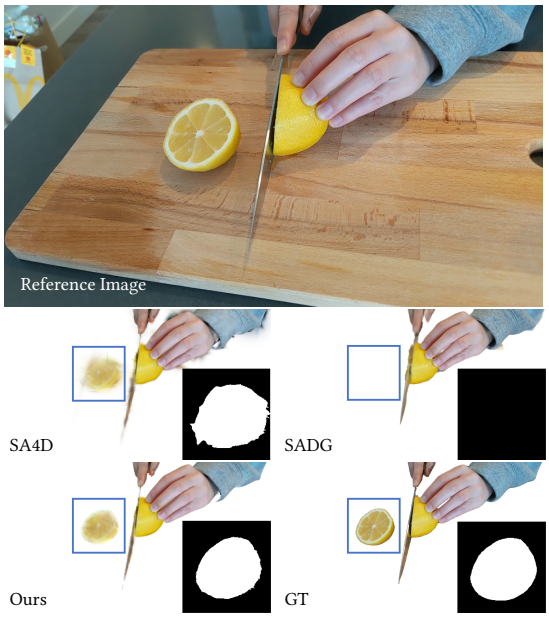

Figure 3: Comparison with SA4D [10] and SADG [21] in the *cut-lemon1* scene. SA4D and our method, which support dynamic Gaussian identities, successfully capture the lemon slice after it is cut off, while SADG fails to reflect this change due to its fixed identities. Compared to SA4D, our temporal segmentation strategy further introduces additional multi-view supervision, reducing incorrectly retained Gaussians.

adjacent segments as well:

$$\mathrm{sim}(s, s+1) = \min\left(\alpha_{s,s+1},\ \alpha_{s+1,s}\right) \tag{4}$$

$$\alpha_{s_1,s_2} = \frac{\sum_{i \in \mathcal{M}_{k_0}^{G,s_1} \cap \mathcal{M}_{k_0}^{G,s_2}} W_{s_1}}{\sum_{i \in \mathcal{M}_{k_0}^{G,s_1}} W_{s_1}} \tag{5}$$

Here, $W_s \in \mathbb{R}^N$ denotes the aggregated rendering contributions of Gaussians in temporal segment s, where the contribution of each Gaussian G_i is computed as $W_s(G_i) = \sum_{t \in s} \sum_{k \in K} W_t(G_i, k)$. A threshold τ is introduced so that the s and $s+1$ are merged when $\mathrm{sim}(s, s+1) > \tau$; otherwise, they stay apart.

The formation of temporal segments and the determination of τ are performed jointly with IGIT iterations. The scene is first partitioned into temporal segments based on each training view frame. After the first IGIT iteration, the average similarities $sim_{avg} = \sum sim(s, s+1)/(S-1)$ between each pair of adjacent frames is computed. The merging threshold τ is then empirically set as $\tau = sim_{avg} - 0.1$. After each IGIT iteration, the similarities between all adjacent temporal segments are evaluated: adjacent segments with similarity exceeding τ are merged, while others stay apart. The updated temporal segments are then used to guide the next IGIT iteration. After several iterations, the temporal segmentation stabilizes. In scenes where Gaussian identities remain relatively consistent, temporal merging typically results in a single segment. In contrast, in scenes with complex variations in Gaussian identities, such as *cut-lemon1*, multiple temporal segments are preserved.

3.3 Frame-wise Gaussian Rendering Range Control (RRC)

After IGIT and TS, we obtain relatively accurate Gaussian point clouds for each temporal segment. However, two types of Gaussians still degrade mask rendering quality. The first type consists of Gaussians near object boundaries or occupying excessive screen space. Although assigned high probabilities of belonging to the target object within a temporal segment, much of their distribution lies outside the true object boundaries. The second type consists of Gaussians with unstable identities within a temporal segment. While the temporal segmentation strategy encourages identity consistency within each temporal segment, it cannot effectively handle individual Gaussians whose identities change within a segment. Naively removing these two types of Gaussians per frame is undesirable, as it may produce overly sparse point clouds and exacerbate flickering, ultimately lowering rendering quality.

Therefore, we propose the frame-wise Gaussian Rendering Range Control (RRC), which retains only the most central and informative portion of each Gaussian's distribution.This strategy aims to produce accurate segmentation boundaries while avoiding excessive sparsity in the Gaussian point cloud, compared to naively removing highly uncertain Gaussians. Similar to the procedure in Sec. 3.1, we project the 2D segmentation labels onto the Gaussians for each training frame, generating the probability p_{i,t,k_0} that Gaussian G_i belongs to the object k_0 at time t. We use p_{i,t,k_0} to initialize the threshold r_{i,t,k_0}, which is then used to control the Gaussian rendering range in both instance tracing and rendering. For each projected Gaussian, we render only the central region whose cumulative probability mass equals r_{i,t,k_0}, while the outer region is excluded from both rendering and occlusion computations.

Unlike IGIT, occlusion effects from preceding Gaussians ($T_i(u, v, t)$) are ignored during instance tracing at this stage, as visibility has been accounted for in IGIT, where Gaussians incorrectly retained due to occlusion have been largely removed. More importantly, the rendering range is controlled independently for each projected 2D Gaussian. By excluding occlusion modeling, the probability p_{i,t,k_0} becomes more stable, facilitating more reliable estimation of the threshold r_{i,t,k_0} and avoiding overly restrictive truncation:

$$W_t(G_i, k) = M_{k_0}^{G,s} \sum_{(u,v)} \mathbb{I}\left[M_t(u, v) = k\right] \phi_{i,t}(u, v)\, \psi_{i,t}(u, v) \tag{6}$$

$$p_{i,t,k_0} = \frac{W_t(G_i, k_0)}{||W_t(G_i, :)||} \tag{7}$$

where the indicator function $\psi_{i,t}(u, v) = \mathbb{I}\left[G_i^{2D}(u, v, t) > (1 - r_{i,t,k_0})\right]$ selects the central region of each projected 2D Gaussian G_i^{2D}, ensuring that only this region contributes to rendering. $M_{k_0}^{G,s}$ denotes the Gaussians segmented by the procedures described in Secs. 3.1 and 3.2 for temporal segment s, to which the current frame t belongs. Accordingly, the rendering formulation for the segmentation result of object k_0 is modified as follows:

$$c(u, v, t) = M_{k_0}^{G,s} \sum_i \phi_{i,t}(u, v)\, \psi_{i,t}(u, v)\, T_i(u, v, t) \tag{8}$$

where $T_i(u, v, t) = \prod_{j<i} \left(1 - \phi_{j,t}(u, v)\, M_{k_0}^{G,s}\, \psi_{j,t}(u, v)\right)$, representing the accumulated transmittance up to G_i^{2D}, with each preceding Gaussian contributing only from its central region defined by r_{i,t,k_0}.

Table 1: Quantitative comparison of our method against SA4D [10] and SADG [21] on HyperNeRF [28] (mIoU/mAcc in %).

Methods	americano		chickchicken		cut-lemon1		espresso		hand		keyboard	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
SA4D [10]	84.08	99.43	87.93	96.91	83.40	96.55	73.08	98.64	**91.77**	98.63	84.97	97.33
SADG [21]	78.64	99.09	93.89	98.56	**87.60**	**97.64**	69.66	98.43	91.46	**98.72**	87.35	97.85
Ours	**87.48**	**99.61**	**95.14**	**98.87**	86.34	97.38	**90.33**	**99.62**	91.61	98.67	**94.11**	**99.07**

Methods	oven-mitts		slice-banana		split-cookie		torchocolate		average	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
SA4D [10]	66.09	91.30	77.25	91.21	86.40	99.32	80.89	99.47	81.58	96.89
SADG [21]	92.18	98.63	89.36	96.58	86.17	99.32	**87.56**	**99.70**	86.39	98.45
Ours	**93.49**	**98.93**	**90.35**	**96.99**	**93.86**	**99.73**	86.32	99.68	**90.90**	**98.86**

Table 2: Quantitative comparison of our method against three SOTA methods on Neu3D [19] (mIoU/mAcc in %).

Methods	coffee_martini		cook_spinach		cut_roasted_beef		flame_steak		sear_steak		average	
	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
SA4D [10]	89.46	99.37	90.62	99.47	90.73	99.45	88.10	99.35	88.83	99.35	89.55	99.40
SADG [21]	91.47	99.50	91.92	99.56	92.88	99.59	87.45	99.32	89.62	99.42	90.67	99.48
Split4D [9]	91.75	99.52	92.81	99.60	**96.55**	**99.81**	88.99	99.41	93.19	99.63	92.66	99.59
Ours	**93.27**	**99.63**	**95.29**	**99.75**	93.34	99.64	**91.85**	**99.59**	**94.46**	**99.70**	**93.64**	**99.66**

Ideally, the removed portion of each Gaussian, corresponding to cumulative probability $1 - r_{i,t,k_0}$, lies outside the target object k_0.In practice, retaining only the central region with probability r_{i,t,k_0} is an approximation and does not guarantee that the removed portion lies entirely outside, so a single application may still cause boundary leakage. We therefore apply this truncation iteratively to progressively refine each Gaussian's rendering threshold r_{i,t,k_0}.

In the initialization stage,the probability p_{i,t,k_0} from untruncated Gaussians before any range control is used to initialize the threshold r_{i,t,k_0}. We then iteratively perform Instance Tracing over all training frames to update each truncated Gaussian's probability p_{i,t,k_0} (Eqs. 6 and 7) and the threshold r_{i,t,k_0}.Let l denote the iteration index, $p^{(l)}_{i,t,k_0}$ is computed based on Gaussian regions truncated according to $r^{(l)}_{i,t,k_0}$. Then, $r^{(l+1)}_{i,t,k_0}$ follows a multiplicative updating rule:

$$r^{(l+1)}_{i,t,k_0} = r^{(l)}_{i,t,k_0} \times p^{(l)}_{i,t,k_0} \tag{9}$$

From empirical observations, we find that when the number of iterations l is sufficiently large, the probability $p^{(l)}_{i,t,k_0}$ for each Gaussian at each frame tends to either 1 (indicating that the Gaussian's rendering range lies entirely within the target object) or 0 (indicating that it lies completely outside the target object). At this point, the value of $r^{(l)}_{i,t,k_0}$ is considered to have effectively converged.

After each iteration l, the average probability $p^{(l)}_{avg,k_0}$ is computed over all segmented Gaussians and frames in the current scene. The iterative process will be terminated when $p^{(l)}_{avg,k_0} - p^{(l-1)}_{avg,k_0} < 0.001$. In most of our experiments, RRC typically stabilizes within 10 iterations. As an example, the evolution of p_{avg,k_0} in the *americano* is illustrated by the green line in Figure 4.

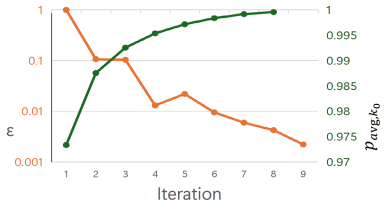

Figure 4: A line chart of the reduction rate $\epsilon^{(l)}$ and the average probability $p^{(l)}_{avg,k_0}$ at iteration l.

After the two iterative stages described in Secs. 3.1 and 3.3, we obtain a segmentation result with well-defined object boundaries for the target object k_0, stored in Gaussian masks $M^G_{k_0} \in \{0,1\}^{N\times S}$ and the cumulative probability $M^r_{k_0} \in [0,1]^{N\times T}$.

4 Evalutaion

In this section, we first introduce the evalutaion setup, including the datasets, benchmarks, evaluation metrics, and baselines (Sec. 4.1). We then compare our method with recent competitive methods on the HyperNeRF [28] and Neu3D [19] datasets (Sec. 4.2). We conduct ablation studies to evaluate the impact of different components of our method on the experimental results (Sec. 4.3). Finally, we discuss the limitations of our method (Sec. 4.4).

4.1 Evalutaion Setup

Datasets: We evaluate our method on two widely-used datasets: HyperNeR [28] is captured using a single moving monocular camera, observing dynamic scenes with diverse human-object interactions; Neu3D [19] consists of multi-view videos recorded with a rig of

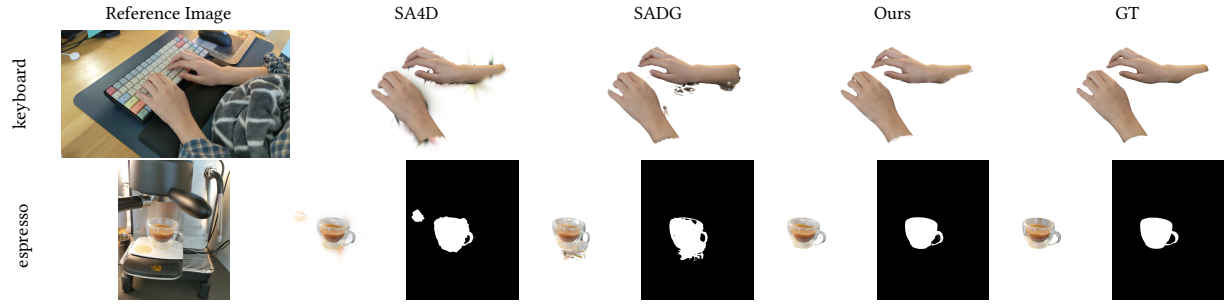

Figure 5: Qualitative comparison of our method against SA4D [10] and SADG [21] on HyperNeRF [28].

Figure 6: Qualitative comparison of our method against three SOTA methods on Neu3D [19].

18–21 cameras at 30 FPS, depicting a person performing cooking tasks in a kitchen. Following SADG [21], we conduct experiments on HyperNeRF scenes: *americano*, *split-cookie*, *oven-mitts*, *espresso*, *chickchicken*, *hand1-dense-v2*, *torchocolate*, *slice-banana*, *keyboard*, and *cut-lemon1*; and on Neu3D scenes: *coffee_martini*, *cook_spinach*, *cut_roasted_beef*, *flame_steak*, and *sear_steak*. We adopt the benchmark annotations provided in SADG for evaluation.

Evaluation Metrics: To evaluate the quality of the segmented masks, we follow the evaluation protocol of prior works, adopting Mean Intersection over Union (mIoU) and Mean Pixel Accuracy (mAcc) as quantitative metrics for segmentation performance. In addition, we compare the runtime of our method with SA4D [10] and SADG [21] for Gaussian identity feature learning and target object extraction on HyperNeRF [28] and Neu3D [19] , excluding scene reconstruction and 2D segmentation. Only segmented Gaussians are considered during rendering, without accounting for occlusions from other Gaussians, consistent with the setting in SADG. All experiments were conducted on a single NVIDIA RTX 3090 GPU.

Baselines: We compare our method with recent competitive methods SA4D [10], SADG [21] and Split4D [9]. Note that as Split4D relies on motion-enabled Gaussians that are not well-suited for monocular datasets such as HyperNeRF [28], it is evaluated only on Neu3D [19]. Both SA4D and our method are trained using the same 4D Gaussian point clouds and DEVA [6] 2D segmentation results , ensuring a fair comparison. For SADG, we follow the original training settings provided in the official implementation. For Split4D, we contacted the authors to obtain results evaluated under the SADG benchmarks and the same mask rendering protocol.

For multi-view dataset Neu3D [19], considering that DEVA [6] requires temporally continuous image sequences, we construct its input by selecting the two training views closest to the test view and concatenate their sequences, with one in chronological order and the other in reverse order. In the experiments, *cam05* and *cam06* are selected, where the image sequence from *cam05* is in chronological order and that from *cam06* is reversed.

4.2 Comparison

Tables 1 and 2 report the quantitative comparison of our method against prior works on HyperNeRF [28] and Neu3D [19] datasets, following SADG [21] benchmark annotations. Our method outperforms the other methods in terms of both mIoU and mAcc.

For HyperNeRF [28], we visualize segmentation results of the *keyboard* and *espresso* in Figure 5. Our method achieves notable improvements over the other two approaches, producing cleaner object boundaries both visually and in mask results. In contrast, SA4D [10] and SADG [21] exhibit erroneous floating points and boundary leakage, negatively affecting rendering quality.

For Neu3D [19], we visualize segmentation results of *cook_spinach* and *flame_steak* in Figure 6. Since only the two training views closest to the test view are fed to DEVA [6], multi-view supervision for our method and SA4D [10] may be limited. Nevertheless, our method still outperforms SADG [21] and Split4D [9] quantitatively, largely due to its improved suppression of boundary overflow in occluded regions caused by foreground objects.

Table 3 reports the runtime of our method, SA4D [10] and SADG [21] to extract the target object on HyperNeRF [28] and

Table 3: Efficiency and accuracy comparison of our method against SA4D [10] and SADG [21] on HyperNeRF [28] and Neu3D [19]. Time denotes the average time (in minutes) required to obtain the target object on each dataset, measured as the total time spent on identity feature learning and target object Gaussians extraction (mIoU/mAcc in %) .

Methods	HyperNeRF			Neu3D		
	time	mIoU	mAcc	time	mIoU	mAcc
SA4D [10]	19.71	81.58	96.89	32.23	89.55	99.40
SADG [21]	63.05	86.39	98.45	93.48	90.67	99.48
Ours	**0.87**	**90.90**	**98.86**	**1.60**	**93.64**	**99.66**

Table 4: Ablation study.We evaluate our method under different configurations on HyperNeRF [28].

Methods	mIoU (%)	mAcc (%)
(i) Ours w/o TS&RRC	84.65	98.07
(ii) Ours w/o TS	90.48	98.82
(iii) Ours w/o RRC	85.33	98.14
(iv) Ours w/ TS&RRC	**90.90**	**98.86**

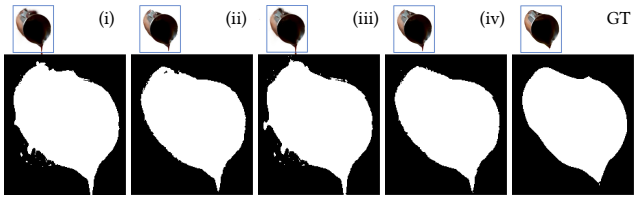

Figure 7: Qualitative ablation study in the *americano* scene.

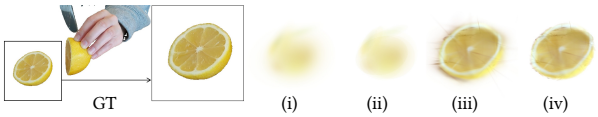

Figure 8: Qualitative ablation study in the *cut-lemon1* scene.

Neu3D [19] datasets, measured as the average total time per scene for both instance feature learning and target object segmentation, excluding scene reconstruction and 2D segmentation. Our method outperforms the other two learning-based approaches in both efficiency and quantitative metrics. This advantage stems from our two-stage training-free iterative refinement framework, which eliminates the costly process of scene-specific identity feature learning while enabling fast and accurate target object segmentation.

4.3 Ablation Studies

Table 4 shows an ablation study on HyperNeRF [28], comparing the full method with variants using only Temporal Segmentation (TS), only Gaussian Rendering Range Control (RRC), or neither.

The results show that using TS to handle variations in Gaussian identities improves segmentation, compared to using temporally consistent Gaussian identities. This is because TS can identify time steps where significant identity changes occur in the Gaussians that make up the target object and adaptively determine the appropriate temporal segments. For relatively stable scenes, longer segments are formed to provide more multi-view supervision and prevent frequent Gaussian flickering, whereas for scenes with rapid changes, shorter segments are generated to capture such variations. In contrast, temporally consistent Gaussian identities are only suitable for stable scenes and struggle when Gaussian identities change significantly. The qualitative results (Figure 8) on *cut-lemon1* demonstrate the effectiveness of TS: as lemon slices are cut off, the Gaussians composing the target object experience significant changes, which are correctly handled only when TS is applied.

When the RRC is removed, both mIoU and mAcc drop substantially. This is because it refines segmentation from the temporal segment level to the frame level, while also distinguishing core and peripheral contribution regions within each Gaussian in the spatial domain, which reduces the constraints imposed by Gaussian shapes and enables more precise segmentation. Qualitative results (Figure 7) on *americano* further illustrate that RRC effectively suppresses Gaussian overflow in boundary regions, highlighting its critical role in improving segmentation accuracy.

4.4 Limitations

Though TIBR4D can achieve efficient 4D Gaussian segmentation with high-quality boundaries, several limitations exist and can be explored in the future: 1) Instance tracing heavily depends on the accuracy of 2D instance masks. Although the proposed two iterative stages substantially reduce spurious floating Gaussians and boundary leakage, severe errors in the video segmentation may still propagate into 4D Gaussians and are difficult to eliminate. 2) Our method focuses on iterative refinement for target object extraction and is therefore not intended for full panoptic segmentation. Although panoptic maps can be obtained either by directly projecting instance tracing probabilities or by extracting and merging individual objects, both strategies are suboptimal in terms of segmentation quality or computational efficiency.

5 Conclusion

We presented TIBR4D, an object-level segmentation framework for dynamic Gaussian scenes that lifts video segmentation into 4D without requiring additional feature learning. The framework adopts a two-stage tracing-guided iterative boundary refinement strategy. In the first stage, Iterative Gaussian Instance Tracing (IGIT) is combined with adaptive temporal segmentation (TS) to extract accurate Gaussian point clouds while handling identity changes. In the second stage, frame-wise Gaussian Rendering Range Control (RRC) refines Gaussian contributions to mitigate boundary overflow and improve segmentation precision. Extensive experiments on HyperNeRF [28] and Neu3D [19] demonstrate that TIBR4D achieves superior segmentation accuracy compared to existing methods [9, 10, 21], while maintaining competitive efficiency. Ablation studies further validate the effectiveness of each component, particularly in challenging scenes involving boundary overflow and identity variations.

Acknowledgments

We would like to thank Yongzhen Hu, Sida Peng, and Xiaowei Zhou from the State Key Laboratory of CAD&CG at Zhejiang University, for providing the Split4D experimental data and helping us with the experiments. This work was partially supported by the Zhejiang Province "Vanguard" and "Geese Leading" Research and Development Plan (2025C01073).

References

[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5855–5864.

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2023. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 19697–19705.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9650–9660.

[4] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. 2025. Segment any 3d gaussians. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 1971–1979.

[5] Jiazhong Cen, Zanwei Zhou, Jiemin Fang, Wei Shen, Lingxi Xie, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, et al. 2023. Segment anything in 3d with nerfs. *Advances in Neural Information Processing Systems* 36 (2023), 25971–25990.

[6] Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. 2023. Tracking anything with decoupled video segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1316–1326.

[7] Bin Dou, Tianyu Zhang, Zhaohui Wang, Yongjia Ma, Zejian Yuan, and Nanning Zheng. 2024. Learning segmented 3D Gaussians via efficient feature unprojection for zero-shot neural scene segmentation. In *International Conference on Neural Information Processing*. Springer, 398–412.

[8] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. 2022. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*. 1–9.

[9] Yongzhen Hu, Yihui Yang, Haotong Lin, Yifan Wang, Junting Dong, Yifu Deng, Xinyu Zhu, Fan Jia, Hujun Bao, Xiaowei Zhou, et al. 2025. Split4D: Decomposed 4D Scene Reconstruction Without Video Segmentation. *ACM Transactions on Graphics (TOG)* 44, 6 (2025), 1–15.

[10] Shengxiang Ji, Guanjun Wu, Jiemin Fang, Jiazhong Cen, Taoran Yi, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. Segment any 4d gaussians. *arXiv preprint arXiv:2407.04504* (2024).

[11] Kim Jun-Seong, GeonU Kim, Kim Yu-Ji, Yu-Chiang Frank Wang, Jaesung Choe, and Tae-Hyun Oh. 2025. Dr. splat: Directly referring 3d gaussian splatting via direct language embedding registration. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 14137–14146.

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

[13] Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. 2023. Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 19729–19739.

[14] Chung Min Kim, Mingxuan Wu, Justin Kerr, Ken Goldberg, Matthew Tancik, and Angjoo Kanazawa. 2024. Garfield: Group anything with radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21530–21539.

[15] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. 2023. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4015–4026.

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

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

[18] Isaac Labe, Noam Issachar, Itai Lang, and Sagie Benaim. 2024. DGD: Dynamic 3D Gaussians Distillation. In *European Conference on Computer Vision*. Springer, 361– 378.

[19] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. 2022. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5521–5531.

[20] Wanhua Li, Renping Zhou, Jiawei Zhou, Yingwei Song, Johannes Herter, Minghan Qin, Gao Huang, and Hanspeter Pfister. 2025. 4d langsplat: 4d language gaussian splatting via multimodal large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 22001–22011.

[21] Yun-Jin Li, Mariia Gladkova, Yan Xia, and Daniel Cremers. 2024. Sadg: Segment any dynamic gaussian without object trackers. *arXiv preprint arXiv: 2411.19290* (2024).

[22] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. 2024. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8508–8520.

[23] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. 2024. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20654–20664.

[24] Weijie Lyu, Xueting Li, Abhijit Kundu, Yi-Hsuan Tsai, and Ming-Hsuan Yang. 2024. Gaga: Group any gaussians via 3d-aware memory bank. *arXiv preprint arXiv:2404.07977* (2024).

[25] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

[26] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193* (2023).

[27] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. 2021. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5865–5874.

[28] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. 2021. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228* (2021).

[29] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10318–10327.

[30] Minghan Qin, Wanhua Li, Jiawei Zhou, Haoqian Wang, and Hanspeter Pfister. 2024. Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20051–20060.

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. PmLR, 8748–8763.

[32] Hongyu Shen, Junfeng Ni, Yixin Chen, Weishuo Li, Mingtao Pei, and Siyuan Huang. 2025. Trace3d: Consistent segmentation lifting via gaussian instance tracing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6656–6666.

[33] Jin-Chuan Shi, Miao Wang, Hao-Bin Duan, and Shao-Hua Guan. 2024. Language embedded 3d gaussians for open-vocabulary scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5333–5343.

[34] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20310–20320.

[35] Junyi Wu, Van Nguyen Nguyen, Benjamin Planche, Jiachen Tao, Changchang Sun, Zhongpai Gao, Zhenghao Zhao, Anwesa Choudhuri, Gengyu Zhang, Meng Zheng, et al. 2025. Consistent Instance Field for Dynamic Scene Understanding. *arXiv preprint arXiv:2512.14126* (2025).

[36] Junyi Wu, Jiachen Tao, Haoxuan Wang, Gaowen Liu, Ramana Rao Kompella, and Yan Yan. 2025. Orientation-anchored Hyper-Gaussian for 4D Reconstruction from Casual Videos. *arXiv preprint arXiv:2509.23492* (2025).

[37] Yanmin Wu, Jiarui Meng, Haijie Li, Chenming Wu, Yahao Shi, Xinhua Cheng, Chen Zhao, Haocheng Feng, Errui Ding, Jingdong Wang, et al. 2024. Opengaussian: Towards point-level 3d gaussian-based open vocabulary understanding. *Advances in Neural Information Processing Systems* 37 (2024), 19114–19138.

[38] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. 2024. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20331–20341.

[39] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. 2024. Gaussian grouping: Segment and edit anything in 3d scenes. In *European Conference on Computer Vision*. Springer, 162–179.

[40] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. 2024. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition. 19447–19456.

[41] Zhaojie Zeng, Yuesong Wang, Lili Ju, and Tao Guan. 2025. Frequency-Aware Density Control via Reparameterization for High-Quality Rendering of 3D Gaussian Splatting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 9833–9841.