



VolSegGS: Segmentation and Tracking in Dynamic Volumetric Scenes via Deformable 3D Gaussians

Siyuan Yao  and Chaoli Wang 

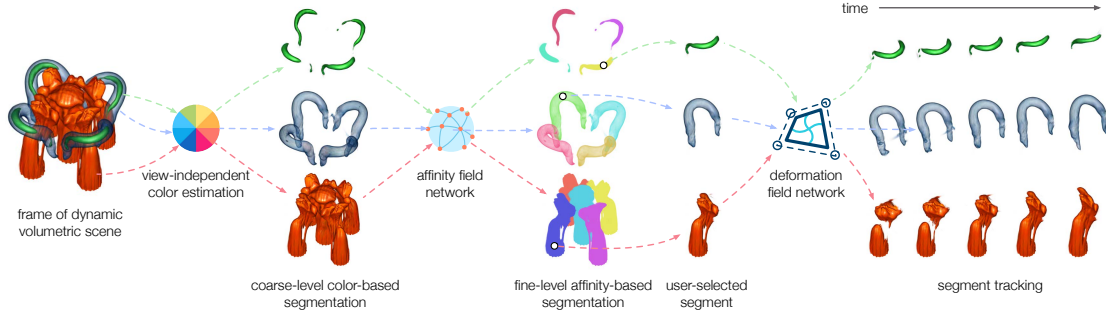


Fig. 1: Overview of VolSegGS. Deformable 3D Gaussians are learned to represent a dynamic scene. Segmentation is then performed in two stages: (1) coarse-level segmentation based on approximate view-independent colors of the Gaussians and (2) fine-level segmentation leveraging an affinity field network. Tracking over the dynamic scene is achieved using a deformation field network.

Abstract—Visualization of large-scale time-dependent simulation data is crucial for domain scientists to analyze complex phenomena, but it demands significant I/O bandwidth, storage, and computational resources. To enable effective visualization on local, low-end machines, recent advances in view synthesis techniques, such as neural radiance fields, utilize neural networks to generate novel visualizations for volumetric scenes. However, these methods focus on reconstruction quality rather than facilitating interactive visualization exploration, such as feature extraction and tracking. We introduce VolSegGS, a novel Gaussian splatting framework that supports interactive segmentation and tracking in dynamic volumetric scenes for exploratory visualization and analysis. Our approach utilizes deformable 3D Gaussians to represent a dynamic volumetric scene, allowing for real-time novel view synthesis. For accurate segmentation, we leverage the view-independent colors of Gaussians for coarse-level segmentation and refine the results with an affinity field network for fine-level segmentation. Additionally, by embedding segmentation results within the Gaussians, we ensure that their deformation enables continuous tracking of segmented regions over time. We demonstrate the effectiveness of VolSegGS with several time-varying datasets and compare our solutions against state-of-the-art methods. With the ability to interact with a dynamic scene in real time and provide flexible segmentation and tracking capabilities, VolSegGS offers a powerful solution under low computational demands. This framework unlocks exciting new possibilities for time-varying volumetric data analysis and visualization.

Index Terms—Volume visualization, novel view synthesis, scene segmentation, segment tracking, deformable Gaussian splatting

1 INTRODUCTION

Domain scientists across scientific and engineering disciplines often model complex phenomena through large-scale simulations. They typically run these simulations on high-performance computing resources to analyze time-dependent processes. As a result, these simulations generate vast amounts of high-resolution raw volumetric data over hundreds or even thousands of timesteps. Standard *direct volume rendering* (DVR) techniques require access to these volumetric data during rendering. The large data size demands significant I/O bandwidth, storage, and computational power for visualization and analysis. This poses considerable challenges, making it difficult for domain scientists to perform these tasks on standard desktop computers.

In DL4SciVis [62], recent advances in *visualization generation* [3, 18, 20, 22] aim to tackle these challenges by enabling *novel view synthesis* (NVS) of volumetric scenes using multi-view 2D rendering images instead of 3D volumetric data. These methods go beyond *data generation* [14–17, 19, 57, 73] and *neural compression* [13, 35, 53, 56, 70]. They significantly reduce data transfer and storage requirements by ensuring

the solutions remain independent of the underlying data resolution. For instance, InSituNet [20] and CoordNet [18] synthesize high-quality visualization images by training deep neural networks on 2D rendering images. However, using generative and global networks leads to slow training, making these solutions less desirable for interactive applications. Furthermore, these works rely on interpolating 2D images for synthesizing novel views without incorporating 3D awareness, yielding subpar-quality synthesized visualizations.

In contrast, a more recent work of ViSNeRF [74] utilizes *neural radiance fields* (NeRF) [40] to represent dynamic volumetric scenes. Leveraging the factorization techniques from TensorRF [6], ViSNeRF requires a small number of training images, achieves fast training, and produces synthesized results with superior quality. Nevertheless, ViSNeRF still falls short in rendering speeds due to the required DVR computations inherent in NeRF.

Another key limitation of these visualization generation techniques is that, in the absence of the original volumetric data and the corresponding *transfer function* (TF), they do not support altering the appearance of visualizations at runtime. This restriction hampers the flexibility to adjust visualizations and examine specific regions in detail. StyleRF-VolVis [58] addresses this limitation by enabling color-based content segmentation of a volumetric scene learned by NeRF to support downstream targeted appearance editing. Nevertheless, this solution is limited to static 3D scenes, lacks real-time rendering for certain style edits, and restricts segmentation to color-based regions without refinement for precise segmentation.

To enhance the performance and flexibility in visualization gener-

• The authors are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA.
E-mail: {syao2, chaoli.wang}@nd.edu.

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.
Digital Object Identifier: xx.xxx/TVCG.201x.xxxxxx

ation and provide the previously unavailable capability of segment extraction and tracking, we propose VolSegGS, **Volumetric scene Segmentation based on Gaussian Splatting (GS)**. As an alternative solution for NeRF, the revival of direct projection of Gaussian splats for volume visualization through the tremendous success of 3DGS [28] points out a promising direction for real-time NVS. Our VolSegGS is a novel framework that utilizes *deformable* 3D Gaussians for *dynamic* volumetric scene representation, offering new 3D segmentation and tracking capabilities. The primary objective is to retain the dynamic scene representation of ViSNeRF while achieving real-time rendering. Specifically, instead of using ViSNeRF to store the densities and colors of a dynamic scene, we leverage its architecture to create an efficient *deformation field network* to capture the deformation of 3D Gaussians. This allows us to represent the dynamic volumetric scene while maintaining interactive rendering through the rasterization of 3D Gaussians, avoiding the computationally intensive DVR process embedded in NeRF.

Furthermore, we propose a *two-level segmentation* strategy for flexible segmentation of the 3D volumetric scene. First, the *coarse-level* segmentation is based on the colors of 3D Gaussians. Second, the *fine-level* segmentation utilizes an *affinity field network* trained with 2D masks generated by the *segment anything model* (SAM) [32]. Users can select a segment of interest at an appropriate level and continuously track it throughout the dynamic scene. This is possible because our segmentation is performed directly on the Gaussians. Therefore, the segmented regions naturally follow temporal deformations of 3D Gaussians, ensuring consistent tracking over time.

Our VolSegGS framework provides a unified solution for segmentation, tracking, and rendering, empowering the exploration of a dynamic volumetric scene with greater efficiency and flexibility. To summarize, our contributions are as follows:

- VolSegGS employs deformable 3D Gaussians to represent dynamic volumetric scenes learned from multi-view 2D volume rendering images. This eliminates the need for the original 3D volume and enables real-time exploratory visualization.
- We propose a two-level segmentation strategy for flexible volumetric scene segmentation and utilize the deformation of 3D Gaussians to continuously track arbitrary segments in real time.
- We evaluate VolSegGS against state-of-the-art methods for dynamic scene representation and 3D segmentation, demonstrating its effectiveness in real-time NVS and 3D segmentation across multiple time-varying datasets.
- We showcase the real-time segment tracking capabilities of the complete VolSegGS framework across various dynamic scenarios, supported by quantitative evaluations that highlight its accuracy and consistency.

2 RELATED WORK

Visualization generation. Visualizing large-scale, time-dependent simulation data demands significant I/O bandwidth, storage, and computational resources. To mitigate these challenges, deep learning techniques have been developed to generate visualizations directly via neural networks. These methods facilitate TF optimization [3], rendering effects adjustment [22], image super-resolution [2, 63, 65], NVS [18], and parameter space exploration [20, 51], all without requiring access to the original volumetric data. Recent advances have introduced neural representations of volumetric data and 3D scenes via *scene representation networks* (SRNs) [64, 68, 69], enabling high-fidelity visualization with strong 3D consistency while greatly reducing storage and I/O costs.

Building upon the SRN framework, VolSegGS leverages deformable 3D Gaussians to represent dynamic volumetric scenes. It is computationally efficient compared to existing approaches and enables real-time NVS, facilitating interactive and dynamic visualization exploration.

Novel view synthesis. For NVS, NeRF [40] has gained significant attention for its ability to reconstruct high-fidelity 3D scenes from 2D images. NeRF represents a 3D scene as a continuous function, utilizing a *multi-layer perceptron* (MLP) to map spatial coordinates to density and color values. However, NeRF suffers from slow training and rendering, primarily due to its implicit representation and

the computational cost of DVR. In response, a series of follow-up works [6, 11, 42, 77] have introduced explicit feature grids, either alone or combined with lightweight MLPs, to enhance 3D scene representation efficiency. While these approaches have reduced training times to minutes, they still fail to achieve truly interactive rendering. A more recent advance, 3DGS [28], represents 3D scenes using 3D Gaussians and replaces DVR with efficient rasterization, enabling real-time rendering without relying on neural networks. Although these methods have been limited to static scenes, numerous studies [44, 45, 67, 71, 72] have explored deformable radiance fields to model temporal changes in dynamic scenes.

In volume visualization, researchers have leveraged NeRF and GS for high-fidelity visualization. For instance, ViSNeRF [74] employs a multi-dimensional NeRF to facilitate flexible visualization synthesis and parameter exploration. ReVolVE [75] reconstructs volumes from multi-view training images for visualization enhancement. StyleRF-VolVis [58] applies NeRF for expressive volumetric stylization, and iVR-GS [59] and TexGS-VolVis [55] utilize editable GS for inverse volume rendering and stylization.

VolSegGS adopts deformable 3D Gaussians to represent dynamic volumetric scenes. Unlike ViSNeRF, which also handles dynamic scenes, VolSegGS achieves real-time rendering while enabling temporal tracking of 3D segments via Gaussian deformation. Additionally, incorporating ViSNeRF’s hybrid representation for the deformation field, VolSegGS improves reconstruction quality and training efficiency over prior deformable radiance field methods.

3D segmentation. In conventional volume visualization using DVR, TFs map voxel intensities to color and opacity, effectively serving as a basic form of 3D segmentation. Building on this concept, traditional methods [24, 25, 37, 46, 54, 61] extract high-dimensional features from volume data to enable more advanced classification using multi-dimensional TFs. However, handling high-dimensional features for large-scale volume data is impractical on local machines, and manually designing multi-dimensional TFs is both complex and time-consuming. Another research direction [9, 60] applies deep learning for semantic segmentation of 3D volume data. While these methods produce expert-quality results, their reliance on manual annotations makes them costly in both time and human effort.

A more recent trend leverages 2D-based foundation segmentation models, performing slice-by-slice segmentation on volume data. For example, MedSAM [38] applies the *segment anything model* (SAM) [32] to segment medical images from CT and MRI scans. Although foundation models reduce training costs, slice-by-slice processing remains inherently inefficient. With the emergence of NeRF and 3DGS, researchers have explored more diverse and efficient approaches for integrating 2D foundation models into 3D segmentation. These approaches include: transforming multi-view 2D masks into 3D masks [5, 7, 23], distilling semantic knowledge from foundation models into 3D representations [12, 29, 33], and training affinity fields with 2D mask supervision from foundation models [4, 8, 30, 76].

VolSegGS introduces a two-level strategy for flexible 3D segmentation of volumetric scenes: (1) coarse-level segmentation relies on the color attributes of 3D Gaussians, aligning with TF-based classification; (2) fine-level segmentation employs an affinity field network trained using 2D masks generated by SAM, allowing foundation models to be efficiently integrated into 3D segmentation. This design ensures compatibility with traditional TF-based segmentation methods while leveraging the strengths of modern foundation models. The affinity field enables multi-scale scene decomposition with a single training pass, making it an efficient alternative to traditional segmentation approaches. Moreover, semantic priors learned from real-world data may not generalize well to simulated datasets, making knowledge distillation approaches less effective for scientific visualization, further highlighting the advantages of our affinity field-based approach.

Segment tracking. Traditional approaches track segments of interest in time-varying volumetric data by explicitly comparing geometric features across consecutive timesteps. Early methods [10, 26, 41, 49, 52] use algorithms that match overlapping regions with similar characteristics between adjacent timesteps. While these methods achieve accurate

tracking, they require each target segment to be processed separately across all timesteps, leading to significant inefficiencies in practical applications. Another class of approaches [48, 66] leverages merge trees to capture the hierarchical topological structure of volume data. By comparing merge trees and constructing connection graphs, these methods track regions defined by the merge trees. While this approach enables global tracking, computing merge trees for every timestep remains computationally expensive.

Unlike previous methods, VolSegGS models the deformation of 3D Gaussians to capture temporal variations in dynamic volumetric scenes. This allows for real-time visualization exploration and segment tracking at any point in time, including unseen intermediate timesteps.

3 VolSegGS

While methods like InSituNet [20] and ViSNeRF [74] have successfully enabled the exploration of dynamic visualization scenes without requiring the original volume data, VolSegGS extends this capability by integrating robust 3D segmentation techniques and enabling real-time tracking and exploration of segmented regions across a dynamic scene.

As shown in Figure 1, VolSegGS begins by optimizing deformable 3D Gaussians (Sections 3.1 and 3.2) to represent the dynamic scene. Users can then select a scene frame at any desired timestep for 3D segmentation, which is applied to the corresponding deformed Gaussians. The segmentation process consists of two key stages: (1) coarse-level color-based segmentation (Section 3.3), where Gaussians are grouped based on their approximate view-independent colors; and (2) fine-level affinity-based segmentation (Section 3.4), which refines the segmentation by learning affinity features from 2D masks generated by SAM. Users choose a scene frame to pick a segment of interest for tracking. Since segmentation is performed directly on the Gaussians, the segmented regions naturally follow Gaussian deformations, ensuring consistent tracking throughout the dynamic scene.

3.1 3D Gaussian Splatting

3DGS [28] is a highly efficient alternative to NeRF [40] for representing 3D scenes. Unlike NeRF, which relies on neural network-based implicit representations, 3DGS employs a rasterization-based approach with a fully explicit representation, enabling real-time rendering. The Gaussian function $G(\mathbf{x})$ at a spatial position \mathbf{x} is defined as

$$G(\mathbf{x}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu})\right), \quad (1)$$

where $\boldsymbol{\mu}$ represents the spatial mean, and Σ is the covariance matrix, which encodes the anisotropic shape of the Gaussian. The covariance matrix Σ can be decomposed as

$$\Sigma = \mathbf{R} \mathbf{S} \mathbf{S}^T \mathbf{R}^T, \quad (2)$$

where \mathbf{R} is the rotation matrix and \mathbf{S} is the scaling matrix. To facilitate separate optimization of these factors, we parameterize the scaling vector \mathbf{s} and the rotation quaternion \mathbf{r} as independent Gaussian attributes rather than directly optimizing the covariance matrix Σ .

During rendering, 3D Gaussians are projected onto the 2D image plane. This requires transforming the 3D covariance matrix Σ into a 2D covariance matrix Σ' , defined as

$$\Sigma' = \mathbf{J} \mathbf{W} \Sigma \mathbf{W}^T \mathbf{J}^T, \quad (3)$$

where \mathbf{W} is the viewing transformation matrix, and \mathbf{J} is the Jacobian matrix of the affine approximation of the projective transformation.

The final pixel color \mathbf{C} in the 2D rendered image is determined by blending N overlapping Gaussians in order, computed as

$$\mathbf{C} = \sum_{i \in N} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (4)$$

where \mathbf{c}_i is the view-dependent color of the i -th Gaussian, parameterized by *spherical harmonic* (SH) coefficients. The opacity term α_i is computed as a weighted product of the Gaussian’s covariance matrix Σ and its intrinsic opacity o . Note that in volume visualization, view-dependent color is produced during the shading process in DVR. In this work, we employ Blinn-Phong shading with ambient, diffuse, and

specular components to enhance perceptual clarity. While specular reflection is inherently view-dependent, our use of headlight illumination also introduces view dependency into the diffuse component. In DVR volumetric scenes, the opacity attribute of Gaussians plays a critical role in representing the scene’s semi-transparency, enabling the visualization of overlapping structures. As a result, each 3D Gaussian at a spatial position \mathbf{x} is parameterized by five learnable attributes: $(\boldsymbol{\mu}, \mathbf{r}, \mathbf{s}, \mathbf{c}, o)$, representing spatial mean, rotation quaternion, scaling vector, view-dependent color, and opacity.

3.2 Deformable 3D Gaussians for Dynamic Scene

Recent works [36, 67, 71, 72] have extended 3DGS to represent dynamic scenes by introducing time-dependent modifications to Gaussian attributes. These modifications, often called *deformations*, allow 3D Gaussians to adapt over time, leading to *deformable 3D Gaussians*, which incorporate time-varying attributes. Here, we present the formulation of deformable 3D Gaussians as used in VolSegGS.

Deformable 3D Gaussian. We define the deformation of a canonical 3D Gaussian G as ΔG , i.e., $\Delta G_t = G_t - G$, where G_t represents the deformed 3D Gaussian at timestep t . The deformation ΔG captures changes in mean position $\Delta \boldsymbol{\mu}$, rotation $\Delta \mathbf{r}$, scaling $\Delta \mathbf{s}$, and opacity Δo . In particular, geometric deformations—including $\Delta \boldsymbol{\mu}$, $\Delta \mathbf{r}$, and $\Delta \mathbf{s}$ —model changes in the geometry of visible regions in a dynamic scene. Meanwhile, opacity deformation Δo captures the appearance and disappearance of scene regions over time. The color \mathbf{c} remains unchanged to ensure consistent coarse-level segment tracking. By predicting ΔG_t , we obtain the deformed Gaussian G_t with attributes $(\boldsymbol{\mu}_t, \mathbf{r}_t, \mathbf{s}_t, \mathbf{c}, o_t)$. These updated Gaussians G_t are then used to render the scene frame at timestep t .

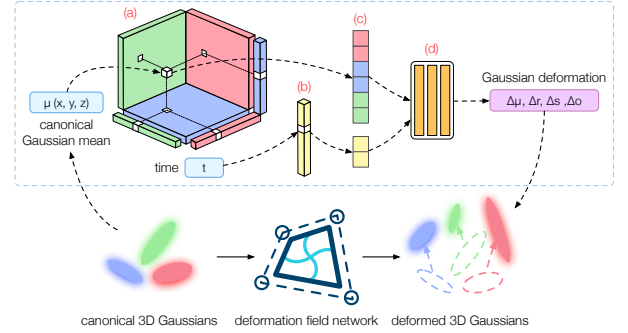


Fig. 2: The deformation field network takes the mean positions of 3D Gaussians as input and outputs their deformations. It features an explicit encoder structure consisting of (a) spatial feature planes and vectors, as well as (b) a temporal feature vector. The features are sampled from these planes and vectors, then (c) concatenated and fed into (d) a lightweight MLP decoder.

Deformation field network. Following [67, 71], for a set of canonical 3D Gaussians \mathcal{G} , we predict the global deformation $\Delta \mathcal{G}$ using a *deformation field network* \mathcal{F} , such that $\Delta \mathcal{G}_t = \mathcal{F}(\mathcal{G}, t)$ at timestep t . Inspired by ViSNeRF [74], as shown in Figure 2, our \mathcal{F} adopts a hybrid architecture, integrating an explicit spatiotemporal structure encoder \mathcal{H} and a lightweight MLP decoder \mathcal{D} . In the encoder \mathcal{H} , the explicit 4D feature tensor $\mathcal{T}^4 \in \mathbb{R}^{XYZT}$ is decomposed into three spatial feature matrices (\mathbf{M}^{XY} , \mathbf{M}^{XZ} , and \mathbf{M}^{YZ}), three spatial feature vectors (\mathbf{v}^X , \mathbf{v}^Y , and \mathbf{v}^Z), and one temporal feature vector (\mathbf{v}^T). This decomposition significantly reduces memory consumption while preserving expressiveness. The formulation is expressed as

$$\mathcal{T}^4 = \mathcal{T}^3 \circ \mathcal{T}^1 \quad (5)$$

$$= \left(\sum_{r=1}^{R_s} \mathbf{M}_r^{XY} \circ \mathbf{v}_r^Z + \mathbf{M}_r^{XZ} \circ \mathbf{v}_r^Y + \mathbf{M}_r^{YZ} \circ \mathbf{v}_r^X \right) \circ \sum_{r=1}^{R_t} \mathbf{v}_r^T, \quad (6)$$

where R_s and R_t are the numbers of low-rank and one-rank components in spatial and temporal space, respectively. If the spatial resolution is N (i.e., $N = X = Y = Z$), and the temporal resolution is T , this decomposition reduces the complexity of the deformation field from

$O(N^3T)$ to $O(R_sN^2 + R_sN + R_fT)$. The encoder \mathcal{H} takes the means of \mathcal{G} as input and outputs the sampled features in spatiotemporal space. The decoder \mathcal{D} takes these encoded spatiotemporal features as input and predicts the deformation $\Delta\mathcal{G}$. As shown in Figure 3, the hybrid structure allows VolSegGS to generate higher-quality results with sharper edges and more precise structures than the fully implicit one.

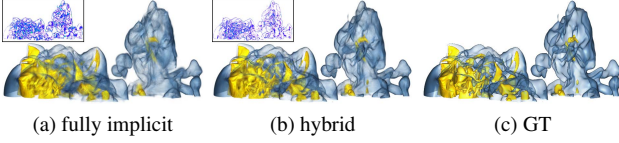


Fig. 3: Comparison of deformation field network encoder structures via rendered images of the Tangaroa dataset. The corner images highlight the perceptible pixel-wise differences, with colors ranging from purple to green to red, corresponding to low, medium, and high differences.

Optimization. Before learning the deformation of 3D Gaussians, we train the canonical 3D Gaussians \mathcal{G} following the 3DGS framework [28], without incorporating temporal information. To optimize the Gaussians, we minimize the L2 loss between the rendered image \hat{I} at a training view and the ground truth (GT) image I

$$\mathcal{L}_{L2} = \|\hat{I} - I\|_2^2. \quad (7)$$

After warming up \mathcal{G} , we jointly optimize \mathcal{G} and \mathcal{F} . Since the encoder \mathcal{H} maintains an explicit representation, we introduce *total variation* (TV) regularization to smooth the learned features. The TV loss for a 2D matrix \mathbf{M} and a 1D vector \mathbf{v} is defined as

$$\begin{aligned} \mathcal{L}_{TV} &= \mathcal{L}_{TV_1} + \mathcal{L}_{TV_2}, \\ \mathcal{L}_{TV_1} &= \sum_{\mathbf{v} \in \mathcal{V}} \sum_i \|\mathbf{v}_i - \mathbf{v}_{i-1}\|_2^2, \\ \mathcal{L}_{TV_2} &= \sum_{\mathbf{M} \in \mathcal{M}} \sum_{i,j} \left(\|\mathbf{M}_{i,j} - \mathbf{M}_{i-1,j}\|_2^2 + \|\mathbf{M}_{i,j} - \mathbf{M}_{i,j-1}\|_2^2 \right), \end{aligned} \quad (8)$$

where \mathcal{V} represents the set of vectors, and \mathcal{M} is the set of matrices.

In practice, we observed that using L2 loss alone introduced irregular artifacts due to overfitting. To mitigate the artifacts and smooth the rendering results, we incorporated an additional *structural dissimilarity* (DSSIM) loss, defined as

$$\mathcal{L}_{DSSIM} = 1 - \text{SSIM}(\hat{I}, I), \quad (9)$$

where $\text{SSIM}(\hat{I}, I)$ is the *structural similarity index measure* (SSIM) between the rendered image \hat{I} and the GT image I .

The joint optimization of \mathcal{G} and \mathcal{F} is guided by the full loss function

$$\mathcal{L} = \mathcal{L}_{L2} + \lambda_1 \mathcal{L}_{TV} + \lambda_2 \mathcal{L}_{DSSIM}, \quad (10)$$

where λ_1 and λ_2 are weights that balance the contributions of TV and DSSIM losses. In the appendix, we provide a detailed analysis of the loss functions and their impact on the performance of VolSegGS.

3.3 Coarse-Level Color-Based Segmentation

In scientific visualization, DVR uses a TF to map scalar voxel values to corresponding colors and opacities, enhancing the contrast between different components in a 3D dataset. However, because DVR consists of overlapping semi-transparent layers, distinguishing individual components in a 2D rendering after compositing becomes challenging. By reconstructing a 3D scene from 2D images using 3D Gaussians, we can identify distinct, coarse-level components based on the color attributes of the Gaussians. For simplicity, we assume that Gaussian colors remain time-invariant, ensuring consistent color-based segmentation across a dynamic scene.

View-independent color approximation. Due to DVR shading and lighting effects, a Gaussian’s color \mathbf{c} may vary depending on the viewing direction \mathbf{d} . Thus, before clustering Gaussians, we approximate the *view-independent* color $\tilde{\mathbf{c}}$ by averaging the *view-dependent* colors $\mathbf{c}_{\mathbf{d}}$ across all sampled viewing directions \mathbf{D}

$$\tilde{\mathbf{c}} \approx \frac{1}{\|\mathbf{D}\|} \sum_{\mathbf{d} \in \mathbf{D}} \mathbf{c}_{\mathbf{d}}. \quad (11)$$

We then apply a clustering algorithm, such as k-means, to group Gaussians based on their approximate view-independent colors. Color differences are measured using Euclidean distance in the RGB color space,

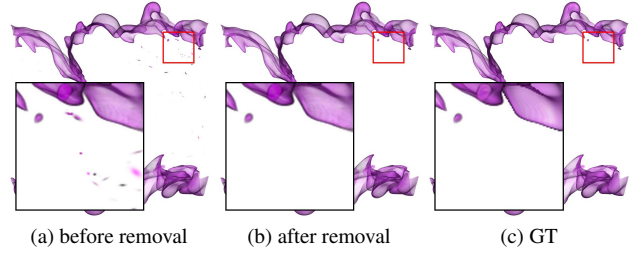


Fig. 4: Outlier removal using the combustion dataset shows the purple segment after coarse-level segmentation.

and the cluster centroids serve as the representative colors for different components. Users can select components by choosing a representative color or clicking directly on the rendered image.

Outlier removal. In practice, color-based segmentation can produce outliers due to the visualization’s lighting effects or semi-transparent layers. When segments are viewed individually, these outliers become evident and can negatively affect the quality of 2D mask generation in the subsequent fine-level segmentation. To mitigate this issue, we employ an outlier removal technique. Specifically, we search for neighbors within a small-radius sphere for each Gaussian. The Gaussian is removed from the segmentation if the number of neighbors falls below a predefined threshold. Although simple, this technique effectively removes outliers and improves the overall quality of the segmentation results, as shown in Figure 4.

3.4 Fine-Level Affinity-Based Segmentation

Color-based segmentation identifies components in the 3D scene based on the colors of 3D Gaussians. However, it cannot distinguish fine-grained structures within components that share similar colors. To address this limitation, we introduce fine-level affinity-based segmentation, which lifts 2D masks generated from SAM to a 3D affinity field, subdividing the coarse-level color-based segmentation results.

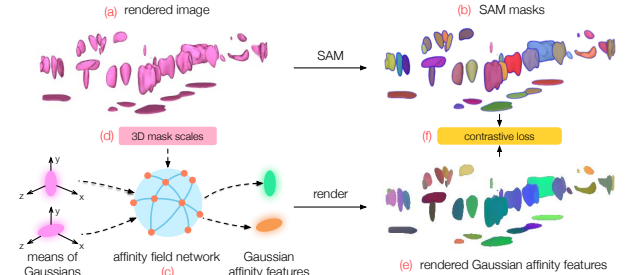


Fig. 5: Affinity field network. (a) A rendered view of the mantle dataset using VolSegGS. (b) 2D masks generated by SAM. (c) The affinity field network takes the means of the Gaussians and 3D mask scales as input to generate affinity features for the Gaussians. (d) 3D mask scales control the segmentation granularity. (e) Affinity features of the Gaussians rendered in the view. (f) Contrastive loss used for optimizing the affinity field network.

Affinity field network. Recent works on Gaussian Grouping [8, 30, 76] have utilized feature fields to model the affinity between Gaussians, effectively grouping those that are closely related. Instead of directly optimizing the Gaussian affinity feature as an additional attribute, as shown in Figure 5 (c), we employ a lightweight MLP as the *affinity field network*, which uses the mean of each Gaussian to query its affinity feature. This approach offers two key advantages: (1) the implicit MLP generates a smoother, more continuous affinity field, helping mitigate the impact of inconsistent masks across different views, and (2) the model remains compact, as its size is independent of the total number of Gaussians. To offer hierarchical fine-level segmentation, the affinity field network takes an additional 3D mask scale input to control the granularity of segmentation, as indicated in Figure 5 (d) and Figure 6.

2D mask generation. As illustrated in Figure 5 (b), we utilize SAM [32] to generate 2D masks for training the affinity field net-

work. First, users select a scene frame at timestep t from the dynamic scene to perform fine-level segmentation on the deformed Gaussians G_t . VolSegGS then renders each coarsely segmented region in the selected scene frame separately from multiple views. Finally, we use SAM to generate 2D masks from the rendered images. We generate a grid of points for each image and use the SAM predictor to produce three candidate masks per point at different scales. We select the mask with the highest confidence score and apply *non-maximum suppression* (NMS) [43] to refine overlapping masks. As a result, each view produces a set of masks representing the segmented regions. To estimate the 3D scale for each mask, we identify the Gaussians within the mask and calculate the standard deviation of their means. Due to the challenge of consistently registering masks across different views, we train the affinity field network on a per-view basis.

Optimization. To optimize the affinity field network, we first select a random training view and obtain all masks generated by SAM along with their corresponding 3D scales. Then, we use the affinity field network to query the affinity feature \mathbf{f} of each deformed Gaussian G_t at the user-selected timestep t , using the 3D scale of its corresponding SAM mask. Next, we render the affinity feature using the same blending process as described in Equation 4, i.e.,

$$\mathbf{F} = \sum_{i \in N} \mathbf{f}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \quad (12)$$

As shown in Figure 5 (f), we compute the *contrastive loss* $\mathcal{L}_{\text{feat}}$ using the rendered affinity features of a view to optimize the affinity field network

$$\mathcal{L}_{\text{feat}} = \sum_{i \in N} \sum_{j \in N} \delta(\mathbf{F}_i, \mathbf{F}_j), \text{ where} \quad (13)$$

$$\delta(\mathbf{F}_i, \mathbf{F}_j) = \begin{cases} 1 - \langle \mathbf{F}_i, \mathbf{F}_j \rangle, & \text{if in the same mask} \\ \langle \mathbf{F}_i, \mathbf{F}_j \rangle, & \text{otherwise} \end{cases} \quad (14)$$

where \mathbf{F}_i and \mathbf{F}_j are rendered affinity features. $\langle \mathbf{F}_i, \mathbf{F}_j \rangle$ denote the cosine similarity between \mathbf{F}_i and \mathbf{F}_j . After training the affinity field network, VolSegGS can segment the deformed Gaussians at timestep t by querying their affinity features at multiple scales, thereby producing hierarchical fine-level segmentation, as shown in Figure 6.

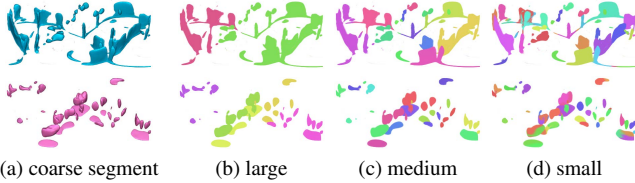


Fig. 6: Multi-scale fine-level segmentation with various granularities using the mantle dataset. Top row: cyan segment. Bottom row: pink segment.

Table 1: Datasets for visualization generation of dynamic scenes, with training images in PNG format.

dataset	volume resolution	volume size (GB)	# timesteps for training	# views per timestep	training image size (MB)
five jets	256×256×256×100	6.25	10	40	79.35
Tangaroa	600×360×240×100	19.31	20	30	145.22
mantle	720×402×360×100	38.82	20	20	43.03
vortex	512×512×512×100	50	30	20	169.66
combustion	960×1440×240×100	123.59	30	30	430.69

4 RESULTS AND DISCUSSION

This section presents qualitative and quantitative results for each stage of our framework, including visualization generation, 3D segmentation, and segment tracking.

4.1 Visualization Generation for Dynamic Scenes

Since effective segmentation relies on high-quality images, we assess the visualization generation quality of VolSegGS for dynamic scenes.

Datasets. As shown in Table 1, VolSegGS is evaluated using dynamic visualization scenes from five time-varying volumetric datasets. We uniformly sample 100 consecutive intermediate timesteps from each dataset and render the scene frames for each timestep at a fixed image resolution of 800×800. While all 100 timesteps are used to generate

inference images for evaluating the quality of VolSegGS and baseline methods, a subset of timesteps is evenly sampled for generating training images. The number of sampled timesteps for training is adjusted based on the variation in speed of the dynamic volumetric scene. To ensure an even distribution of training views around the volume data, camera positions are determined using the spherical Fibonacci point set [39], which uses a Fibonacci spiral to create a nearly uniform arrangement without clustering. The test set is rendered using a spherical camera system with 181 views, arranged along a spiral path with increasing elevation and azimuth angles, capturing the changing scene frames across the 100 timesteps.

Baselines. To evaluate the performance of VolSegGS, we compare it with three methods from scientific visualization and two deformable GS techniques:

- InSituNet [20]: A surrogate model using a GAN for parameter-space exploration of ensemble simulations. We modify InSituNet by adding upscaling convolutional blocks to output 1024×1024 images, which are then resized to 800×800 for evaluation.
- CoordNet [18]: A coordinate-based implicit neural representation model designed for multiple visualization tasks, including NVS and temporal interpolation.
- ViSNeRF [74]: A NeRF-based model utilizing efficient multi-dimensional factorization for dynamic visualization synthesis.
- 4DGS [72]: A deformable GS method that employs explicit time-conditioned 3D Gaussians and time-varying, view-dependent colors, supported by 4D SH.
- Deformable 3D Gaussians (D3DGS) [71]: A deformable GS method using an implicit deformation network to control the deformation of mean, rotation, and scale of Gaussians, with fixed color and opacity.

Training. All methods are trained and evaluated on a machine with an NVIDIA A40 GPU featuring 48 GB of video memory. VolSegGS follows a two-stage training process: (1) warming up canonical 3D Gaussians using the training images for 3,000 iterations, and (2) jointly training the canonical 3D Gaussians and the deformation field network for 20,000 iterations. The training batch size is set to one image. The learning rate for Gaussian attributes is consistent with 3DGS [28], while the initial learning rate for the deformation field network’s encoder is set to $1e^{-3}$, and for the decoder, it is set to $1e^{-4}$. For the full loss function, we set $\lambda_1 = 1e^{-4}$ for \mathcal{L}_{TV} . $\mathcal{L}_{\text{DSSIM}}$ is introduced only during the final 5,000 iterations of joint training, with $\lambda_2 = 0.2$.

Training parameters are set for the baselines following the reported configurations. InSituNet is trained for 125,000 iterations with a batch size of four images. CoordNet is trained for 300 epochs with a batch size of 32,000 pixels. ViSNeRF is trained for 90,000 iterations with a batch size of 4,096 pixels. Both 4DGS and D3DGS are trained for 30,000 iterations with a batch size of one image. We maintain approximately 150,000 Gaussians for 4DGS, D3DGS, and VolSegGS to ensure a fair comparison among GS methods.

Qualitative results. Figure 7 shows that 2D-based methods (InSituNet and CoordNet) produce the least satisfactory results. Specifically, InSituNet generates high-quality images but often deviates from the requested view or timestep. This issue arises because InSituNet relies on the closest available view, which may differ significantly from the requested parameters. As the available views are limited compared to the demand, the nearest matching view frequently remains substantially different from the requested one. On the other hand, the images generated by CoordNet appear blurry due to insufficient views for accurate interpolation within its implicit neural representation. As a result, both 2D-based methods require considerably more training images to effectively capture variations across views and timesteps.

Among the 3D-aware methods, ViSNeRF produces relatively blurry and noisy results for the Tangaroa, vortex, and combustion datasets, while 4DGS, D3DGS, and VolSegGS deliver sharper, more detailed results. This difference is mainly due to the rapid and simultaneous changes in scene content across these datasets. While 4DGS, D3DGS, and VolSegGS effectively capture these changes by modeling scene deformation, ViSNeRF struggles to maintain geometric consistency due to its reliance on interpolation between static scene frames. Another

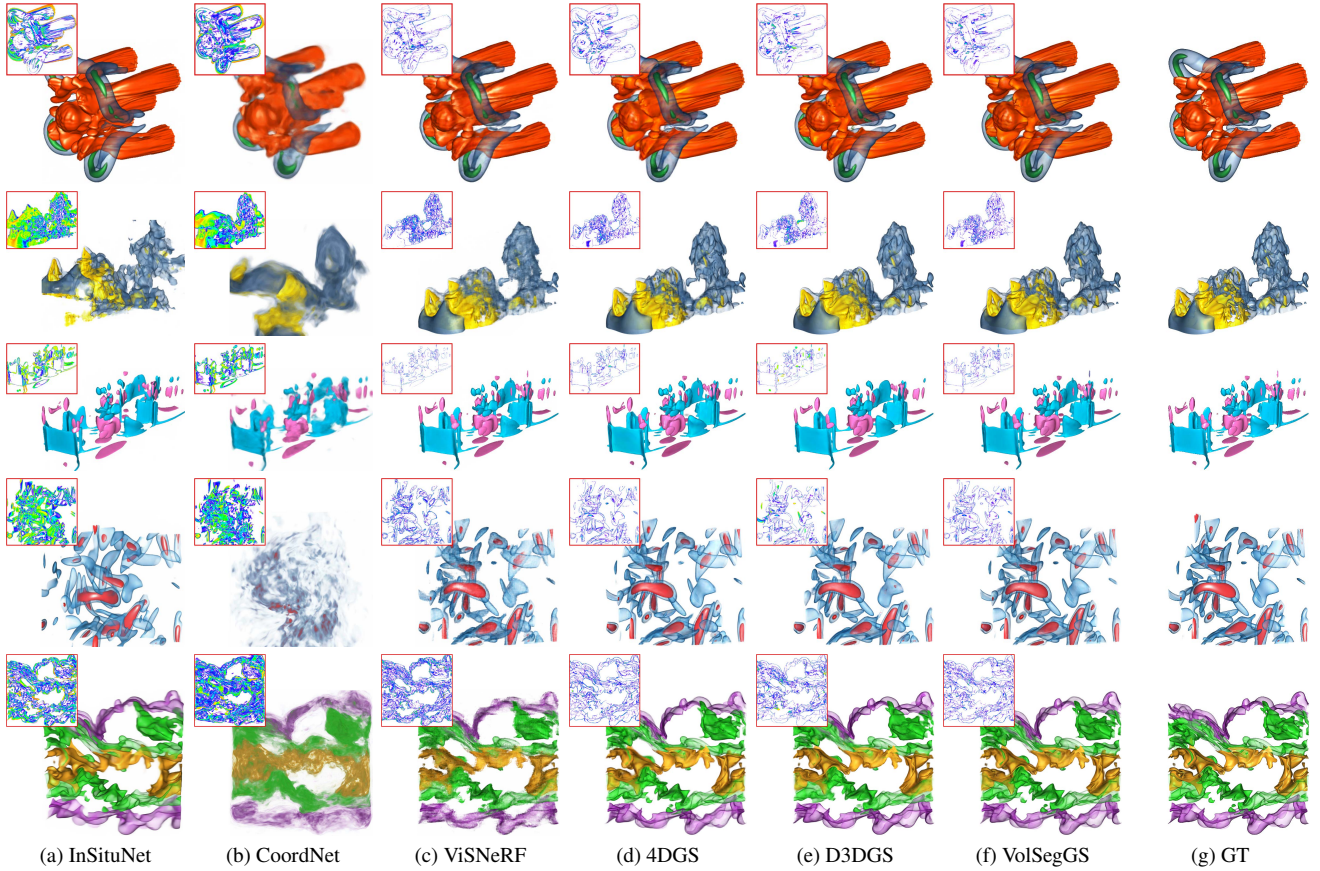


Fig. 7: Visualization generation. Top to bottom: a selected timestep of five jets, Tangaroa, mantle, vortex, and combustion.

Table 2: Visualization generation for dynamic scenes: average PSNR (dB), SSIM, LPIPS, and rendering framerate (FPS) across all 181 synthesized views, training time (TT, in minutes), and model size (MS, in MB). The best ones are highlighted in bold.

dataset	method	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	FPS \uparrow	TT \downarrow	MS \downarrow
five jets	InSituNet	14.93	0.829	0.167	5.89	1621.45	318.85
	CoordNet	16.65	0.844	0.176	0.52	2065.83	5.71
	ViSNeRF	27.36	0.955	0.035	0.28	53.98	12.32
	4DGS	26.14	0.955	0.044	214.13	68.87	96.41
	D3DGS	25.45	0.948	0.037	37.20	42.82	37.01
	VolSegGS	27.16	0.965	0.030	88.81	16.07	41.62
		15.57	0.810	0.217	5.95	1591.25	318.85
Tangaroa	CoordNet	16.87	0.828	0.234	0.49	3105.13	5.71
	ViSNeRF	25.21	0.906	0.092	0.31	60.62	12.45
	4DGS	27.33	0.945	0.059	213.62	69.38	96.96
	D3DGS	26.49	0.940	0.055	36.11	43.85	37.63
	VolSegGS	28.15	0.953	0.042	88.65	16.80	43.48
		15.11	0.860	0.204	5.86	1632.91	318.85
	CoordNet	15.75	0.861	0.229	0.51	2070.31	5.71
mantle	ViSNeRF	29.36	0.975	0.021	0.29	68.16	12.99
	4DGS	28.56	0.975	0.034	213.67	69.67	97.89
	D3DGS	26.69	0.972	0.046	36.14	43.35	37.54
	VolSegGS	29.02	0.979	0.030	89.29	15.98	43.21
		14.76	0.748	0.322	5.78	1593.98	318.85
	CoordNet	15.56	0.764	0.375	0.52	3139.56	5.71
	ViSNeRF	24.94	0.919	0.096	0.29	64.68	12.39
vortex	4DGS	25.82	0.940	0.081	209.21	73.57	105.96
	D3DGS	23.64	0.932	0.119	36.40	42.86	37.51
	VolSegGS	26.37	0.947	0.074	89.21	15.90	43.28
		13.85	0.680	0.361	5.29	1604.72	318.85
	CoordNet	14.72	0.696	0.370	0.50	4655.92	5.71
	ViSNeRF	22.45	0.800	0.202	0.28	69.33	14.15
	4DGS	25.61	0.894	0.112	213.48	78.92	105.85
combustion	D3DGS	24.55	0.882	0.107	36.57	42.72	37.48
	VolSegGS	25.76	0.897	0.092	88.35	16.92	43.37

notable observation is that ViSNeRF generally preserves specular highlights better than 4DGS, D3DGS, and VolSegGS, especially for the five jets and vortex datasets. This advantage arises because ViSNeRF uses a view-direction-conditioned MLP decoder to learn view-dependent colors, whereas 4DGS, D3DGS, and VolSegGS rely solely on SH.

Among the GS methods, which all model Gaussian deformation, the generation quality is generally similar across all datasets. However, due to the use of an implicit MLP in the deformation network, D3DGS and VolSegGS produce smoother synthesized images than 4DGS, which

uses explicit 4D Gaussians. This difference is particularly noticeable in the transparent regions of the Tangaroa, vortex, and combustion datasets. Additionally, D3DGS leads to more missing parts in the synthesized images compared to 4DGS and VolSegGS, particularly for the Tangaroa, mantle, and vortex datasets. Upon investigation, we attribute this issue to using L1 and DSSIM losses in D3DGS. Specifically, these losses provide insufficient gradient signals for capturing subtle structural details, preventing the deformation network from accurately positioning and representing Gaussians in these regions, resulting in missing content in the synthesized images. In contrast, VolSegGS employs L2 loss (which more effectively penalizes large errors in small regions) and postpones the application of DSSIM loss until the final 5,000 iterations, ensuring better structural preservation.

Quantitative results. We evaluate the quality of the synthesized images compared to the GT images using three metrics: *peak signal-to-noise ratio* (PSNR), *structural similarity index* (SSIM), and *learned perceptual image patch similarity* (LPIPS) [78]. Additionally, we report the rendering framerate, training time, and model size to assess method efficiency. Quantitative results are presented in Table 2. Overall, 3D-aware methods (ViSNeRF, 4DGS, D3DGS, and VolSegGS) consistently outperform 2D-based ones (InSituNet and CoordNet) across all datasets. Among the 3D-aware methods, ViSNeRF achieves the highest PSNR for the five jets and mantle datasets but falls behind the GS methods (4DGS, D3DGS, and VolSegGS) for the Tangaroa, vortex, and combustion datasets. While VolSegGS delivers the best generation quality within the GS methods, it also produces competitive results compared to ViSNeRF for the five jets and mantle datasets.

Rendering framerate. The GS methods offer real-time framerates, thanks to their rasterization-based rendering pipeline. Other methods evaluated in our experiments fail to meet the performance requirements for real-time exploration of dynamic scenes. Among the Gaussian-based approaches, 4DGS achieves the highest FPS. This is due to 4DGS directly utilizing fully explicit 4D Gaussians to rep-

resent dynamic scenes, making it more efficient in rendering than D3DGS and VolSegGS, which rely on deformation networks. However, 4DGS requires several regularization terms during optimization to ensure training stability, significantly increasing the overall training cost. Specifically, the regularization process involves computationally expensive k-nearest-neighbor calculations to enhance consistency among neighboring Gaussians. In contrast, D3DGS and VolSegGS employ deformation networks with an MLP that takes Gaussian means and time as inputs, naturally providing spatial smoothness without additional regularization.

Despite having the second-highest FPS, VolSegGS still comfortably meets real-time requirements, with a minimum of 87 FPS. In contrast, D3DGS achieves only about 36 FPS, making it less ideal than VolSegGS for typical 60Hz displays. The lower FPS of D3DGS is primarily due to its reliance on a large MLP-based deformation network for predicting the deformation of 3D Gaussians. VolSegGS, on the other hand, employs a hybrid deformation network consisting of an explicit feature grid combined with a lightweight MLP. This architectural difference allows VolSegGS to achieve faster training speeds than D3DGS, although it slightly increases the model size.

Table 3: Render performance comparison: DVR vs. VolSegGS.

dataset	DVR			VolSegGS		
	CPU/GPU memory (GB)	loading time (s)	rendering time (ms)	CPU/GPU memory (GB)	preparation+training time (min)	rendering time (ms)
five jets	1.23/0.91	0.68	191	2.46/1.53	1.38+16.07	11
Tangaroa	1.72/0.90	1.84	247	2.51/1.54	3.08+16.80	11
mantle	2.54/0.88	3.33	212	2.50/1.56	2.52+15.98	11
vortex	3.20/1.42	4.55	342	2.53/1.53	5.70+15.90	11
combustion	5.14/1.88	9.07	761	2.51/1.47	15.96+16.92	11

Comparison with DVR. VolSegGS achieves a rendering speedup of $17\times$ to $69\times$ compared with DVR using ParaView, as shown in Table 3. In the table, loading time refers to the average time needed to load the volume data of a single timestep into memory, and rendering time is the average time required to render a test view. For VolSegGS, the preparation time includes the full duration required to load data into ParaView and render all training views across the sampled timesteps listed in Table 1. At the cost of the upfront preparation and training time, VolSegGS eliminates data loading overhead when switching timesteps during inference. In contrast, DVR methods incur significant data loading costs, which prevent them from reaching the real-time rendering speeds demonstrated by VolSegGS. This makes VolSegGS particularly promising for real-time exploratory visualization of dynamic scenes and provides a reliable foundation for subsequent interactive segmentation and tracking tasks.

Summary. VolSegGS offers the best balance of generation quality, rendering efficiency, and training cost among all the methods compared. While VolSegGS does not significantly outperform existing deformable Gaussian methods in rendering quality, it provides a solid foundation for subsequent segmentation and tracking in exploratory visualization.

4.2 3D Segmentation for Static Scenes

In this section, we assess the 3D segmentation quality of VolSegGS on static scenes, comparing it against baseline 3D segmentation methods.

Datasets. To compare segmentation methods specifically on static scenes, we select a representative timestep from each dataset listed in Table 1, as indicated in Table 4. For each selected timestep, we render 30 views at a resolution of 800×800 , which serve as input images for training a 3DGS model. These training images are also segmented using SAM to provide 2D mask supervision for VolSegGS and baseline 3D segmentation methods. We use the same view sampling method described in Section 4.1 to generate 30 training and 181 test views. To quantitatively evaluate segmentation quality, we render manually segmented volumes using DVR as the GT and compute PSNR, SSIM, and LPIPS scores across 181 test views. Additionally, we generate corresponding 2D masks for these test views and use *intersection over union* (IoU) for comparison.

Baselines. We select two state-of-the-art 3D Gaussian segmentation methods, SAGA [4] and SAGD [23], as baseline methods. To ensure a fair comparison, we pretrain a single 3DGS model per dataset and apply VolSegGS and the baseline segmentation methods to the same pretrained model. Each 3DGS model is trained for 30,000 iterations

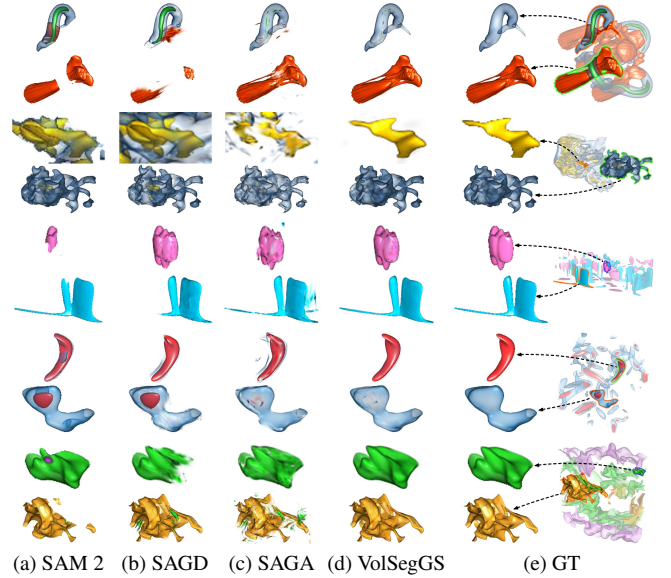


Fig. 8: 3D segmentation. Top to bottom: selected segmentation results of five jets, Tangaroa, mantle, vortex, and combustion.

without limiting the number of Gaussians while keeping all other settings at their default values. Additionally, we include SAM 2 [47], given its capability for 2D video segmentation. Since our test images are rendered along a predefined camera path, we assemble these images into a video sequence and apply SAM 2 for video segmentation.

- SAM 2 [47]: A foundation model that provides promptable visual segmentation for images and videos. It leverages a transformer architecture combined with streaming memory to enhance the performance of segment tracking.
- SAGA [4]: An efficient segmentation approach specifically designed for 3DGS. It employs scale-gated affinity features for each Gaussian to effectively capture inter-Gaussian relationships. The affinity features are optimized using contrastive learning, guided by automatically generated 2D masks obtained from SAM.
- SAGD [23]: A training-free segmentation method tailored for 3DGS. It projects 2D masks produced by SAM onto 3D Gaussians from each view and determines segmentation labels through a voting strategy that aggregates binary mask labels across all views.

Training. All experiments are conducted on a machine with an NVIDIA RTX 4090 GPU with 24 GB of video memory. We train the affinity field network of VolSegGS for 5,000 iterations using a batch size of 8,192 pixels and the Adam optimizer with a learning rate of $1e^{-3}$. For the baseline methods, we use the publicly available pretrained checkpoint (sam2.1_hiera_large) for SAM 2. For SAGA, the affinity features are optimized following the original paper’s settings, with 10,000 iterations and a batch size of 1,000 pixels. SAGD does not require training; therefore, we directly apply this method to the pretrained 3DGS model using SAM masks.

Qualitative results. Figure 8 highlights VolSegGS’s superior segmentation performance compared to SAM 2, SAGD, and SAGA. Due to minimal occlusions, all methods perform well for the least challenging mantle dataset. However, VolSegGS excels in preserving the most accurate shape. Specifically, VolSegGS retains most of the original 3D Gaussians for the mantle dataset’s cyan segment, whereas SAGA and SAGD either omit necessary Gaussians or introduce extraneous ones. This advantage stems from VolSegGS’s MLP-based deformation field network, which enables smooth and precise spatial segmentation. In contrast, methods relying entirely on explicit features of 3D Gaussians tend to introduce significant noise. A similar trend is observed in both segments of the five jets dataset: although SAGA performs relatively well, it still misses some Gaussians and introduces unnecessary ones, ultimately degrading the quality.

All baseline methods struggle to separate segments beneath overly-

Table 4: 3D segmentation for static scenes: average PSNR (dB), SSIM, LPIPS, and IoU across all 181 synthesized views, training time (TT, in minutes), and segmentation time (ST, in seconds). Refer to Figure 8 for the actual segments. The best ones are highlighted in bold.

dataset	method	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	IoU \uparrow	TT \downarrow	ST \downarrow
five jets $t = 61$ blue segment	SAM 2	26.49	0.983	0.032	66.99	—	0.05
	SAGD	25.70	0.982	0.032	74.91	—	1.26
	SAGA	32.62	0.990	0.022	86.76	8.67	0.14
	VolSegGS	37.49	0.995	0.007	96.88	1.53	0.15
five jets $t = 61$ red segment	SAM 2	20.88	0.967	0.064	44.78	—	0.05
	SAGD	20.09	0.971	0.054	46.38	—	5.13
	SAGA	29.94	0.983	0.024	91.95	8.67	0.14
	VolSegGS	37.06	0.995	0.005	94.21	1.53	0.16
Tangaroa $t = 1$ yellow segment	SAM 2	29.89	0.993	0.029	20.34	—	0.05
	SAGD	18.14	0.946	0.098	2.50	—	7.44
	SAGA	23.68	0.969	0.067	4.69	9.81	0.15
	VolSegGS	46.94	0.999	0.005	62.45	1.85	0.15
Tangaroa $t = 1$ blue segment	SAM 2	29.19	0.988	0.024	60.91	—	0.05
	SAGD	29.76	0.984	0.027	92.95	—	11.52
	SAGA	29.16	0.980	0.020	97.64	9.81	0.14
	VolSegGS	37.60	0.996	0.006	97.71	1.85	0.15
mantle $t = 1$ pink segment	SAM 2	34.28	0.996	0.019	36.67	—	0.05
	SAGD	42.14	0.999	0.002	89.59	—	1.05
	SAGA	38.25	0.996	0.043	61.27	9.21	0.14
	VolSegGS	42.59	0.999	0.001	90.39	1.67	0.15
mantle $t = 1$ cyan segment	SAM 2	32.93	0.995	0.016	72.34	—	0.04
	SAGD	33.84	0.995	0.016	78.54	—	3.27
	SAGA	32.31	0.992	0.057	62.89	9.21	0.14
	VolSegGS	39.08	0.998	0.003	87.75	1.67	0.15
vortex $t = 31$ red segment	SAM 2	29.50	0.994	0.033	40.79	—	0.05
	SAGD	35.02	0.996	0.008	68.60	—	1.41
	SAGA	31.98	0.991	0.019	54.75	10.72	0.14
	VolSegGS	41.03	0.999	0.003	83.56	1.78	0.15
vortex $t = 31$ blue segment	SAM 2	31.74	0.994	0.012	84.35	—	0.05
	SAGD	33.04	0.994	0.009	88.67	—	4.37
	SAGA	32.90	0.992	0.048	81.59	10.72	0.14
	VolSegGS	39.05	0.997	0.005	95.76	1.78	0.15
combustion $t = 100$ green segment	SAM 2	30.92	0.996	0.017	34.77	—	0.05
	SAGD	33.92	0.998	0.005	84.45	—	2.21
	SAGA	36.99	0.998	0.007	77.60	10.01	0.14
	VolSegGS	45.74	0.999	0.001	91.30	1.55	0.15
combustion $t = 100$ yellow segment	SAM 2	19.80	0.964	0.087	35.98	—	0.05
	SAGD	28.08	0.977	0.025	90.41	—	7.55
	SAGA	23.45	0.942	0.098	65.33	10.01	0.15
	VolSegGS	32.02	0.984	0.015	92.65	1.55	0.15

ing layers for the yellow segment of the Tangaroa dataset. VolSegGS, however, successfully splits segments beneath thin, semi-transparent layers by leveraging its two-level segmentation strategy. Despite verifying the correctness of input prompts using SAM masks, SAM 2 and SAGD still fail to produce accurate segmentation. SAM 2 fails due to inherent ambiguity in segment tracking, while SAGD’s voting strategy proves ineffective in resolving segmentation under such conditions. SAGA tries to segment occluded parts but is significantly impacted by noisy affinity features on Gaussians, leading to suboptimal results.

The two-level segmentation strategy of VolSegGS also effectively removes inner parts of different colors within semi-transparent segments in other datasets. SAM 2 and SAGD struggle to remove the red part for the blue segment of the vortex dataset because they rely on 2D input prompts. If users manually exclude the red part, the resulting mask remains incomplete and fails to fully capture the blue segment, leading to suboptimal segmentation. SAGA, on the other hand, leverages affinity features that effectively distinguish the blue part from the red. However, the red part is not entirely removed due to noise, leaving behind residual artifacts. Similar phenomena can also be observed from the blue segments of the five jets and the Tangaroa datasets. For the yellow segment of the combustion dataset, the close proximity of the green and yellow parts causes segmentation errors when using 2D masks for supervision. During projection onto 3D Gaussians, this overlap results in unintended blending, hindering SAGD and SAGA from extracting the yellow segment cleanly without interference from the green part.

Quantitative results. Table 4 presents the quantitative evaluation of 3D segmentation quality across all datasets. VolSegGS consistently outperforms baseline methods across all datasets in terms of PSNR, SSIM, LPIPS, and IoU, aligning with the qualitative results. From an efficiency standpoint, SAGA and VolSegGS optimize affinity features for segmenting 3D Gaussians. However, VolSegGS achieves faster training times than SAGA due to its affinity field network, which ensures smoothness and consistency among neighboring Gaussians. In contrast, SAGA explicitly optimizes affinity features for individual Gaussians, requiring more iterations to achieve the same level of consistency. Although SAGD does not require training, it is less efficient than VolSegGS in segmentation speed. Each time users provide a point prompt, SAGD introduces noticeable delays compared to SAGA and VolSegGS, making real-time interaction less responsive.

Summary. Across all static visualization scenes, VolSegGS consistently outperforms baseline segmentation methods in terms of segmentation quality. It offers a reliable solution for 3D segmentation in static scenes and establishes a strong foundation for enabling interactive segmentation and tracking in exploratory visualization.

4.3 Segment Tracking for Dynamic Scenes

In this section, we present the tracking results of VolSegGS on dynamic scenes. While we acknowledge a few unpublished concurrent works on 4D segmentation with Gaussians [27, 34], we cannot directly compare them due to their lack of open-source code. Instead, we present three use-case scenarios to illustrate the accuracy and robustness of VolSegGS in segment-tracking tasks.

Single-segment tracking. In this scenario, we select three segments at timestep 20 from the combustion dataset and track each segment individually until timestep 100. Figure 9 presents the masked results of the full scene and individually tracked segments across five selected timesteps. The results in Table 5 indicate robust tracking performance, with all individual segments maintaining an IoU above 80 throughout the sequence. By focusing on single-segment tracking, users can analyze the evolution of specific segments without distractions from the surrounding scene.

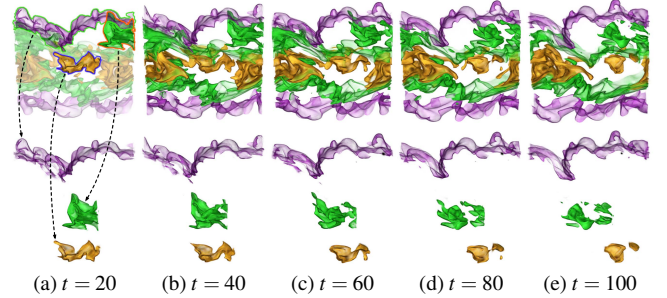


Fig. 9: Single-segment tracking with combustion. Top row: the full scene. Rest rows: tracking results of individual segments.

Table 5: Single-segment tracking with combustion: average PSNR (dB), SSIM, LPIPS, and IoU across 181 synthesized views.

segment	timestep	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	IoU \uparrow
purple	20	29.46	0.963	0.045	86.86
	40	29.23	0.960	0.049	85.77
	60	29.07	0.964	0.045	88.08
	80	28.42	0.960	0.042	88.46
	100	29.04	0.959	0.048	86.96
yellow	20	33.13	0.989	0.015	88.01
	40	34.55	0.992	0.011	87.55
	60	35.05	0.992	0.011	88.46
	80	33.60	0.992	0.011	83.59
	100	36.61	0.995	0.011	86.63
green	20	33.81	0.986	0.014	91.51
	40	32.86	0.986	0.013	88.87
	60	32.34	0.986	0.013	87.53
	80	30.33	0.982	0.012	86.08
	100	33.53	0.989	0.011	83.77

Grouped segment tracking. In this scenario, we select multiple segments at timestep 50 from the vortex dataset and group them for tracking as a whole over time. As shown in the top row of Figure 10, for the forward tracking, we observe the segment on the top-right corner splitting into two segments at timestep 80. In fact, if we start with four segments at timestep 90 and do backward tracking, the results will be the same as shown in (d) to (a), and the two segments on the right join into one segment at timestep 70. For the backward tracking, the segment in the middle disappears at timestep 20. VolSegGS maintains the group as a whole, ensuring that the segmentation is not disrupted by individual segments’ split, join, or disappearance. Changes in the rest of the scene also do not affect the tracked group. Table 6 shows that the grouped segment consistently achieves an IoU above 80 in both directions, reflecting reliable and robust tracking performance. With grouped segment tracking, users can perform a comparative analysis of multiple segments and study their collective behaviors over time.

Edited segment tracking. In this scenario, we demonstrate that VolSegGS enables versatile editing of segments while effectively tracking the edited segments across multiple timesteps. To illustrate this

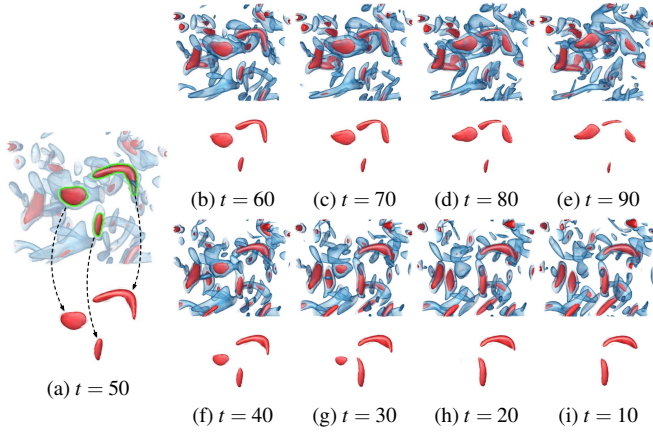


Fig. 10: Grouped segment tracking with vortex. (a) shows segmentation processed at timestep 50. (b) to (e) are forward tracking results, and (f) to (i) are backward tracking results.

Table 6: Grouped segment tracking with vortex: average PSNR (dB), SSIM, LPIPS, and IoU across 181 synthesized views.

timestep	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	IoU \uparrow
10	37.04	0.996	0.008	82.07
20	36.28	0.996	0.007	84.01
30	34.12	0.994	0.009	82.15
40	33.87	0.994	0.009	82.42
50	33.99	0.994	0.009	82.09
60	35.28	0.995	0.009	83.09
70	34.93	0.995	0.009	82.28
80	34.81	0.995	0.009	81.65
90	34.96	0.995	0.009	80.35

capability, we select several segments from the five jets dataset at timestep 60. As shown in Figure 11, we first (1) divide the red region into five segments, assigning each a unique color. Next, we (2) increase the opacity of a specific segment within the blue region. Finally, we (3) transform a segment from the green region, repositioning it away from the center and scaling it up. The backward tracking results from timestep 60 reveal that these edited segments remain consistent throughout the dynamic scene. For instance, the recolored segments remain visually distinguishable even when merging at the top and splitting vertically at timestep 20. The segment with increased opacity clearly maintains its opacity over time. Additionally, the transformed segment is consistently visible at an enlarged scale and remains separate from the main structure. This demonstrates that VolSegGS allows users to conveniently edit segments of interest at any timestep, with modifications seamlessly propagated across all other timesteps.

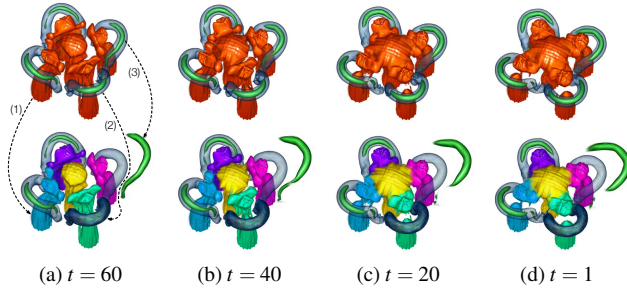


Fig. 11: Edited segment tracking with five jets. Each edited segment is modified with a different (1) color, (2) opacity, or (3) transformation.

Summary. We show the tracking capability of VolSegGS through three use-case scenarios: single, grouped, and edited segment tracking. VolSegGS effectively tracks individual, group, and edited segments over time, establishing a reliable foundation for the interactive exploration of dynamic scenes. Furthermore, VolSegGS enables real-time segment tracking throughout the dynamic scene by following the deformation of 3D Gaussians. This real-time tracking capability greatly enhances the user experience, facilitating segment evolution analysis in dynamic

scenes and enabling comparative studies across multiple segments.

4.4 Limitations

While VolSegGS learns dynamic visualization scenes from DVR images and enables real-time scene exploration and segmentation, its reliance on DVR images rather than volume data introduces three key limitations. First, the surrogate-level segmentation labels generated by VolSegGS are derived from rendered visualizations and are not directly transferable to the original volumetric data. While these labels are sufficient to support exploratory visualization tasks, such as interactive segmentation, object tracking, and downstream editing, for broader applicability, future work could investigate robust methods for mapping these labels back to volume data. Second, the DVR images used to train VolSegGS are rendered using a single TF. As a result, the overall effectiveness of segmentation and tracking is highly dependent on the chosen TF. This may limit the flexibility of VolSegGS in handling complex scenes with intricate geometries and heavy occlusions, where multiple TFs are often required to comprehensively capture structural details. A potential solution is to train multiple sets of deformable Gaussians on scenes rendered with different TFs, providing users with greater flexibility for segmentation and tracking across varying TFs. This is feasible because the Gaussians are relatively independent of one another, making merging them relatively straightforward. We view multi-TF support as a natural extension and future improvement. Third, as shown in Tables 1 and 3, both preparation time and storage requirement for generating training images tend to increase with the increasing volume size. Few-shot learning should offer a potential future direction to reduce the required training data.

Another limitation of VolSegGS is its potential difficulty with long-term tracking. For extended sequences, the deformation field network may struggle to capture the changes in the entire scene over time, especially if there are large intervals between frames or significant overall deformations. This issue could be mitigated by increasing the number of 3D Gaussians and concatenating multiple deformation field networks to handle longer sequences. However, this would increase the model size and training cost, which could reduce its overall usability.

5 CONCLUSIONS AND FUTURE WORK

We introduced VolSegGS, a novel dynamic volumetric scene segmentation framework that enables real-time rendering using deformable Gaussian representations trained on volume rendering images. Our method employs a robust two-level segmentation strategy: (1) coarse-level segmentation partitions the scene based on estimated view-independent colors of Gaussians, and (2) fine-level segmentation refines these results through an affinity field network optimized with automatically generated 2D mask supervision. Furthermore, we utilize Gaussian deformation to track segments over time while maintaining real-time performance. Our evaluations on multiple time-varying datasets show that VolSegGS outperforms state-of-the-art methods in dynamic scene representation and 3D segmentation. VolSegGS provides a practical solution for exploratory visualization of large-scale volumetric datasets by enabling interactive segmentation, tracking, and editing.

Our future work will focus on addressing the current limitations of VolSegGS and further enhancing its capabilities for dynamic volume visualization and analysis. First, we plan to investigate reliable methods for mapping image-based segmentation labels back to the original volumetric data, enabling more fundamental analysis tasks. We also aim to improve the flexibility of VolSegGS by introducing multi-TF support, allowing users to seamlessly switch between different TFs. To reduce data preparation time and storage costs, we will explore few-shot learning techniques to minimize the number of required training images. We also intend to develop efficient methods for concatenating multiple dynamic scene clips, enabling VolSegGS to handle longer and more complex temporal sequences. Additionally, given the reliance on SAM masks, a promising direction is to fine-tune the SAM model on volumetric datasets and evaluate its impact on segmentation accuracy and robustness. Finally, we envision VolSegGS as a platform for broader innovation in scientific visualization. Future research may incorporate language-guided interaction [1] or enable complex operations such as relighting, inpainting, and spatiotemporal enhancement.

ACKNOWLEDGMENTS

This research was supported in part by the U.S. National Science Foundation through grants IIS-1955395, IIS-2101696, OAC-2104158, and IIS-2401144, and the U.S. Department of Energy through grant DE-SC0023145. The authors would like to thank the anonymous reviewers for their insightful comments.

APPENDIX

1 ABLATION STUDY

We conduct an ablation study on two key components of VolSegGS: dynamic scene representation learning and segmentation. First, to train deformable 3D Gaussians, we analyze the impact of several factors, including the choice of loss function, the initialization of canonical 3D Gaussians, Gaussian opacity deformation, and the structure of the deformation field network. Next, we examine how the proposed two-level segmentation strategy contributes to overall segmentation quality improvement.

Table 1: Comparison of training VolSegGS on different loss combinations using the mantle dataset: average PSNR (dB), SSIM, and LPIPS across all 181 synthesized views. Training time (TT, in minutes) is also reported. The best ones are highlighted in bold.

loss	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	TT \downarrow
L1	27.94	0.974	0.036	15.70
L2	29.18	0.974	0.032	15.52
L2+SSIM	29.02	0.979	0.030	15.98

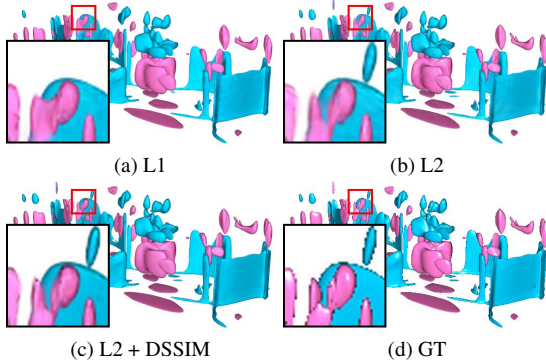


Fig. 1: Comparison of training VolSegGS on different loss combinations using the mantle dataset.

Table 2: Comparison of VolSegGS on the TV loss using the combustion dataset: average PSNR (dB), SSIM, and LPIPS across all 181 synthesized views. Training time (TT, in minutes) is also reported. The best ones are highlighted in bold.

TV loss	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	TT \downarrow
without	18.90	0.759	0.296	16.87
with	25.76	0.897	0.092	16.92

Loss function. From Table 1, we observe that the L2 loss function achieves the highest PSNR, while the combination of L2 and SSIM losses yields the best performance in terms of SSIM and LPIPS. As illustrated in Figure 1, the L1 loss results in smooth outputs with missing fine structures, whereas the L2 loss better preserves subtle details but introduces noticeable artifacts. The L2+SSIM loss produces the most visually appealing results, retaining fine details while effectively reducing artifacts. Moreover, we find that TV loss plays a critical role in the convergence of the deformation field network, as evidenced by the results in Table 2 and Figure 2. Without TV loss, the model fails to learn a coherent deformation field, likely due to the lack of spatiotemporal neighborhood consistency.

Initialization of Canonical 3D Gaussians. As shown in Table 3, initializing the canonical 3D Gaussians for 3,000 iterations provides

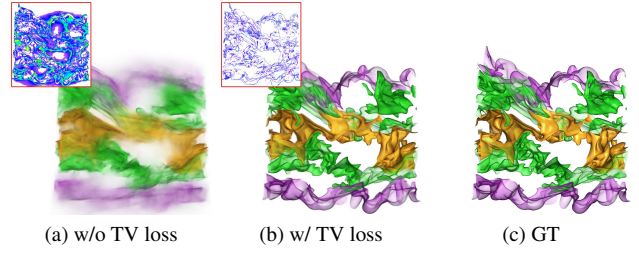


Fig. 2: Comparison of VolSegGS on the TV loss using the combustion dataset.

Table 3: Comparison of VolSegGS on initializing the canonical 3D Gaussians using the five jets dataset: average PSNR (dB), SSIM, LPIPS, and rendering framerate (FPS) across all 181 synthesized views. Training time (TT, in minutes) is also reported. The best ones are highlighted in bold.

initialization	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	FPS \uparrow	TT \downarrow
without	26.17	0.957	0.039	89.36	15.73
with	27.16	0.965	0.030	88.31	16.07

performance improvements with minimal increase in training time. The gains in PSNR, SSIM, and LPIPS exceed those achieved by an additional 5,000 iterations of joint training, as shown in Table 9. Figure 3 further illustrates that initialization leads to visibly enhanced detail reconstruction.

Gaussian opacity deformation. From Table 4, we observe that incorporating deformable opacity enables VolSegGS to achieve higher performance across PSNR, SSIM, and LPIPS. As shown in Figure 4, using fixed opacity leads to visible artifacts caused by small floating Gaussians, whereas deformable opacity more accurately models the disappearance, resulting in cleaner renderings.

Structure of deformation field network. Table 5 shows that the hybrid design delivers superior performance in PSNR, SSIM, and LPIPS compared to the fully implicit design. When both are trained for 30,000 iterations jointly with the warmed-up canonical 3D Gaussians, the hybrid design converges faster, requiring less training time. Additionally, it achieves a higher rendering framerate, despite having a slightly larger model size. Figure 5 further highlights that the fully implicit design leads to blurred reconstructions, whereas the hybrid design enables more accurate recovery of details.

Two-level segmentation. As shown in Table 6 and Figure 6, the coarse-level segmentation primarily relies on color, making it difficult to distinguish individual components that share similar colors. Fine-level segmentation captures structure but ignores color, which can make it difficult to separate inner and outer parts with different appearances. Our two-level approach successfully combines both, enabling a clear separation of regions based on color and structure. This validates its effectiveness in segmenting volume visualization scenes.

2 HYPERPARAMETER ANALYSIS

For hyperparameter analysis, we investigate three aspects that impact the rendering quality using deformable 3D Gaussians in VolSegGS: the number of sampled timesteps for training, the number of sampled views per timestep for training, and the number of joint training iterations.

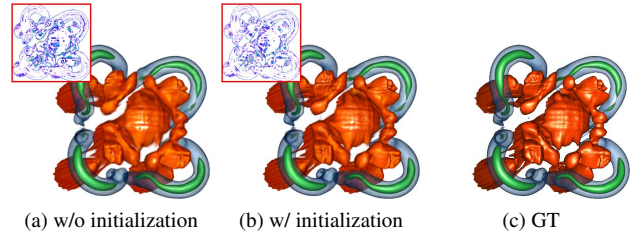


Fig. 3: Comparison of VolSegGS on initializing the canonical 3D Gaussians using the five jets dataset.

Table 4: Comparison of VolSegGS on the Gaussian opacity deformation using the vortex dataset: average PSNR (dB), SSIM, and LPIPS across all 181 synthesized views. Training time (TT, in minutes) is also reported. The best ones are highlighted in bold.

opacity	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	TT \downarrow
fixed	26.02	0.937	0.094	15.95
deformable	26.37	0.947	0.074	15.90

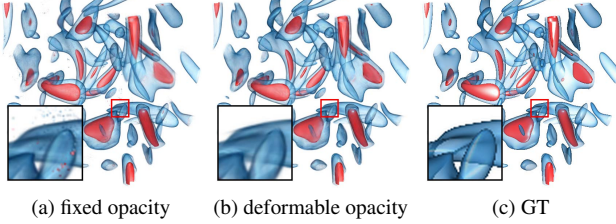


Fig. 4: Comparison of VolSegGS on the Gaussian opacity deformation using the vortex dataset.

Additionally, we evaluate the effect of the number and diversity of SAM masks from different views on the performance of the affinity field network.

Number of sampled timesteps for training. Table 7 shows that, with an insufficient number of sampled timesteps for training, VolSegGS may have difficulty reconstructing the scene accurately. Figure 7, allocating 30 timesteps allows the model to recover most of the details in the scene of the combustion dataset.

Number of sampled views per timestep for training. According to Table 8, training VolSegGS with a limited number of views per timestep reduces reconstruction quality. Figure 8 illustrates that with 30 views per timestep, the model could recover most details in the Tangaroa scene.

Number of joint training iterations. Table 9 suggests that 20,000 iterations are sufficient for jointly training the canonical 3D Gaussians and the deformation field network. As shown in Figure 9, VolSegGS can reconstruct most of the fine details in the five jets dataset after being trained for 20,000 iterations.

Number and view distribution of SAM masks. In this analysis, we investigate the impact of different numbers and spatial distributions of views on the segmentation performance of VolSegGS. As shown in Table 10 and Figure 10, our default setting generates SAM masks from 30 views, corresponding to the number of training views per timestep. We then evaluate reduced configurations using only 10 views, either evenly distributed or biased in the viewing direction along the x -, y -, or z -axis, respectively.

The results show that the affinity field network trained with SAM masks remains largely robust even when the number of views is reduced from 30 to 10. The performance drop is minimal, indicating that the network can still effectively leverage limited 2D segmentation input. However, both the number and spatial distribution of views do influence segmentation quality, as noise and ambiguity in SAM masks can lead to localized errors. Interestingly, we observe consistent improvements when the views are biased toward a specific direction. In these cases, clustering views spatially enhances the consistency of SAM masks, and for less occluded regions, such as the yellow segment, this strategy can even outperform the evenly distributed setting with more views. In contrast, more heavily occluded regions, such as the green segment,

Table 5: Comparison of VolSegGS on different structures of deformation field network using the Tangaroa dataset: average PSNR (dB), SSIM, LPIPS, and rendering framerate (FPS) across all 181 synthesized views, training time (TT, in minutes), and model size (MS, in MB). The best ones are highlighted in bold.

structure	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	FPS \uparrow	TT \downarrow	MS \downarrow
implicit	26.32	0.931	0.067	37.58	23.48	37.36
hybrid	28.15	0.953	0.042	88.65	16.80	43.48

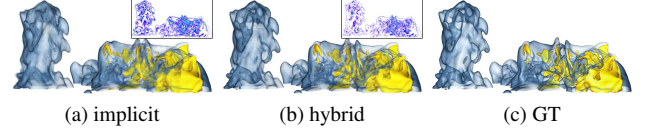


Fig. 5: Comparison of VolSegGS on different structures of deformation field network using the Tangaroa dataset.

Table 6: Comparison of VolSegGS on different segmentation methods using the vortex dataset: average PSNR (dB), SSIM, LPIPS, and IoU across all 181 synthesized views. The best ones are highlighted in bold.

segment	method	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	IoU \uparrow
red	coarse only	18.77	0.954	0.223	12.56
	fine only	31.44	0.993	0.017	45.84
	coarse+fine	42.01	0.999	0.003	84.22
blue	coarse only	16.32	0.841	0.339	7.62
	fine only	37.78	0.997	0.007	95.26
	coarse+fine	40.51	0.998	0.005	95.21

require a greater number of diverse viewpoints to achieve satisfactory segmentation results.

Note that, in the paper, we use evenly distributed views to ensure fair, consistent, and standardized experimental conditions.

3 METHOD COMPARISON AND ADDITIONAL DISCUSSION

Comparison with segmentation methods. Existing volume segmentation methods [21, 24, 25, 31, 37, 46, 50, 54, 61] primarily rely on TFs to classify voxels. Earlier methods [24, 25, 37, 46, 54, 61] improved segmentation quality by incorporating higher-dimensional features and multi-dimensional TFs. However, they often suffer from increased computational overhead and the complexity of designing multi-dimensional TFs. More recent methods [21, 31, 50] have shifted toward leveraging deep learning to assist in TF design, yet this significantly increases segmentation time.

In contrast, VolSegGS introduces a visual segmentation approach that achieves 3D segmentation by reconstructing visualizations from rendered images. Specifically, VolSegGS employs a color-based coarse segmentation strategy that aligns with TF-based colorization. Additionally, it offers a flexible, multi-scale fine segmentation capability, enabling further subdivision of coarse segments based on visual cues. While fine-level segmentation requires an initial preparation time of several minutes, it supports immediate inference. By leveraging an efficient scene representation based on 3D Gaussians instead of raw volumetric data, VolSegGS enables real-time rendering and segmentation for large-scale datasets.

It is important to note that, unlike the previously mentioned methods, VolSegGS does not support direct segmentation on raw volume data. This limitation may restrict its applicability in certain use cases and hinder direct performance comparisons with volume-based approaches. Rather than serving as a replacement, VolSegGS can complement existing methods by leveraging their TFs for coarse-level segmentation of 3D scenes.

Comparison with feature-tracking methods. Existing feature-tracking methods [10, 26, 41, 48, 49, 52, 66] for time-varying scalar field data primarily rely on deterministic algorithms. Most prior works [10, 26, 41, 49, 52] track individual features by comparing voxel values or isosurfaces across adjacent timesteps. Meanwhile, a separate line of research [48, 66] enables global feature tracking by computing and

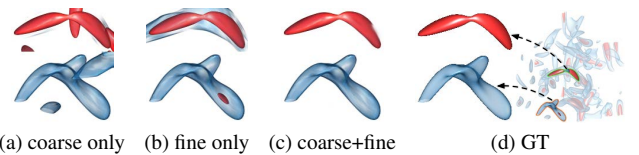


Fig. 6: Comparison of VolSegGS on different segmentation methods using the vortex dataset.

Table 7: Comparison of training VolSegGS on different numbers of sampled timesteps using the combustion dataset: average PSNR (dB), SSIM, and LPIPS across all 181 synthesized views. The best ones are highlighted in bold.

# timesteps	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
10	23.91	0.857	0.114
20	25.07	0.881	0.100
30	25.76	0.897	0.092
40	25.98	0.901	0.090

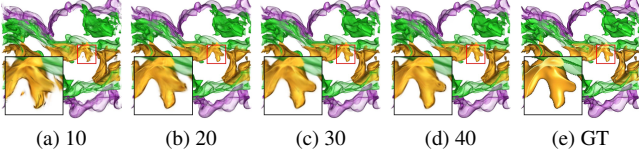


Fig. 7: Comparison of training VolSegGS on different numbers of sampled timesteps using the combustion dataset.

comparing merge trees.

In contrast, VolSegGS introduces a novel feature-tracking approach by learning a deformation field from DVR images of time-varying data. Unlike prior methods, VolSegGS tracks global features without relying on predefined critical points, isosurfaces, or merge trees. Instead, it offers greater flexibility by enabling users to track arbitrary segments without requiring additional recomputation. The time required to train the 3D Gaussians with the deformation field network is comparable to the time needed to compute merge trees. However, once trained, VolSegGS enables real-time tracking and rendering of any arbitrary segment, even for large-scale datasets. Moreover, as illustrated in Figure 11, VolSegGS can visualize the global deformation velocity of the entire scene, providing a comprehensive understanding of the volumetric scene’s evolution.

Although VolSegGS lacks the capability to directly track features in raw volume data, it is primarily designed as a visualization tool, emphasizing real-time, exploratory interaction with dynamic visualization scenes.

SAM masks for segmentation. Relying on SAM masks for segmentation may present challenges as well, as SAM has not been fine-tuned on scientific datasets. When all SAM masks from multiple views fail to accurately capture a segment, VolSegGS could lead to incomplete segmentation or mistakenly encompass adjacent regions. To mitigate this issue, our affinity feature network helps smooth segmentation results in the implicit space, while the multi-scale fine-level segmentation allows users to select smaller parts to assemble a complete segment. However, this approach may be suboptimal in certain cases and is intended only as a workaround. It would be valuable for future work to investigate fine-tuning SAM on visualization datasets for segmentation quality improvement.

REFERENCES

- [1] K. Ai, K. Tang, and C. Wang. NLI4VolVis: Natural language interaction for volume visualization via multi-LLM agents and editable 3D Gaussian splatting. *IEEE Transactions on Visualization and Computer Graphics*, 32(1), 2026. Accepted. 9
- [2] D. Bauer, Q. Wu, and K.-L. Ma. FoVolNet: Fast volume rendering using foveated deep neural networks. *IEEE Transactions on Visualization and*

Table 8: Comparison of training VolSegGS on different numbers of sampled views per timestep using the Tangaroa dataset: average PSNR (dB), SSIM, and LPIPS across all 181 synthesized views. The best ones are highlighted in bold.

# timesteps	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow
10	25.27	0.918	0.058
20	27.43	0.945	0.045
30	28.15	0.953	0.042
40	28.37	0.955	0.041

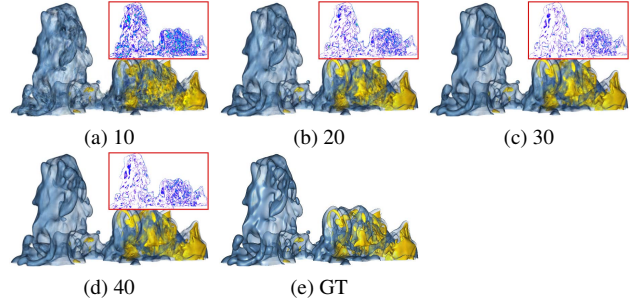


Fig. 8: Comparison of training VolSegGS on different numbers of sampled views per timestep using the Tangaroa dataset.

Table 9: Comparison of training VolSegGS on different numbers of iterations using the five jets dataset: average PSNR (dB), SSIM, and LPIPS across all 181 synthesized views. Training time (TT, in minutes) is also reported. The best ones are highlighted in bold.

# iterations	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	TT \downarrow
5,000	23.63	0.926	0.068	3.85
10,000	25.42	0.952	0.038	7.78
15,000	26.64	0.962	0.034	11.85
20,000	27.16	0.965	0.030	16.07
25,000	27.20	0.965	0.029	19.80
30,000	27.40	0.966	0.029	24.02

Computer Graphics, 29(1):515–525, 2023. doi: 10.1109/TVCG.2022.3209498 2

- [3] M. Berger, J. Li, and J. A. Levine. A generative model for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1636–1650, 2019. doi: 10.1109/TVCG.2018.2816059 1, 2
- [4] J. Cen, J. Fang, C. Yang, L. Xie, X. Zhang, W. Shen, and Q. Tian. Segment any 3D Gaussians. *arXiv preprint arXiv:2312.00860*, 2023. doi: 10.48550/arXiv.2312.00860 2, 7
- [5] J. Cen, Z. Zhou, J. Fang, C. Yang, W. Shen, L. Xie, D. Jiang, X. Zhang, and Q. Tian. Segment anything in 3D with NeRFs. In *Proceedings of Advances in Neural Information Processing Systems*, 2023. 2
- [6] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. TensorRF: Tensorial radiance fields. In *Proceedings of European Conference on Computer Vision*, pp. 333–350, 2022. doi: 10.1007/978-3-031-19824-3_20 1, 2
- [7] X. Chen, J. Tang, D. Wan, J. Wang, and G. Zeng. Interactive segment anything NeRF with feature imitation. *arXiv preprint arXiv:2305.16233*, 2023. doi: arXiv.2305.16233 2
- [8] S. Choi, H. Song, J. Kim, T. Kim, and H. Do. Click-Gaussian: Interactive segmentation to any 3D Gaussians. In *Proceedings of European Conference on Computer Vision*, pp. 289–305, 2024. doi: 10.1007/978-3-031-72646-0_17 2, 4
- [9] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In *Proceedings of International Conference on Medical Image Computing*

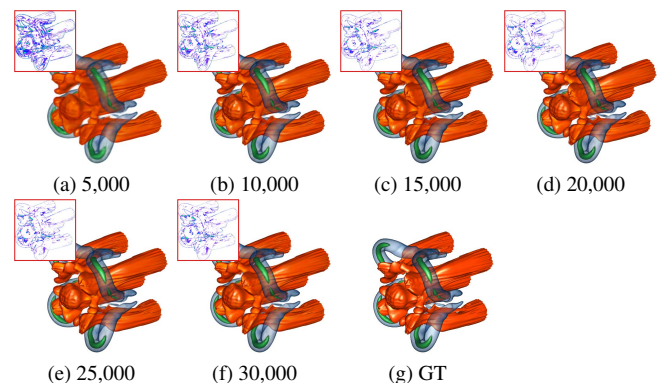


Fig. 9: Comparison of training VolSegGS on different numbers of iterations using the five jets dataset.

Table 10: Comparison of VolSegGS on different numbers and spatial distributions of views using the combustion dataset: average PSNR (dB), SSIM, LPIPS, and IoU across all 181 synthesized views. The best ones are highlighted in bold.

segment	# views	PSNR↑	SSIM↑	LPIPS↓	IoU↑
green	30 (evenly)	38.03	0.996	0.005	94.14
	10 (evenly)	32.44	0.993	0.016	89.65
	10 (x-axis)	37.04	0.996	0.008	90.91
	10 (y-axis)	37.11	0.995	0.006	88.53
	10 (z-axis)	37.34	0.995	0.006	91.71
yellow	30 (evenly)	43.48	0.999	0.004	93.48
	10 (evenly)	42.04	0.999	0.010	91.88
	10 (x-axis)	43.48	0.999	0.003	93.48
	10 (y-axis)	43.49	0.999	0.003	93.49
	10 (z-axis)	43.48	0.999	0.003	93.48

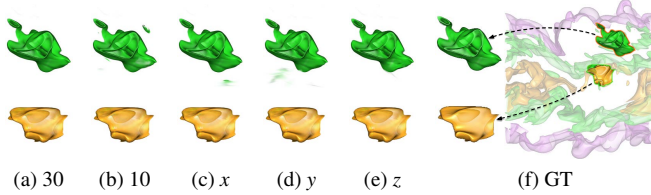


Fig. 10: Comparison of VolSegGS on different numbers and spatial distributions of views using the combustion dataset. 30 and 10 refer to segmentation results using SAM masks generated from 30 and 10 evenly distributed views, respectively. x , y , and z refer to segmentation results using SAM masks generated from 10 views biased along the x -, y -, and z -axis, respectively.



Fig. 11: Visualization of the x -axis deformation velocity: red indicates positive values, green indicates negative values, and brightness represents magnitude. Left to right: combustion, Tangaroa, and mantle.

and Computer-Assisted Intervention, pp. 424–432, 2016. doi: 10.1007/978-3-319-46723-8_49 2

- [10] S. Dutta and H.-W. Shen. Distribution driven extraction and tracking of features for time-varying data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 22(12):837–846, 2016. doi: 10.1109/TVCG.2015.2467436 2, 11
- [11] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5501–5510, 2022. doi: 10.1109/CVPR52688.2022.00542 2
- [12] R. Goel, D. Sirikonda, S. Saini, and P. J. Narayanan. Interactive segmentation of radiance fields. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4201–4211, 2023. doi: 10.1109/CVPR52729.2023.00409 2
- [13] P. Gu, D. Z. Chen, and C. Wang. NeRVI: Compressive neural representation of visualization images for communicating volume visualization results. *Computers & Graphics*, 116:216–227, 2023. doi: 10.1016/J.CAG.2023.08.024 1
- [14] P. Gu, J. Han, D. Z. Chen, and C. Wang. Reconstructing unsteady flow data from representative streamlines via diffusion and deep learning based denoising. *IEEE Computer Graphics and Applications*, 41(6):111–121, 2021. doi: 10.1109/MCG.2021.3089627 1
- [15] P. Gu, J. Han, D. Z. Chen, and C. Wang. Scalar2Vec: Translating scalar fields to vector fields via deep learning. In *Proceedings of IEEE Pacific Visualization Symposium*, pp. 31–40, 2022. doi: 10.1109/PACIFICVIS53943.2022.00012 1
- [16] J. Han and C. Wang. TSR-VFD: Generating temporal super-resolution for unsteady vector field data. *Computers & Graphics*, 103:168–179, 2022. doi: 10.1016/J.CAG.2022.02.001 1
- [17] J. Han and C. Wang. VCNet: A generative model for volume completion. *Visual Informatics*, 6(2):62–73, 2022. doi: 10.1016/J.VISINF.2022.04.004 1
- [18] J. Han and C. Wang. CoordNet: Data generation and visualization generation for time-varying volumes via a coordinate-based neural network. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):4951–4963, 2023. doi: 10.1109/TVCG.2022.3197203 1, 2, 5
- [19] J. Han, H. Zheng, Y. Xing, D. Z. Chen, and C. Wang. V2V: A deep learning approach to variable-to-variable selection and translation for multivariate time-varying data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1290–1300, 2021. doi: 10.1109/TVCG.2020.3030346 1
- [20] W. He, J. Wang, H. Guo, K. Wang, H. Shen, M. Raj, Y. G. Nashed, and T. Peterka. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):23–33, 2020. doi: 10.1109/TVCG.2019.2934312 1, 2, 3, 5
- [21] X. He, S. Yang, Y. Tao, H. Dai, and H. Lin. Graph convolutional network-based semi-supervised feature classification of volumes. *Journal of Visualization*, 25(2):379–393, 2022. doi: 10.1007/S12650-021-00787-7 11
- [22] F. Hong, C. Liu, and X. Yuan. DNN-VolVis: Interactive volume visualization supported by deep neural network. In *Proceedings of IEEE Pacific Visualization Symposium*, pp. 282–291, 2019. doi: 10.1109/PACIFICVIS.2019.00041 1, 2
- [23] X. Hu, Y. Wang, L. Fan, J. Fan, J. Peng, Z. Lei, Q. Li, and Z. Zhang. SAGD: Boundary-enhanced segment anything in 3D Gaussian via Gaussian decomposition. *arXiv preprint arXiv:2401.17857*, 2024. doi: 10.48550/arXiv:2401.17857 2, 7
- [24] R. Huang and K.-L. Ma. RGVis: Region growing based techniques for volume visualization. In *Proceedings of Pacific Conference on Computer Graphics and Applications*, pp. 355–363, 2003. doi: 10.1109/PCCGA.2003.1238277 2, 11
- [25] C. Y. Ip, A. Varshney, and J. F. J    . Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2355–2363, 2012. doi: 10.1109/TVCG.2012.231 2, 11
- [26] G. Ji, H.-W. Shen, and R. Wenger. Volume tracking using higher dimensional isosurfacing. In *Proceedings of IEEE Visualization Conference*, 2003. doi: 10.1109/VISUAL.2003.1250374 2, 11
- [27] S. Ji, G. Wu, J. Fang, J. Cen, T. Yi, W. Liu, Q. Tian, and X. Wang. Segment any 4D Gaussians. *arXiv preprint arXiv:2407.04504*, 2024. doi: 10.48550/arXiv.2407.04504 8
- [28] B. Kerbl, G. Kopanas, T. Leimk    ler, and G. Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, 2023. doi: 10.1145/3592433 2, 3, 4, 5
- [29] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. LERF: Language embedded radiance fields. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 19672–19682, 2023. doi: 10.1109/ICCV51070.2023.01807 2
- [30] C. M. Kim, M. Wu, J. Kerr, K. Goldberg, M. Tancik, and A. Kanazawa. GARField: Group anything with radiance fields. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 21530–21539, 2024. doi: 10.1109/CVPR52733.2024.02034 2, 4
- [31] S. Kim, Y. Jang, and S.-E. Kim. Image-based TF colorization with CNN for direct volume rendering. *IEEE Access*, 9:124281–124294, 2021. doi: 10.1109/ACCESS.2021.3100429 11
- [32] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Doll    r, and R. B. Girshick. Segment anything. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 3992–4003, 2023. doi: 10.1109/ICCV51070.2023.00371 2, 4
- [33] S. Kobayashi, E. Matsumoto, and V. Sitzmann. Decomposing NeRF for editing via feature field distillation. In *Proceedings of Advances in Neural Information Processing Systems*, 2022. 2
- [34] Y.-J. Li, M. Gladkova, Y. Xia, and D. Cremers. SADG: Segment any dynamic Gaussian without object trackers. *arXiv preprint arXiv:2411.19290*, 2024. doi: 10.48550/arXiv.2411.19290 8
- [35] Y. Lu, P. Gu, and C. Wang. FCNR: Fast compressive neural representation of visualization images. In *Proceedings of IEEE VIS Conference (Short Papers)*, pp. 31–35, 2024. doi: 10.1109/VIS55277.2024.00014 1
- [36] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan. Dynamic 3D Gaussians: Tracking by persistent dynamic view synthesis. In *Proceedings of International Conference on 3D Vision*, pp. 800–809, 2024. doi: 10.1109/3DV62453.2024.00044 3
- [37] B. Ma and A. Entezari. Volumetric feature-based classification and visibility analysis for transfer function design. *IEEE Transactions on Vi-*

- ualization and Computer Graphics, 24(12):3253–3267, 2018. doi: 10.1109/TVCG.2017.2776935 2, 11
- [38] J. Ma, Y. He, F. Li, L. Han, C. You, and B. Wang. Segment anything in medical images. *Nature Communications*, 15(1):654, 2024. doi: 10.1038/s41467-024-44824-z 2
- [39] R. Marques, C. Bouville, M. Ribardière, L. P. Santos, and K. Bouatouch. Spherical Fibonacci point sets for illumination integrals. *Computer Graphics Forum*, 32(8):134–143, 2013. doi: 10.1111/CGF.12190 5
- [40] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of European Conference on Computer Vision*, pp. 405–421, 2020. doi: 10.1007/978-3-030-58452-8_24 1, 2, 3
- [41] C. Muelder and K.-L. Ma. Interactive feature extraction and tracking by utilizing region coherency. In *Proceedings of IEEE Pacific Visualization Symposium*, 2009. doi: 10.1109/PACIFICVIS.2009.4906833 2, 11
- [42] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15, 2022. doi: 10.1145/3528223.3530127 2
- [43] A. Neubeck and L. V. Gool. Efficient non-maximum suppression. In *Proceedings of International Conference on Pattern Recognition*, pp. 850–855, 2006. doi: 10.1109/ICPR.2006.479 5
- [44] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 5865–5874, 2021. doi: 10.1109/ICCV48922.2021.00581 2
- [45] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10318–10327, 2021. doi: 10.1109/CVPR46437.2021.01018 2
- [46] T. M. Quan, J. Choi, H. Jeong, and W.-K. Jeong. An intelligent system approach for probabilistic volume rendering using hierarchical 3D convolutional sparse coding. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):964–973, 2018. doi: 10.1109/TVCG.2017.2744078 2, 11
- [47] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. SAM 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. doi: 10.48550/arXiv.2408.00714 7
- [48] H. Saikia and T. Weinkauff. Global feature tracking and similarity estimation in time-dependent scalar fields. *Computer Graphics Forum*, 36(3):1–11, 2017. doi: 10.1111/cgf.13163 3, 11
- [49] A. Schnorr, D. N. Helmrich, D. Denker, T. W. Kuhlen, and B. Hentschel. Feature tracking by two-step optimization. *IEEE Transactions on Visualization and Computer Graphics*, 26(6):2219–2233, 2020. doi: 10.1109/TVCG.2018.2883630 2, 11
- [50] O. Sharma, T. Arora, and A. Khattar. Graph-based transfer function for volume rendering. *Computer Graphics Forum*, 39(1):76–88, 2020. doi: 10.1111/CGF.13663 11
- [51] N. Shi, J. Xu, H. Guo, J. Woodring, and H.-W. Shen. VDL-Surrogate: A view-dependent latent-based model for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):820–830, 2023. doi: 10.1109/TVCG.2022.3209413 2
- [52] D. Silver and X. Wang. Tracking and visualizing turbulent 3D features. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):129–141, 1997. doi: 10.1109/2945.597796 2, 11
- [53] H. Son, J. Noh, S. Jeon, C. Wang, and W.-K. Jeong. MC-INR: Efficient encoding of multivariate scientific simulation data using meta-learning and clustered implicit neural representations. In *Proceedings of IEEE VIS Conference (Short Papers)*, 2025. Accepted. 1
- [54] K. P. Soundararajan and T. Schultz. Learning probabilistic transfer functions: A comparative study of classifiers. *Computer Graphics Forum*, 34(3):111–120, 2015. doi: 10.1111/CGF.12623 2, 11
- [55] K. Tang, K. Ai, J. Han, and C. Wang. TexGS-VolVis: Expressive scene editing for volume visualization via textured Gaussian splatting. *IEEE Transactions on Visualization and Computer Graphics*, 32(1), 2026. Accepted. 2
- [56] K. Tang and C. Wang. ECNR: Efficient compressive neural representation of time-varying volumetric datasets. In *Proceedings of IEEE Pacific Visualization Conference*, pp. 72–81, 2024. doi: 10.1109/PACIFICVIS60374.2024.00017 1
- [57] K. Tang and C. Wang. STSR-INR: Spatiotemporal super-resolution for time-varying multivariate volumetric data via implicit neural representation. *Computers & Graphics*, 119:103874, 2024. doi: 10.1016/J.CAG.2024.01.001 1
- [58] K. Tang and C. Wang. StyleRF-VolVis: Style transfer of neural radiance fields for expressive volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):613–623, 2025. doi: 10.1109/TVCG.2024.3456342 1, 2
- [59] K. Tang, S. Yao, and C. Wang. iVR-GS: Inverse volume rendering for explorable visualization via editable 3D Gaussian splatting. *IEEE Transactions on Visualization and Computer Graphics*, 31(6):3783–3795, 2025. doi: 10.1109/TVCG.2025.3567121 2
- [60] I. E. Toubal, Y. Duan, and D. Yang. Deep learning semantic segmentation for high-resolution medical volumes. In *Proceedings of IEEE Applied Imagery Pattern Recognition Workshop*, pp. 1–9, 2020. doi: 10.1109/AIPR50011.2020.9425041 2
- [61] F.-Y. Tzeng, E. B. Lum, and K.-L. Ma. An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):273–284, 2005. doi: 10.1109/TVCG.2005.38 2, 11
- [62] C. Wang and J. Han. DL4SciVis: A state-of-the-art survey on deep learning for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 29(8):3714–3733, 2023. doi: 10.1109/TVCG.2022.3167896 1
- [63] S. Weiss, M. Chu, N. Thuerey, and R. Westermann. Volumetric isosurface rendering with deep learning-based super-resolution. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):3064–3078, 2021. doi: 10.1109/TVCG.2019.2956697 2
- [64] S. Weiss, P. Hermüller, and R. Westermann. Fast neural representations for direct volume rendering. *Computer Graphics Forum*, 41(6):196–211, 2022. doi: 10.1111/CGF.14578 2
- [65] S. Weiss, M. Isik, J. Thies, and R. Westermann. Learning adaptive sampling and reconstruction for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(7):2654–2667, 2022. doi: 10.1109/TVCG.2020.3039340 2
- [66] W. Widanagamaachchi, C. Christensen, V. Pascucci, and P.-T. Bremer. Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In *Proceedings of IEEE Symposium on Large Data Analysis and Visualization*, 2012. doi: 10.1109/LDAV.2012.6378962 3, 11
- [67] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang. 4D Gaussian splatting for real-time dynamic scene rendering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 20310–20320, 2024. doi: 10.1109/CVPR52733.2024.01920 2, 3
- [68] Q. Wu, D. Bauer, M. J. Doyle, and K.-L. Ma. Interactive volume visualization via multi-resolution hash encoding based neural representation. *IEEE Transactions on Visualization and Computer Graphics*, 30(8):5404–5418, 2024. doi: 10.1109/TVCG.2023.3293121 2
- [69] S. W. Wurster, T. Xiong, H.-W. Shen, H. Guo, and T. Peterka. Adaptively placed multi-grid scene representation networks for large-scale data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):965–974, 2024. doi: 10.1109/TVCG.2023.3327194 2
- [70] M. Yang, K. Tang, and C. Wang. Meta-INR: Efficient encoding of volumetric data via meta-learning implicit neural representation. In *Proceedings of IEEE Pacific Visualization Conference (Visualization Notes)*, pp. 246–251, 2025. doi: 10.1109/PacificVis64226.2025.00030 1
- [71] Z. Yang, X. Gao, W. Zhou, S. Jiao, Y. Zhang, and X. Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 20331–20341, 2024. doi: 10.1109/CVPR52733.2024.01922 2, 3, 5
- [72] Z. Yang, H. Yang, Z. Pan, and L. Zhang. Real-time photorealistic dynamic scene representation and rendering with 4D Gaussian splatting. In *Proceedings of International Conference on Learning Representations*, 2024. 2, 3, 5
- [73] S. Yao, J. Han, and C. Wang. GMT: A deep learning approach to generalized multivariate translation for scientific data analysis and visualization. *Computers & Graphics*, 112:92–104, 2023. doi: 10.1016/J.CAG.2023.04.002 1
- [74] S. Yao, Y. Lu, and C. Wang. ViSNeRF: Efficient multidimensional neural radiance field representation for visualization synthesis of dynamic volumetric scenes. In *Proceedings of IEEE Pacific Visualization Conference*, pp. 235–245, 2025. doi: 10.1109/PacificVis64226.2025.00029 1, 2, 3, 5
- [75] S. Yao and C. Wang. ReVolVE: Neural reconstruction of volumes for visualization enhancement of direct volume rendering. *Computers & Graphics*, 2025. Accepted. 2

- [76] M. Ye, M. Danelljan, F. Yu, and L. Ke. Gaussian Grouping: Segment and edit anything in 3D scenes. In *Proceedings of European Conference on Computer Vision*, pp. 162–179, 2024. doi: [10.1007/978-3-031-73397-0_10](https://doi.org/10.1007/978-3-031-73397-0_10) 2, 4
- [77] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 5752–5761, 2021. doi: [10.1109/ICCV48922.2021.00570](https://doi.org/10.1109/ICCV48922.2021.00570) 2
- [78] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. doi: [10.1109/CVPR.2018.00068](https://doi.org/10.1109/CVPR.2018.00068) 6