4D Scaffold Gaussian Splatting with Dynamic-Aware Anchor Growing for Efficient and High-Fidelity Dynamic Scene Reconstruction

Woong Oh Cho, In Cho, Seoha Kim, Jeongmin Bae, Youngjung Uh, Seon Joo Kim
Yonsei University
{wocho, join, hailey07, jaymin.bae, yj.uh, seonjookim}@yonsei.ac.kr

Abstract

Modeling dynamic scenes through 4D Gaussians offers high visual fidelity and fast rendering speeds, but comes with significant storage overhead. Recent approaches mitigate this cost by aggressively reducing the number of Gaussians. However, this inevitably removes Gaussians essential for high-quality rendering, leading to severe degradation in dynamic regions. In this paper, we introduce a novel 4D anchor-based framework that tackles the storage cost in different perspective. Rather than reducing the number of Gaussians, our method retains a sufficient quantity to accurately model dynamic contents, while compressing them into compact, grid-aligned 4D anchor features. Each anchor is processed by an MLP to spawn a set of neural 4D Gaussians, which represent a local spatiotemporal region. We design these neural 4D Gaussians to capture temporal changes with minimal parameters, making them well-suited for the MLP-based spawning. Moreover, we introduce a dynamic-aware anchor growing strategy to effectively assign additional anchors to under-reconstructed dynamic regions. Our method adjusts the accumulated gradients with Gaussians' temporal coverage, significantly improving reconstruction quality in dynamic regions. Experimental results highlight that our method achieves state-of-the-art visual quality in dynamic regions, outperforming all baselines by a large margin with practical storage costs.

Introduction

Reconstructing dynamic scenes from multi-view videos has received significant attention due to its wide applications. Beyond methods based on neural radiance fields (NeRFs) (Li et al. 2022; Wang et al. 2022; Attal et al. 2023), 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) has become a major approach for dynamic scene reconstruction, due to its ability to render high-quality novel views in real-time. The two main categories of this approach are 1) modeling temporal changes as deformations of canonical 3D Gaussians (Yang et al. 2023; Bae et al. 2024; Jiawei et al. 2024) and 2) employing 4D Gaussians to approximate a scene's 4D volumes (Yang et al. 2024; Li et al. 2023; Lee et al. 2024c,a).

Directly optimizing 4D Gaussians offers higher visual quality and faster rendering than deformation-based methods. By modeling temporal changes through multiple 4D Gaussians, each covering a certain time range, they effectively capture complex dynamic regions without the heavy computation cost of deformation fields. However, this approach accompanies a large number of 4D Gaussians, which subsequently leads to substantial storage overhead–often exceeding 6GB for a 10-second video (see Figure 1, left).

Several approaches have attempted to address the storage cost by reducing the number of Gaussians. They achieve this by more complicated motion modeling (Lee et al. 2024b), aggressive pruning (Li et al. 2023), or motion interpolation (Lee et al. 2024a). While these methods effectively reduce storage, they also inevitably remove Gaussians in dynamic regions, sacrificing the expressiveness needed to represent complex temporal changes. As a result, these methods often suffer from quality degradation in dynamic components.

In this paper, we introduce a 4D anchor-based framework that addresses the storage overhead from a different perspective. Instead of reducing the number of Gaussians–which is crucial for maintaining the expressiveness–our method maintains a sufficient number to render dynamic regions with high visual quality. This is achieved by representing dynamic scenes through structured anchor features (Lu et al. 2024), aligned with a sparse 4D grid. Each anchor holds a compact feature vector to model a local spatiotemporal region. This feature is processed by shared multi-layer perceptrons (MLP) to output properties of neural Gaussians.

Considering the limited capacity of the shared shallow MLP, we propose a compact parametrization of the neural Gaussians that effectively captures temporal changes. This includes two key modelings: linear motion to model the time-varying 3D position, and a generalized Gaussian function to model temporal opacity, which is better suited for capturing sudden appearance changes. This design enables our framework to effectively represent complex dynamic trajectories through a sequence of linear segments.

While the anchor-based scheme effectively reduces storage, a naïve extension of 3D scaffolding (Lu et al. 2024) fails to accurately capture dynamic regions (see Figure 1, Scaff-naive). The limitation arises from the anchor growing strategy used in (Lu et al. 2024), which is originally designed for static scenes. In our 4D framework, each anchor and its neural Gaussians are active only within a specific time range, while the anchor growing strategy accumulates gradients across all frames without considering this coverage.

Figure 1: Dynamic region quality comparison. **(left)** Comparison of dynamic region quality (y-axis) versus storage cost (x-axis) on N3DV. **(right)** Results on the *flame_salmon* scene from N3DV. Scaff-naive refers to the naive anchor-based model without our neural Gaussian design and dynamic-aware anchor growing. Our method significantly outperforms all baselines with efficient storage usage, while other methods either exhibit degraded quality in dynamic regions or excessive storage costs.

As a result, dynamic regions appearing in only a few frames receive less gradient signals, leading to under-reconstructed dynamic components in the final scene.

We present a dynamic-aware anchor growing strategy to properly allocate new anchors to under-reconstructed dynamic regions. Our approach adjusts the accumulated gradients based on each Gaussian's temporal coverage. We adjust dynamic Gaussians with shorter coverage to have larger gradient, compensating for the penalty caused by their short appearance. This modification encourages new anchors to be allocated more in under-reconstructed dynamic regions, achieving substantial improvement in reconstruction quality.

Our anchor-based framework, coupled with the dynamic-aware anchor growing strategy, represents complex dynamic components in high-quality while addressing storage costs. We validate our approach through extensive experiments conducted on the N3DV (Li et al. 2022) and Technicolor (Sabater et al. 2017) datasets. Notably, our method outperforms all baselines by a large margin with practical storage overhead, as illustrated in Figure 1.

Related Work

Efficient 3D Gaussians. In recent years, 3D Gaussian Splatting (3DGS) has attracted significant attention for achieving real-time rendering by representing scenes with 3D Gaussian primitives and introducing a tile-based rasterizer. For photorealistic rendering quality, 3DGS requires a significant number of 3D Gaussians, which increases storage costs. To address this problem, previous works remove unnecessary Gaussians that do not harm rendering quality (Fan et al. 2023; Girish, Gupta, and Shrivastava 2023), replace Gaussians or their parameters with efficient representations (Niedermayr, Stumpfegger, and Westermann 2024; Papantonakis et al. 2024) and compress Gaussian parameters using existing compression techniques such as entropy coding and image compression (Niedermayr, Stumpfegger, and Westermann 2024; Chen et al. 2025; Xie et al. 2025). In this paper, we address compressing Gaussians in spatiotemporal space, which have been relatively unexplored compared to 3D Gaussians.

Dynamic 3D Gaussians. Two main approaches are proposed to extend 3DGS (Kerbl et al. 2023) into dynamic scene reconstruction. The first approach involves deforming 3D Gaussians along with temporal changes (Yang et al. 2023; Wu et al. 2023; Bae et al. 2024; Shaw et al. 2024; Jiawei et al. 2024; Kwak et al. 2025; Yun et al. 2025). These deformable 3D Gaussians offer the advantage of compact storage requirements but exhibit relatively slow rendering speeds and low visual quality. In contrast, the other approach directly employs 4D Gaussians in the spatio-temporal domain. 4DGS (Yang et al. 2024) demonstrate superior visual quality and faster rendering speeds, but suffer from higher storage requirements. Although some works (Li et al. 2023; Lee et al. 2024a,c) improve storage efficiency using fewer Gaussians, they tend to focus on the holistic scene, thereby neglecting the quality of dynamic areas. To address this, we take an alternative approach: reducing storage while maintaining quality by preserving the Gaussian count.

Feature-based neural rendering. Recently, a growing trend in scene reconstruction has been to integrate neural features as additional inputs to enhance model performance. For instance, there have been attempts to extract features from source views and utilize them for novel view synthesis, enabling few-view reconstruction (Yu et al. 2021; Chen et al. 2021) or view interpolation through transformers (Wang et al. 2021; Reizenstein et al. 2021; T et al. 2023). In 3DGS studies, Compact3DGS (Deng et al. 2024) uses a hash grid instead of per-Gaussian SH coefficients and STG (Li et al. 2023) renders features into RGB via shallow MLPs. Some works generate Gaussian attributes from a multi-level tri-plane (Wu and Tuytelaars 2024) and predict the attributes of local 3D Gaussians from anchor features (Lu et al. 2024). We introduce a 4D anchor-based framework that includes dynamic linear motion and temporal opacity derived from a generalized Gaussian distribution, considering real-world dynamics.

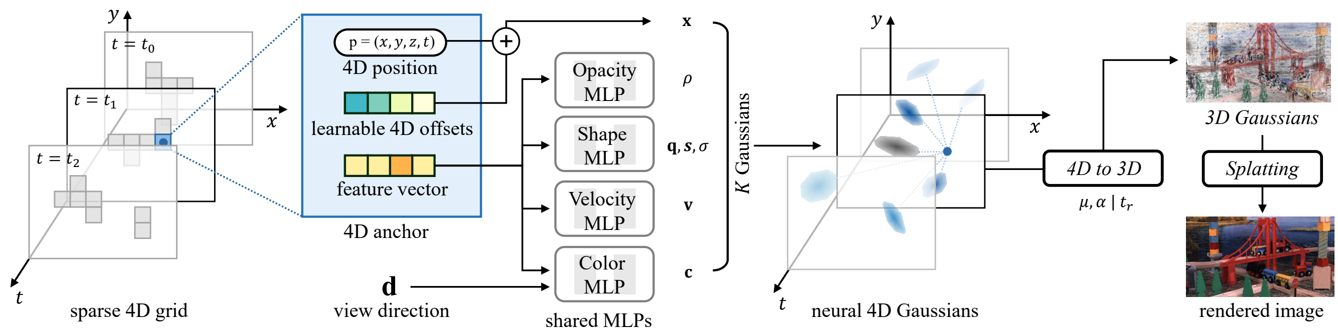

Figure 2: Proposed 4D anchor-based framework. We begin with a sparse 4D grid of anchors, each defined by a unique 4D spatiotemporal position $\mathbf{p}$ and learnable offsets. Shared MLPs utilize these anchors to generate neural 4D Gaussians, capturing dynamic appearance changes with view direction $\mathbf{d}$. These 4D Gaussians are then projected to 3D Gaussians at specific time t_r for rendering, using a 3D Gaussian splatting pipeline to produce the final rendered image. We omit the z-axis for simplicity.

Method

We reconstruct dynamic 3D scenes through a sparse, grid-aligned 4D anchor grid. Each anchor holds a compressed feature vector that specifies nearby 4D Gaussians. The dynamic-aware anchor growing strategy encourages new anchors to be closely placed at under-reconstructed dynamic regions. Furthermore, we design each 4D Gaussian to move along a line segment, and define the temporal opacity function which fits a piecewise persistent period. Through these components, our framework employs a sufficient number of Gaussians to achieve high-quality reconstruction of dynamic regions, while the storage overhead is significantly reduced via compressed anchor features.

4D Anchor-Based Framework

The overview of our method is illustrated in Figure 2. Our method begins with a set of sparse 4D anchor points, each of which has a unique, grid-aligned spatiotemporal 4D position $\mathbf{p} \in \mathbb{R}^4$ with a feature vector $\mathbf{f} \in \mathbb{R}^C$. We leverage shared MLPs to produce K neural 4D Gaussians from these anchor features. Corresponding 3D Gaussians are computed from these 4D Gaussians to render a frame at timestep t.

Initializing 4D anchors. Similar to 3D Scaffold-GS (Lu et al. 2024), we utilize static point clouds to initialize 4D anchor points. We first obtain the static point cloud from multi-view frames at a certain timestep t_0 using Structure-from-Motion (SfM) (Schönberger and Frahm 2016). The 4D positions of anchors are initialized from a set of voxels $\mathbf{V} \in \mathbb{R}^{N\times 3}$ obtained from this point cloud:

$$\mathbf{p} = (x_v, y_v, z_v, t_0), \quad \forall v \in \mathbf{V}, \tag{1}$$

$$\mathbf{x}_k = \mathbf{p} + \Delta\mathbf{x}_k, \quad k \in \{1, 2, ..., K\}, \tag{2}$$

where (x_v, y_v, z_v) is the center coordinates of the voxel v. Each anchor point also accompanies two learnable parameters: a feature vector $\mathbf{f}$, and 4D offsets $\Delta\mathbf{x}_k \in \mathbb{R}^4$ for determining properties of K neural 4D Gaussians. 4D Gaussian position $\mathbf{x}$ is determined by anchor position and offset.

Neural 4D Gaussians. We leverage shared MLPs and learnable parameters of the 4D anchors to generate K neural 4D Gaussians from each anchor. Specifically, shared MLPs take the anchor feature $\mathbf{f}$ and yield properties of K neural Gaussians, which include base opacity $\rho \in \mathbb{R}$, quaternion $\mathbf{q} \in \mathbb{R}^4$ and scaling $\mathbf{s} \in \mathbb{R}^3$ for the covariance matrix, view-dependent color $\mathbf{c} \in \mathbb{R}^3$. Our shared MLPs also produce a temporal scale $\sigma \in \mathbb{R}$ to compute our temporal opacity, and the neural velocity $\mathbf{u} \in \mathbb{R}^3$. Note that the color MLP additionally takes the view direction $\mathbf{d} \in \mathbb{R}^3$ as inputs for view-dependent modeling.

Rendering neural Gaussians. To render the neural 4D Gaussians, we compute 3D Gaussian parameters at time t_r from our time-varying properties. For the k-th Gaussian of the anchor $\mathbf{p}$, the center $\mu_k \in \mathbb{R}^3$ and the opacity $\alpha_k \in \mathbb{R}$ of the corresponding 3D Gaussian at time t_r are derived as

$$\mu_k = \mathbf{x}_k^{xyz} + h(t_r, \mathbf{x}_k^t, \mathbf{u}), \tag{3}$$

$$\alpha_k = \rho_k \cdot g(t_r, \mathbf{x}_k^t, \sigma_k), \tag{4}$$

where $\mathbf{x}_k^{xyz} \in \mathbb{R}^3$ and $\mathbf{x}_k^t \in \mathbb{R}$ are the spatial and the temporal position of the 4D Gaussian. $h(\cdot)$, $g(\cdot)$ model time-varying values of the positions and the opacity with the neural velocity and temporal opacity function, which will be further described in the following section.

After deriving 3D Gaussians at time t_r, we utilize the existing 3D Gaussian splatting pipeline (Kerbl et al. 2023) to render the frames. Same as 3D Scaffold-GS (Lu et al. 2024), only Gaussians within the view frustum and having opacity higher than the threshold are passed to the renderer.

Compact Parametrization of 4D Gaussians

The properties of neural 4D Gaussians are predicted by a shallow shared MLP. Increasing the number of Gaussian properties complicates the optimization process. To this end, we propose a compact parametrization of 4D Gaussians that captures temporal changes as a set of linear segments. Our neural Gaussian design involves two key modelings. First, we represent time-varying spatial positions as linear motions. Second, we employ a generalized Gaussian function to model temporal opacity, which effectively captures sudden appearance changes with a single parameter.

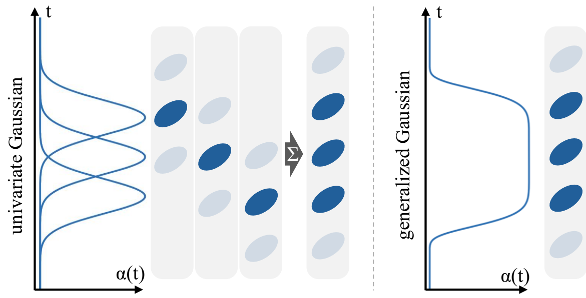

Figure 3: Illustration of the modified temporal opacity. To reconstruct the sudden and continuous appearance of an object, the univariate Gaussian function requires the sum of multiple Gaussians, while the generalized Gaussian achieves the same expressiveness with only a single Gaussian.

Linear motion. As described in Eq. (3), the time-varying 3D position of the 3D Gaussian $\mu_k \in \mathbb{R}^3$ is determined by the time-varying function $h(\cdot)$. We formulate the $h(\cdot)$ as a linear motion based on the neural velocity $\mathbf{u}$:

$$h(t, \mathbf{x}_k^t, \mathbf{u}) = (t - \mathbf{x}_k^t)\mathbf{u}. \tag{5}$$

This formulation simplifies the temporal slicing of Gaussian positions proposed in 4DGS (Lee et al. 2024a), using only 3 parameters per Gaussian. Despite its simplicity, a set of Gaussians with linear motion effectively captures the temporal dynamics of scene elements with piecewise linear approximation, particularly when combined with our modified temporal opacity.

Modified temporal opacity. Previous 4D Gaussian methods (Yang et al. 2024; Li et al. 2023) typically adopt the univariate Gaussian function to make Gaussians appear within a specific time range. However, this formulation lacks the capacity to model abrupt changes in real-world scene elements, limiting the expressiveness of individual Gaussians.

We reformulate the temporal opacity function based on a generalized Gaussian distribution. As shown in Figure 3, this offers steeper derivatives at the beginning and the end, making it better suited to fit piecewise persistent periods. Our time-varying opacity $g(\cdot)$ described in Eq. (4) is formulated as

$$g(t, \mathbf{x}_k^t, \sigma_k) = exp(-(|t - \mathbf{x}_k^t|/\sigma_k)^\beta), \tag{6}$$

where β is a tunable hyperparameter. In practice, we set the inverse sigma as a learnable parameter, as it leads to more stable training. We also model $\beta = 2\beta'$ and β' as a hyperparameter, which removes the $|\cdot|$ in the above equation.

While the issue of temporal opacity is also explored in (Lee et al. 2024a), we show that efficiently representing temporal coverage with fewer parameters is crucial for the performance of the anchor-based scheme.

Dynamic-Aware Anchor Growing

Growing new anchors to under-reconstructed dynamic regions is a critical factor for high-quality results. The direct application of previous anchor growing to the dynamic reconstruction fails to identify such regions, as it neglects the temporal coverage of the Gaussians.

The previous anchor growing strategy (Lu et al. 2024) designed for static scenes gathers the gradients as a mean over N iterations:

$$\nabla_g = \frac{\sum^N \|\nabla_{2\text{D}}\|}{N}. \tag{7}$$

With this strategy, dynamic regions appearing in a short period will have lower ∇_g, as these regions will be penalized by the denominator N regardless of their actual errors.

To address this, we propose a dynamic-aware anchor growing operation. The core of our anchor growing is the computation of the accumulated gradients of each 4D Gaussian ∇_g. We formulate ∇_g as a weighted sum of the gradient over N iterations:

$$\nabla_g = \frac{\sum^N w(\alpha', \sigma)\|\nabla_{2\text{D}}\|}{\sum^N w(\alpha', \sigma)}, \tag{8}$$

where $\nabla_{2\text{D}}$ is the 2D position gradient of the Gaussian, and $\alpha' = g(t_r, \mathbf{x}^t, \sigma)$ is the time-variant component of the Gaussian opacity. We define the weight term w as a function of α and σ, with γ as a hyperparameter:

$$w(\alpha', \sigma) = \alpha'(1/\sigma)^\gamma. \tag{9}$$

We collect the accumulated gradients of all Gaussians and then voxelize them. New anchors are placed at the centers of the voxels having $\nabla_{2\text{D}}$ higher than each threshold.

The dynamic-aware anchor growing strategy differs from the original anchor growing operation (Lu et al. 2024) in two key aspects. First, gradients are accumulated only when the Gaussian's temporal opacity is greater than zero (*i.e.*, when it is activated). It allows our method to accurately gather the gradients of 4D Gaussians placed at the regions where dynamic scene elements only appear in a short period. Second, the temporal coverage σ directly influences to the anchor growing operation. This encourages regions with a short temporal coverage to receive stronger gradients, making it easier for anchors to grow in those areas. By accumulating gradients as a weighted sum, our method effectively places new anchors to under-reconstructed dynamic regions.

Training and Implementation Details

Training objective. Both learnable anchor parameters and MLP weights are jointly optimized through the rendering loss. We employ $\mathcal{L}_1$ with SSIM loss $\mathcal{L}_{\text{SSIM}}$ and volume regularization $\mathcal{L}_{\text{vol}}$ following the 3D Scaffold-GS (Lu et al. 2024). The full training objective with weighting coefficients $\lambda_{\text{SSIM}} = 0.2$ and $\lambda_{\text{vol}} = 0.01$ is

$$\mathcal{L} = (1 - \lambda_{\text{SSIM}})\mathcal{L}_1 + \lambda_{\text{SSIM}}\mathcal{L}_{\text{SSIM}} + \lambda_{\text{vol}}\mathcal{L}_{\text{vol}}. \tag{10}$$

Implementation details. Similar to 4DGS, our model is trained for 120K iterations with a single batch, which takes approximately 3 hours on a single NVIDIA A6000 GPU. We set $\beta = 2$ and $\gamma = 1$ for our modified temporal opacity model and dynamic-aware densification. For 4D voxel grid size, the spatial grid size is set to 0.001 and the temporal grid size is 0.0333 for the N3DV. For the Technicolor dataset, these values are 0.0001 and 0.001, respectively. The entire time range is scaled to the interval [0, 10]. We set

Model	Dynamic region PSNR↑	SSIM↑	LPIPS↓	#Gaussians	Full region PSNR↑	SSIM↑	LPIPS↓	#Gaussians	Computational cost FPS↑	Storage↓	Training time↓
4DGaussian (2023)	25.33	0.833	0.166	-	30.71	0.935	0.056	192 K	51.9	57 MB	50 m
E-D3DGS (2024)	26.92	0.884	0.112	-	<u>32.04</u>	**0.951**	**0.034**	180 K	74.5	66 MB	1 h 52 m
Grid4D (2024)	26.65	0.877	0.129	-	31.92	<u>0.949</u>	<u>0.039</u>	202 K	127.1	48 MB	1 h 20 m
STG (2023)	25.84	0.860	0.127	44 K	31.24	0.941	0.051	434 K	125.9	63 MB	1 h 8 m
4DGS (2024)	<u>27.65</u>	<u>0.907</u>	<u>0.075</u>	3306 K	**32.14**	0.947	0.047	3333 K	61.4	6194 MB	9 h 30 m
4DGS† (1GB)	26.70	0.877	0.123	557 K	31.59	0.943	0.052	581 K	152.7	1080 MB	3 h 37 m
Ex4DGS (2024a)	26.33	0.874	0.121	52 K	32.01	0.947	0.048	268 K	91.9	115 MB	1 h 6 m
Scaff-naive	24.79	0.811	0.199	206 K	31.21	0.942	0.053	732 K	159.7	83 MB	2 h 57 m
Ours-light	27.50	0.902	0.076	181 K	31.54	0.944	0.045	314 K	148.2	90 MB	2 h 51 m
Ours	**28.86**	**0.927**	**0.054**	533 K	32.03	0.947	0.041	775 K	129.9	149 MB	3 h 6 m

Table 1: Quantitative results on N3DV dataset. 4DGS† refers to the result with 1GB storage, for fair comparisons.

$K = 10$ following (Lu et al. 2024). After training, we prune the invalid anchor points of which all Gaussians have negative opacity. To improve inference speed, we cache the time-invariant and view-invariant outputs of MLPs. We provide more implementation details and hyperparameter analysis in the supplements.

Experiments

In this section, we first evaluate our method through the comparisons with several state-of-the-art baselines for dynamic scene reconstruction. Then we conduct ablations and analysis to explore the effectiveness of our main features. Our code will be made publicly available.

Datasets. We evaluate our method on two representative real-world datasets, which are Neural 3D Video (N3DV) (Li et al. 2022) and Technicolor (Sabater et al. 2017).

- **Neural 3D Video dataset (N3DV)** comprises 17 to 21 synchronized videos of six scenes, with each video containing 300 frames. Following previous works, we downsample the resolution to 1352×1014 and use the first camera as the test view. We exclude *cam13* for the *coffee_martini* scene due to synchronization issues.
- **Technicolor dataset** contains 16 synchronized videos across five scenes, with each video comprising 50 frames. We retain the original resolution of 2048×1088 and use *cam10* as the test view.

Baselines. We choose the following state-of-the-art competitors: 4DGaussian (Wu et al. 2023), E-D3DGS (Bae et al. 2024) and Grid4D (Jiawei et al. 2024) representing deformation-based methods, and 4DGS (Yang et al. 2024), STG (Li et al. 2023), C3DGS (Lee et al. 2024c) and Ex4DGS (Lee et al. 2024a) representing 4D Gaussians. We reproduce them using the official code to report their performance. We also introduce Scaff-naive as an additional baseline. This is an anchor-based model without our neural Gaussian design and the anchor growing, which instead adopts linear motion and the temporal opacity modeling used in (Yang et al. 2024). We present a variant of our model, Ours-light, which employs a larger voxel size to achieve a more reduced storage footprint. For fair comparisons, we report STG trained using 300 frames and initial SfM points derived from the first frames, same as other methods.

Metrics. To assess the reconstruction quality, we measure peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), and LPIPS (Zhang et al. 2018) of the rendered images. To compute LPIPS, we follow (Bae et al. 2024) and employ AlexNet (Krizhevsky, Sutskever, and Hinton 2012). To measure the visual quality in dynamic regions, we use a combined mask of Global-Median and Temporal-Difference (Li et al. 2022), binarized with a threshold of 50. We assess storage efficiency by calculating the total size of the output files, including MLP weights, as well as Gaussian and anchor parameters. In addition to storage size, we also report the number of Gaussians in both dynamic and full regions. FPS and training time are measured on a single NVIDIA A6000 GPU, using the *flame_salmon* and *Fabien* scenes from the N3DV and the Technicolor, respectively.

Experimental Results

N3DV. Table 1 presents the quantitative results on the N3DV. Our model outperforms all baselines in terms of visual quality in dynamic regions, while also delivering competitive results in full regions. While our method employs more Gaussians to achieve high-quality results, it maintains efficient storage overhead and FPS, thanks to the anchor-based compression scheme. Our storage-efficient variant (Ours-light) achieves the second-best storage efficiency among 4D Gaussian methods, while also surpassing other efficient baselines in visual quality. Scaff-naive yields degraded results, indicating that the naïve extension of the scaffolding is insufficient to accurately reconstruct dynamic 3D scenes. 4DGS often models static components through multiple dynamic Gaussians, resulting in an excessive number of Gaussians and storage costs. Although our method does not leverage additional techniques to model far-background areas (Li et al. 2023; Yang et al. 2024), it still exhibits plausible reconstruction quality in both static and full regions.

The effectiveness of our method is also demonstrated in qualitative results. As shown in Figure 4, our model effectively represents dynamic regions with a sufficient number

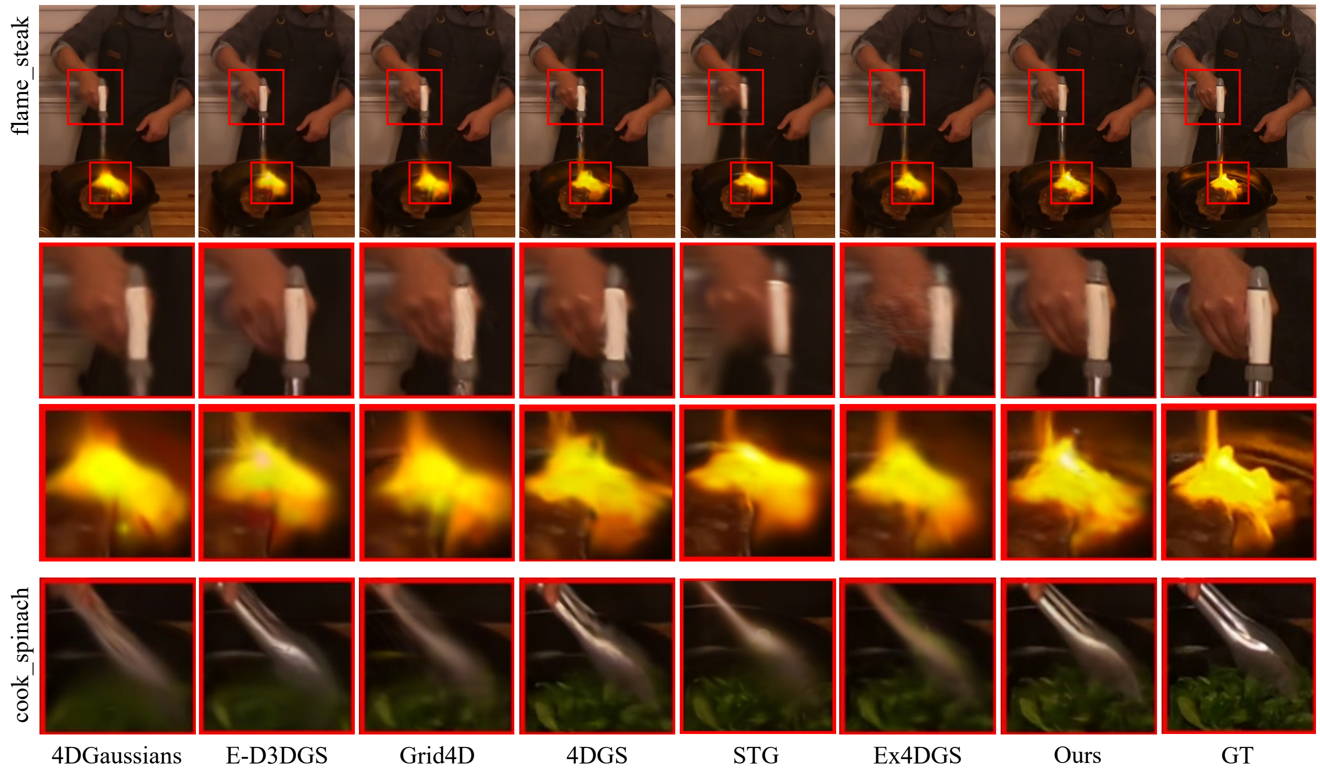

Figure 4: Qualitative comparisons on N3DV dataset. Our method achieves high-fidelity in both static and dynamic regions.

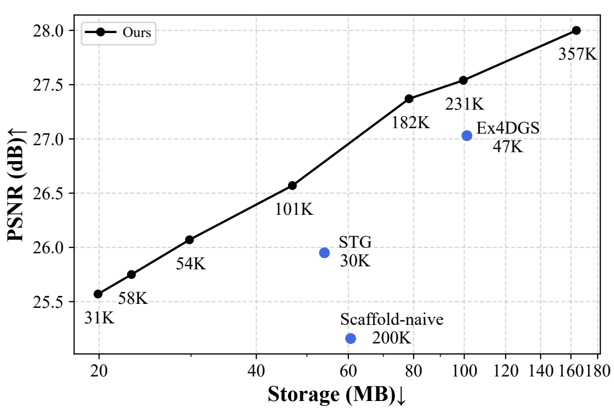

Figure 5: Quality-storage trade-off. The results on the *flame_steak* scene of the N3DV, along with the number of Gaussians, are reported. We increase the number of employed Gaussians of our model by adjusting the anchor grid size.

of Gaussians, thereby achieving high-quality dynamic reconstruction. In contrast, other efficient baselines struggle to capture complex scene dynamics (see the boxed regions). Please refer to the supplements for more visual comparisons, including uncropped results and videos.

Technicolor. We deliver the quantitative comparisons on the Technicolor dataset in Table 2. Our model achieves state-of-the-art performance across all visual quality evaluation metrics. This is further exemplified by our storage-efficient model, Ours-light. Despite its compact size of approximately 100 MB, Ours-light delivers competitive visual quality on par with 4DGS, highlighting the effectiveness of the proposed anchor-based scheme. Qualitative comparisons on the Technicolor dataset are provided in the supplements.

Analysis and Ablation Study

To provide deeper insights of our main features, we conduct the ablations and analysis in dynamic regions on the N3DV.

Quality-storage trade-off. We first analyze the quality-storage trade-off of our model by gradually increasing the number of employed Gaussians. This is controlled by adjusting the voxel size of the anchor grid. As presented in Figure 5, the performance of our model is improved with more Gaussians, confirming the necessity of using a sufficient quantity. Notably, our method outperforms other storage-efficient baselines when using similar or smaller storage costs. Thanks to the anchor-based scheme, our method employs 3.37× or 7.0× more Gaussians and achieves higher visual quality, yet still maintains a lower storage overhead.

Understanding the anchor growing. We then ablate the proposed anchor growing strategy. As reported at the bottom of Table 3, the dynamic-aware anchor growing leads to noticeable improvement in visual quality. By properly allocating new anchors to under-reconstructed dynamic regions,

Model	Dynamic region PSNR↑	SSIM↑	LPIPS↓	#Gaussians	Full region PSNR↑	SSIM↑	LPIPS↓	#Gaussians	Computational cost FPS↑	Storage↓	Training time↓
4DGaussian (2023)	23.99	0.709	0.280	-	29.62	0.843	0.176	268 K	23.4	72 MB	31 m
E-D3DGS (2024)	29.39	0.886	0.148	-	33.38	0.907	0.100	212 K	60.8	77 MB	2 h 34 m
STG (2023)	28.35	0.869	0.174	71 K	33.33	0.912	0.096	132 K	149.0	30 MB	1 h 24 m
4DGS (2024)	<u>31.91</u>	<u>0.930</u>	**0.093**	5597 K	33.30	0.910	0.095	5759 K	141.5	10699 MB	6 h 11 m
4DGS† (1GB)	29.18	0.890	0.144	538 K	28.72	0.856	0.166	591 K	156.6	1098 MB	3 h 36 m
C3DGS (2024c)	27.21	0.843	0.209	87 K	32.52	0.901	0.110	122 K	178.7	18 MB	1 h 9 m
Ex4DGS (2024a)	30.58	0.918	<u>0.109</u>	120 K	33.40	0.914	0.088	426 K	78.4	140 MB	1 h 53 m
Scaff-naive	28.00	0.853	0.195	721 K	33.34	0.915	0.081	2056 K	127.0	129 MB	1 h 59 m
Ours-light	30.95	0.916	0.117	186 K	<u>33.97</u>	<u>0.923</u>	<u>0.074</u>	480 K	131.4	108 MB	1 h 40 m
Ours	**31.94**	**0.932**	**0.093**	1165 K	**34.11**	**0.925**	**0.072**	1272 K	110.8	278 MB	2 h 30 m

Table 2: Quantitative results on Technicolor dataset. For fair comparison, 4DGS† result of using 1GB storage is reported.

DA	Motion	Opacity	PSNR↑	LPIPS↓	Storage↓
✓	Polynomial	Ours	27.82	0.070	257 MB
✓	Linear	4DGS	27.93	0.065	195 MB
✓	Linear	Ex4DGS	28.34	0.056	413 MB
	Linear	Ours	25.77	0.153	180 MB
✓	Linear	Ours	29.57	0.050	149 MB

Table 3: Ablation results on the main components. "DA" refers to the dynamic-aware anchor growing, "Motion" and "Opacity" for each modeling of the neural Gaussian design.

our model accurately represents dynamic scenes with practical storage overhead.

To further explore the effects of the anchor growing strategy, we provide visual comparisons in Figure 6. The previous method fails to accurately collect gradients in dynamic regions due to their short appearing periods. This leads to a lack of anchors in dynamic regions, resulting in degraded visual quality. In contrast, our method successfully accumulates higher gradients in these dynamic regions (see the red-boxed parts in Figure 6). Consequently, the anchors generated by our method more accurately represent dynamic regions at each timestep. These results demonstrate that our anchor-growing strategy effectively captures under-reconstructed dynamic regions, playing a critical role in achieving high-quality reconstruction.

Effects of neural Gaussian design. We first evaluate our neural design by replacing each modeling with the formulation in previous works. Specifically, we replace the linear motion with the polynomial trajectory (Li et al. 2023), and replace our modified temporal opacity with the one used in 4DGS (Yang et al. 2024) and Ex4DGS (Lee et al. 2024a). As shown in Table 3, our model with the proposed components achieves the best results. Our compact parametrization enhances the expressiveness of each Gaussian with minimal parameters, improving both visual quality and efficiency.

Conclusion

Our framework employs a sufficient number of Gaussians to capture complex dynamic regions, while addressing the storage overhead through the anchor-based compression. The dynamic-aware anchor growing and the neural Gaussian design lead to a substantial improvement. Experimental results support the validity of our method, achieving state-of-the-art visual quality and practical storage costs.

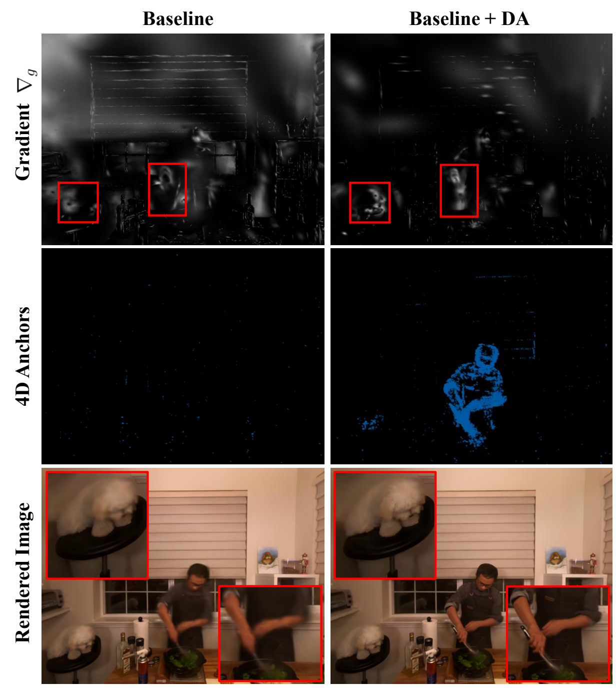

Figure 6: Anchor growing analysis. Top-to-bottom: accumulated gradients at 5000th training iteration, final 4D anchors at 106 frame, and the rendering results. Our dynamic-aware anchor growing effectively accumulates gradients in under-reconstructed dynamic regions.

Limitations and future work. Since our method is currently designed for multi-view video datasets, applying it to monocular videos can introduce additional challenges. Like other methods, our method still suffers from reconstructing elements that appear very shortly (1 or 2 frames). Resolving this challenge can be an interesting future work.

Acknowledgements

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) for the Artificial Intelligence Graduate School Program, Yonsei University (No. RS-2020-II201361), the Global AI Frontier Lab International Collaborative Research (No. RS-2024-00469482 and RS-2024-00509279), and the AI Research Hub Project (No. RS-2024-00457882).

References

Attal, B.; Huang, J.-B.; Richardt, C.; Zollhoefer, M.; Kopf, J.; O'Toole, M.; and Kim, C. 2023. HyperReel: High-fidelity 6-DoF video with ray-conditioned sampling. In *CVPR*, 16610–16620.

Bae, J.; Kim, S.; Yun, Y.; Lee, H.; Bang, G.; and Uh, Y. 2024. Per-Gaussian Embedding-Based Deformation for Deformable 3D Gaussian Splatting. In *ECCV*.

Chen, A.; Xu, Z.; Zhao, F.; Zhang, X.; Xiang, F.; Yu, J.; and Su, H. 2021. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 14124–14133.

Chen, Y.; Wu, Q.; Lin, W.; Harandi, M.; and Cai, J. 2025. Hac: Hash-grid assisted context for 3d gaussian splatting compression. In *ECCV*, 422–438. Springer.

Deng, T.; Chen, Y.; Zhang, L.; Yang, J.; Yuan, S.; Liu, J.; Wang, D.; Wang, H.; and Chen, W. 2024. Compact 3d gaussian splatting for dense visual slam. *arXiv preprint arXiv:2403.11247*.

Fan, Z.; Wang, K.; Wen, K.; Zhu, Z.; Xu, D.; and Wang, Z. 2023. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*.

Girish, S.; Gupta, K.; and Shrivastava, A. 2023. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. *arXiv preprint arXiv:2312.04564*.

Jiawei, X.; Zexin, F.; Jian, Y.; and Jin, X. 2024. Grid4D: 4D Decomposed Hash Encoding for High-Fidelity Dynamic Scene Rendering. *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG*, 42(4).

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25.

Kwak, S.; Kim, J.; Jeong, J. Y.; Cheong, W.-S.; Oh, J.; and Kim, M. 2025. MoDec-GS: Global-to-Local Motion Decomposition and Temporal Interval Adjustment for Compact Dynamic 3D Gaussian Splatting. In *CVPR*.

Lee, J.; Won, C.; Jung, H.; Bae, I.; and Jeon, H.-G. 2024a. Fully Explicit Dynamic Guassian Splatting. In *NeurIPS*.

Lee, J. C.; Rho, D.; Sun, X.; Ko, J. H.; and Park, E. 2024b. Compact 3D Gaussian Representation for Radiance Field. In *CVPR*, 21719–21728.

Lee, J. C.; Rho, D.; Sun, X.; Ko, J. H.; and Park, E. 2024c. Compact 3D Gaussian Splatting for Static and Dynamic Radiance Fields. *arXiv preprint arXiv:2408.03822*.

Li, T.; Slavcheva, M.; Zollhoefer, M.; Green, S.; Lassner, C.; Kim, C.; Schmidt, T.; Lovegrove, S.; Goesele, M.; Newcombe, R.; et al. 2022. Neural 3d video synthesis from multi-view video. In *CVPR*.

Li, Z.; Chen, Z.; Li, Z.; and Xu, Y. 2023. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. *arXiv preprint arXiv:2312.16812*.

Lu, T.; Yu, M.; Xu, L.; Xiangli, Y.; Wang, L.; Lin, D.; and Dai, B. 2024. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *CVPR*, 20654–20664.

Niedermayr, S.; Stumpfegger, J.; and Westermann, R. 2024. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *CVPR*, 10349–10358.

Papantonakis, P.; Kopanas, G.; Kerbl, B.; Lanvin, A.; and Drettakis, G. 2024. Reducing the Memory Footprint of 3D Gaussian Splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1): 1–17.

Reizenstein, J.; Shapovalov, R.; Henzler, P.; Sbordone, L.; Labatut, P.; and Novotny, D. 2021. Common Objects in 3D: Large-Scale Learning and Evaluation of Real-life 3D Category Reconstruction. arXiv:2109.00512.

Sabater, N.; Boisson, G.; Vandame, B.; Kerbiriou, P.; Babon, F.; Hog, M.; Gendrot, R.; Langlois, T.; Bureller, O.; Schubert, A.; et al. 2017. Dataset and pipeline for multi-view light-field video. In *Proceedings of the IEEE conference on computer vision and pattern recognition Workshops*, 30–40.

Schönberger, J. L.; and Frahm, J.-M. 2016. Structure-from-Motion Revisited. In *CVPR*.

Shaw, R.; Nazarczuk, M.; Song, J.; Moreau, A.; Catley-Chandar, S.; Dhamo, H.; and Pérez-Pellitero, E. 2024. Swings: sliding windows for dynamic 3D gaussian splatting. In *ECCV*. ECCV.

T, M. V.; Wang, P.; Chen, X.; Chen, T.; Venugopalan, S.; and Wang, Z. 2023. Is Attention All That NeRF Needs? In *The Eleventh International Conference on Learning Representations*.

Wang, F.; Tan, S.; Li, X.; Tian, Z.; and Liu, H. 2022. Mixed neural voxels for fast multi-view video synthesis. *arXiv preprint arXiv:2212.00190*.

Wang, Q.; Wang, Z.; Genova, K.; Srinivasan, P.; Zhou, H.; Barron, J. T.; Martin-Brualla, R.; Snavely, N.; and Funkhouser, T. 2021. IBRNet: Learning Multi-View Image-Based Rendering. In *CVPR*.

Wu, G.; Yi, T.; Fang, J.; Xie, L.; Zhang, X.; Wei, W.; Liu, W.; Tian, Q.; and Xinggang, W. 2023. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. *arXiv preprint arXiv:2310.08528*.

Wu, M.; and Tuytelaars, T. 2024. Implicit gaussian splatting with efficient multi-level tri-plane representation. *arXiv preprint arXiv:2408.10041*.

Xie, S.; Zhang, W.; Tang, C.; Bai, Y.; Lu, R.; Ge, S.; and Wang, Z. 2025. MesonGS: Post-training Compression of 3D

Gaussians via Efficient Attribute Transformation. In *ECCV*, 434–452. Springer.

Yang, Z.; Gao, X.; Zhou, W.; Jiao, S.; Zhang, Y.; and Jin, X. 2023. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. *arXiv preprint arXiv:2309.13101*.

Yang, Z.; Yang, H.; Pan, Z.; and Zhang, L. 2024. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. In *ICLR*.

Yu, A.; Ye, V.; Tancik, M.; and Kanazawa, A. 2021. pixelNeRF: Neural Radiance Fields from One or Few Images. In *CVPR*.

Yun, Y.; Bae, J.; Son, H.; Kim, S.; Lee, H.; Bang, G.; and Uh, Y. 2025. Compensating Spatiotemporally Inconsistent Observations for Online Dynamic 3D Gaussian Splatting. In *ACM SIGGRAPH 2025 conference proceedings*, SIGGRAPH Conference Papers '25, 1–9. ACM.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 586–595.