

A Compact Dynamic 3D Gaussian Representation for Real-Time Dynamic View Synthesis

Kai Katsumata[✉], Duc Minh Vo[✉], and Hideki Nakayama[✉]

The University of Tokyo, Japan

{katsumata, vmduc, nakayama}@nlab.ci.i.u-tokyo.ac.jp

Abstract. 3D Gaussian Splatting (3DGS) has shown remarkable success in synthesizing novel views given multiple views of a static scene. Yet, 3DGS faces challenges when applied to dynamic scenes because 3D Gaussian parameters need to be updated per timestep, requiring a large amount of memory and at least a dozen observations per timestep. To address these limitations, we present a compact dynamic 3D Gaussian representation that models positions and rotations as functions of time with a few parameter approximations while keeping other properties of 3DGS including scale, color, and opacity invariant. Our method can dramatically reduce memory usage and relax a strict multi-view assumption. In our experiments on monocular and multi-view scenarios, we show that our method not only matches state-of-the-art methods, often linked with slower rendering speeds, in terms of high rendering quality, but also significantly surpasses them by achieving a rendering speed of 118 frames per second at a resolution of $1,352 \times 1,014$ on a single GPU.

1 Introduction

The landscape of novel view synthesis of scenes captured through multiple images/videos has undergone a revolutionary transformation, owing principally to major breakthroughs in neural radiance field (NeRF) approaches [6, 41, 57]. Although they achieve remarkable visual quality, particularly in dynamic scenes [4, 21, 31, 34, 45], NeRFs inevitably confront hurdles in terms of high-speed training and rendering [41, 43, 44, 48]. This limitation is attributed to their reliance on multi-layer perceptrons (MLPs). Recently, 3D Gaussian Splatting (3DGS) [26] introduced a differentiable 3D Gaussian representation and point-based rasterization, signaling a departure from neural network reliance. 3DGS has emerged as a promising solution that not only accelerates training and rendering processes but also delivers high-quality rendered scenes, rivaling the levels set by NeRF [41] on static scenes.

Nonetheless, in the realm of dynamic scene synthesis, 3DGS faces challenges related to memory usage and the need for many observations [38]. In particular, a significant number of 3D Gaussian parameters must be stored per timestep, resulting in a non-negligible increase in memory usage and the need for numerous observations per timestep. This poses challenges in monocular or few-view setups, as their strict multi-view assumption demands advanced facilities or expertise, limiting flexibility in capturing setups. Exploring 3DGS without multi-view assumption enables dynamic view synthesis with a simple and easy camera setup, which is the primary goal of this study.

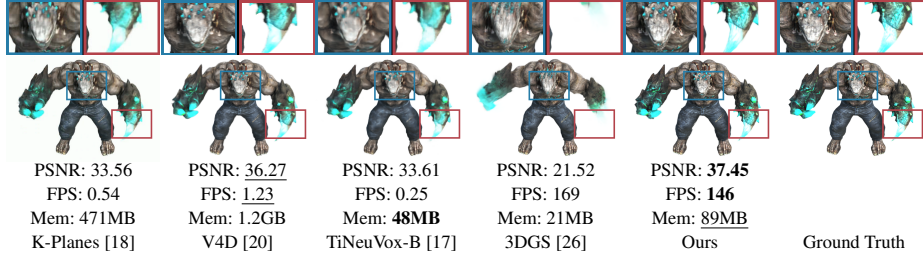


Fig. 1: We show examples of novel view synthesis on the MUTANT scene in the D-NeRF dataset, visual quality (PSNR), rendering speed (FPS), and memory used to store optimized parameters. Our method yields reconstruction fidelity competitive with SoTAs with real-time rendering, achieving $100\times$ faster rendering speed than V4D and reasonable memory size. Non-obvious differences in quality are highlighted. **Bold** typeface number indicates the best result among the methods with the competitive rendering quality (excepting for 3DGS), and the underline one does the second best.

To achieve memory-efficient real-time dynamic view synthesis from monocular and multi-view videos, we present a compact dynamic 3D Gaussian representation, containing time-invariant and time-varying parameters to capture dynamic motion effectively. Similarly to [26, 38], we use scaling factors in the covariance matrix, opacity, and color as time-invariant parameters. As modeling the change in positions over time is important to represent dynamic scenes [43–45], we express each 3D Gaussian position as a function of time to model the temporal change in the position. We also represent 3D Gaussian rotation as a time-varying parameter because the rotation of the objects in the world can be typically changed. Inspired by the studies that model motion as periodic [2, 71], we fit the position using the Fourier approximation. We fit the rotation using the linear approximation. The time-varying parameters make our representation dynamic, meaning that a 3D Gaussian moves and rotates over time. Moreover, as we use a function with a few parameters to represent the position, the small degree of freedom contributes to the smoothness of reconstructed scenes, enhancing the robustness against unseen views. Crucially, the memory consumption of our representation is solely determined by the number of 3D Gaussians and the number of the approximation function parameters, remaining independent of input length. Beyond optimizing Gaussian representations through image-level reconstruction, we further enhance temporal consistency by supervising the Gaussian with optical flow obtained from input videos. This ensures high-quality reconstruction and facilitates the generalization of the representation.

Our experiments on dynamic datasets (D-NeRF [45], DyNeRF [31], and HyperNeRF [44]) demonstrate the effectiveness of optimizing our dynamic 3D Gaussian from both monocular and multi-view videos, showing that our proposed method achieves rendering quality that rivals that of previous NeRFs [17, 18, 20]. In addition to faithful rendering quality, the proposed method achieves rendering speeds similar to a fast radiance field method [26] while avoiding large memory increases caused by a dynamic extension (see Fig. 1). Finally, we show an editing application enabled by the explicit property of 3D Gaussian representations. In summary, our contributions are as follow:

- We present a compact dynamic 3D Gaussian representation with time-varying Gaussian parameters equipped with basis functions for representing dynamic scenes.

- As a 3D Gaussian representation is defined over all the timesteps, the 3D Gaussian parameters can be optimized with the frames at all the timesteps, enabling dynamic scene reconstruction from monocular or few-view videos.
- Our dynamic 3D Gaussian representation facilitates real-time high-quality dynamic scene rendering of high-resolution images of $1,352 \times 1,014$ with a frame rate of 118 FPS using a single GPU.

2 Related Work

We briefly overview radiance fields for dynamic scenes and discuss recent efficient explicit representation methods (grid-, plane-, hash-, and point-based), contextualizing our work within real-time dynamic view synthesis.

2.1 Dynamic view synthesis

Applications in virtual reality and computer vision often need reconstruction of dynamic scenes. Several works extend NeRF [41] to handle dynamic scenes in multi-view or monocular setups by time-varying NeRF [21, 31, 45, 60]. The regularization techniques for temporal smoothness enable suitable scene representations from monocular videos [33]. Additional sensory information is also useful for spatio-temporal regularization. Some attempts [21, 33, 58] employ depth or flow, which are observed or predicted with external networks to reconstruct from sparse observations. Deformation-based approaches [43, 44, 54, 59], another research direction in dynamic reconstruction, combine static NeRF and deformation fields. Although tremendous efforts show high visual quality for dynamic view synthesis, the frequent querying of MLP in NeRFs results in the drawback of slow optimization and rendering [65]. Our study aims to enable real-time dynamic view synthesis with high visual quality. We aim to extend 3DGS to dynamic scene reconstruction to achieve high-speed rendering while maintaining the rendering quality from sparse training views.

2.2 Explicit Radiance Fields

Recent studies [11, 19, 69] have addressed the issue in implicit models (*i.e.*, NeRFs) by exploring explicit models, reducing optimization and rendering time. Plenoxels [19] directly optimizes 3D grid representation instead of neural networks. Generally, explicit models sacrifice visual quality for fast training time [19]. Hybrid approaches [11, 17, 18, 20, 42, 53] aim to achieve better trade-offs between training time and visual quality. Instant-NGP allows for a compact MLP by exploiting a multi-level hash grid to encode positions to feature vectors [42]. Plane-based approaches are designed principally to represent bounded scenes [3, 9, 10, 14, 18, 23]. MERF [49] employs a multiresolution representation and a fast contraction function to reconstruct unbounded scenes. For dynamic scenes, K-planes [18] decomposes 4D dynamic volumes into multiple feature planes and employs an MLP-based feature decoder for determining color and density. Structured representations still grapple with the trade-off between rendering speed and quality. In this study, unstructured 3D Gaussians promise large gains in rendering speed.

2.3 Point-based rendering

Points, which naturally come from depth sensors, Structure from Motion (SfM) [51], or common Multi-View Stereo (MVS) algorithms [50, 52], offer a useful representation of fine-grained scenes and complex objects, and facilitate computationally efficient rendering. Consequently, they have been studied comprehensively in the vision and graphics community. The differentiable pipeline for point-based rendering results in points can be used for reconstructing 3D scenes [26–29]. 3DGS [26] achieves real-time rendering with high visual quality for unbounded static scenes at the expense of the generalization performance derived from NeRF’s continuous neural field representation. 3DGS is replacing NeRFs as the backbone of text-to-3D models, leading to faster 3D generation [1, 12, 55, 66, 68]. Recently, Dynamic 3D Gaussians [38] employs 3DGS for dynamic scenes, which models dynamic scenes by the Gaussian position and rotation at each timestamp. The position and rotation of Gaussians at every timestamp are effective in modeling scenes from dense multi-view dynamic scenes. However, this approach presents difficulties in reconstructing monocular dynamic scenes, resulting in excessive memory consumption, particularly for extended input sequences. Specifically, the space complexity of the method for a scene with T frames is $O(TN)$, where N is the number of 3D Gaussians. Our goal is to reduce memory consumption by representing time-varying position and rotation with approximation using a few parameters. The space complexity of our method is $O(LN)$, where L is the number of parameters of the approximation, and usually $L < T$.

Concurrent works on dynamic view synthesis includes approaches combining Gaussian Splatting with MLPs [5, 22, 32, 35, 36], approaches focusing on Gaussian representation [13, 15, 16, 64, 67], and approaches for specific targets [30, 46, 47, 72]. Space-timeGaussian [32] focuses on dynamic view synthesis from multi-view videos, unlike this study, by combining Gaussian Splatting and MLPs. [64] aims to model motion by employing a deformation field network while sacrificing rendering speed. [67] splits Gaussians in a time direction, and each Gaussian only focuses on a local temporal space. Four-dimensional (4D) Rotor Gaussian Splatting [15] models a local temporal space via temporal slicing for fast rendering. We aim to build a memory-efficient Gaussian representation for dynamic scenes, even for monocular scenes, while maintaining pure 3D Gaussian representation in order not to sacrifice the gift of 3D Gaussians, such as outstanding rendering speed and ease of direct editing of the scene.

3 Method

Given images with timesteps and camera parameters obtained from videos, our task is to learn a 4D spatial-temporal representation of a dynamic scene that enables fast and high-quality view rendering. To achieve this, we use 3DGS in dynamic view synthesis. The original 3D Gaussian representation [26] is defined by a position (mean), a covariance matrix (decomposed into a rotation matrix and a scaling vector), a color (determined by spherical harmonics (SH) [8] coefficient), and an opacity. To represent dynamic scenes, each 3D Gaussian in our method (Fig. 2) regards the position and rotation as time-varying parameters and others as time-invariant parameters over time (Sec. 3.1). Given a set of 3D Gaussians, intrinsic and extrinsic camera parameters, and

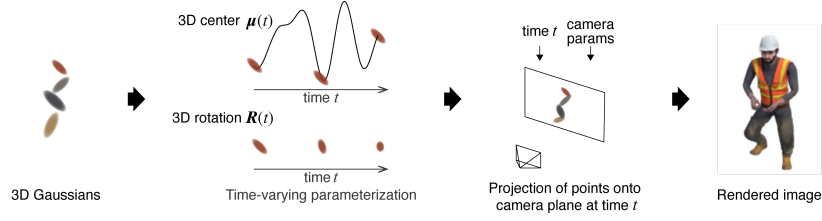


Fig. 2: Overview of our dynamic view synthesis framework. Our dynamic 3D Gaussian representation takes temporal modeling of 3D centers and rotations with Fourier and linear approximation, respectively. Our representation parameters are shared over all the timesteps, and observations of each timestep hint at the representation for other timesteps, enabling compact representation and reconstruction of dynamic scenes from few-view videos. In this figure, we only illustrate the time-varying parameterization of one Gaussian for the sake of simplicity.

a timestep, we render images with the 3DGS technique [26], which renders an image by employing Gaussians within the camera plane out of a set of 3D Gaussians (Sec. 3.2). We update the Gaussian parameters to decrease the distance between rendered and training images in image and flow spaces (Sec. 3.3). Flow reconstruction loss enhances the temporal consistency of the learned representation, resulting in plausible image reconstruction. The small degrees of freedom of our representation essentially facilitate the reconstruction of dynamic scenes from a few observations.

3.1 Dynamic 3D Gaussian representation

One possible extension of 3DGS [38] to dynamic scenes is to model the scenes per timestep explicitly. Although that strategy allows for flexible modeling for dynamic scenes, it requires 3D Gaussian parameters per timestep, increasing the memory size proportionally to video length. Since the representation for each time is optimized by observations with the number of cameras, the strategy lacks sufficient observations in monocular or few-view video setups, limiting its effectiveness in such scenarios.

To design a compact dynamic 3D Gaussian representation, we express 3D Gaussian parameters using only a few parameters to achieve faithful reconstruction without a large increase in parameters. Our dynamic scene representation comprises a set of dynamic 3D Gaussians, extending the static 3D Gaussian introduced in [26]. This representation allows 3D Gaussians to move through the scene over time, using time-varying parameters (center position and rotation factors) and time-invariant parameters (scale, color, and opacity). Each dynamic Gaussian encapsulates the following parameters:

- 1) a 3D center at time t : $[x(t), y(t), z(t)]^T \in \mathbb{R}^3$,
- 2) a 3D rotation at time t represented by a quaternion:
 $[q_x(t), q_y(t), q_z(t), q_w(t)]^T \in \mathbb{R}^4$,
- 3) a scaling factor: $[s_x, s_y, s_z]^T \in \mathbb{R}^3$,
- 4) SH coefficients representing color with the degrees of freedom k :
 $h \in \mathbb{R}^{3 \times (k+1)^2}$,
- 5) an opacity: $o \in \mathbb{R}$.

Each Gaussian at time t is characterized by a 3D center $\boldsymbol{\mu}(t) = [x(t), y(t), z(t)]^\top$ and a 3D covariance matrix $\boldsymbol{\Sigma}(t)$. The density of the 3D Gaussian at the intersection \mathbf{x} with a ray is obtained as follows:

$$G_t(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}(t))^\top \boldsymbol{\Sigma}(t)^{-1}(\mathbf{x}-\boldsymbol{\mu}(t))}. \quad (1)$$

To constrain the covariance matrix $\boldsymbol{\Sigma}(t)$ such that it is a positive semi-definite matrix during optimization, the covariance matrix $\boldsymbol{\Sigma}(t)$ is decomposed by using a scaling matrix $\mathbf{S} = \text{diag}(s_x, s_y, s_z)$ and a rotation matrix $\mathbf{R}(t)$ as $\boldsymbol{\Sigma}(t) = \mathbf{R}(t)\mathbf{S}\mathbf{S}^\top\mathbf{R}(t)^\top$. Here, the rotation matrix $\mathbf{R}(t)$ is represented by quaternion $(q_x(t), q_y(t), q_z(t), q_w(t))$. Since most parts of the dynamic scene hardly change in scale because the solid objects (*e.g.*, humans, animals, and things) scarcely expand or shrink, we maintain the scale parameter as a constant to reduce the model size. In what follows, we formally define the 3D center and rotation.

Since motion in dynamic scenes is primarily described by changing the position of points like scene or optical flow [37, 61], we model the 3D center with an expressive approximation. We approximate the 3D position $x(t), y(t), z(t)$ using Fourier approximation. At time t , it is represented by

$$\begin{aligned} x(t) &= w_{x,0} + \sum_{i=1}^L w_{x,2i-1} \sin(2i\pi t) + w_{x,2i} \cos(2i\pi t), \\ y(t) &= w_{y,0} + \sum_{i=1}^L w_{y,2i-1} \sin(2i\pi t) + w_{y,2i} \cos(2i\pi t), \\ z(t) &= w_{z,0} + \sum_{i=1}^L w_{z,2i-1} \sin(2i\pi t) + w_{z,2i} \cos(2i\pi t), \end{aligned} \quad (2)$$

where, $w_{\cdot,0}, \dots, w_{\cdot,2L}$ are the intercept and coefficients of the position, and L is the number of terms (harmonics). We remark that a polynomial approximation is inadequate due to underfitting with a small number of bases and overfitting with higher-order polynomials. For these reasons, we choose the Fourier approximation.

3DGS uses anisotropic 3D Gaussians, resulting in the need for dynamic modeling of Gaussian rotations. We approximate the 3D rotation (quaternion) over time using a linear approximation because a unit quaternion can be approximated locally as linear when considering its tangent plane. At time t , it is defined as

$$\begin{aligned} q_x(t) &= w_{qx,0} + w_{qx,1}t, & q_y(t) &= w_{qy,0} + w_{qy,1}t, \\ q_z(t) &= w_{qz,0} + w_{qz,1}t, & q_w(t) &= w_{qw,0} + w_{qw,1}t, \end{aligned} \quad (3)$$

where $w_{\cdot,0}$ and $w_{\cdot,1}$ are intercepts and coefficients of the rotation, respectively. We project the quaternion $q(t)$ onto the unit quaternion by normalizing it: $q(t)/\|q(t)\|$, to ensure that the quaternion at time t is a unit quaternion.

For each Gaussian, the preceding definitions yield $3L+8+3+3(k+1)^2+1$ parameters with respect to the 3D center, 3D rotation, scale, color, and opacity. Notably, the parameter count for each Gaussian is defined merely by the number of approximation terms and spherical harmonic degrees of freedom, with no regard to time length. Compared to methods that store parameters for each timestep, our approach saves on memory usage. Memory consumption in our dynamic scene representation is determined by two hyperparameters (*i.e.*, L and k) and the number of Gaussians used. Furthermore, the representation defined as a function of time over continuous time inhibits discontinuous movement through time. This characteristic improves robustness in novel view synthesis settings.

3.2 Rendering via 3D Gaussian Splatting

Rendering with 3D Gaussian applies splatting techniques [26] to the Gaussian within the camera planes. Zwicker *et al.* [73] introduced the projection of the 3D covariance matrix to the 2D one. The 3D covariance matrix Σ is projected into a 2D one Σ' given a viewing transformation \mathbf{W} as $\Sigma'(t) = \mathbf{J}\mathbf{W}\Sigma(t)\mathbf{W}^T\mathbf{J}^T$, where \mathbf{J} is the Jacobian of the affine approximation of the projective transformation at Gaussian center $\mu(t)$:

$$\mathbf{J} = \begin{bmatrix} \frac{1}{v_z} & 0 & -\frac{v_x}{v_z^2} \\ 0 & \frac{1}{v_z} & -\frac{v_y}{v_z^2} \\ 0 & 0 & 0 \end{bmatrix}, \quad (4)$$

where $[v_x, v_y, v_z]^T = \mathbf{W}\mu(t)$ is the camera coordinate of the Gaussian center $\mu(t)$ obtained by the viewing transformation, which projects the points from world space to camera space.

Similar to NeRF style volumetric rendering, point-based rendering computes the color C of a pixel by evaluating the blending of N ordered points that overlap the pixel $C = \sum_{i=1}^N c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)$, where c_i represents the color of a Gaussian evaluated by SH coefficients, and α_i represents the density that is calculated from a 2D Gaussian with the 2D covariance Σ' and 2D center μ' at time t and the optimized opacity o .

3.3 Optimization of the dynamic 3D Gaussian representation

We optimize the Gaussian parameters, *i.e.*, intercepts and coefficients of position and rotation w , a scaling factor s_x, s_y, s_z , SH coefficients h , and an opacity o , based on the iterations of rendering and a comparison of the rendered images with training frames in the captured videos. To compare the rendered and training views, the loss function contains the L1 loss and the structural similarity (SSIM) [63] loss \mathcal{L}_{D-SSIM} :

$$\mathcal{L}_{\text{recon}} = (1 - \lambda)|\hat{I} - I| + \lambda \mathcal{L}_{D-SSIM}, \quad (5)$$

where \hat{I} and I are the rendered and target images, respectively. The loss function moves and rotates the anisotropic Gaussians and changes their color and opacity so that each Gaussian covers a homogeneous area. Since the loss just fixes incorrectly positioned Gaussians, the over- or under-representation of the set of Gaussians for the scene needs a mechanism for creating Gaussians that reconstruct the scene or destroy extra Gaussians. We also follow the divide and prune techniques in 3DGS for producing a compact and precise representation of the scene. We surveil the gradients of each Gaussian and densify Gaussians by splitting a Gaussian with a large gradient and a large scale into two small Gaussians, and cloning a Gaussian with a large gradient and a small scale to two Gaussians. Moreover, we remove transparent Gaussians with an opacity less than a threshold value of 0.005.

Following [26], we initialize a set of Gaussians using a set of sparse points from SfM [51] for real scenes, and we initialize a set of Gaussians randomly using a uniform distribution for synthetic scenes owing to the absence of the prior. We adopt a two-stage optimization strategy consisting of static and dynamic stages. Deeming the

frames in the captured datasets as static scenes, we optimize static representation in the static stage to learn the prior of Gaussians. In other words, we optimize the parameters that are consistent all over time (*i.e.*, scale, SH coefficients, and opacity) and the intercepts for the center and rotation ($w_{x,0}, w_{y,0}, w_{z,0}, w_{qx,0}, w_{qy,0}, w_{qz,0}, w_{qw,0}$) among the Gaussian parameters in the static stage. After the static stage, we optimize all the parameters of the set of Gaussians to reconstruct a dynamic region as a dynamic stage.

Another challenge in the dynamic scene reconstruction is ambiguity caused by the limited number of captured views at a timestep. Since a dynamic scene contains temporal changes, such as moving objects and changing shapes, sharing the scene information over frames with different timesteps is difficult. To overcome the ambiguity, we employ flow information. Similar to our 3D Gaussian, scene flow [39, 40, 62] is defined as the position of a point in 3D space and its motion. These 3D points originate from different mechanisms than those in 3D Gaussian, making matching in 3D space difficult. Since optical flow defined on the image plane can be directly matched with a 3D Gaussian and is readily to compute from monocular inputs, we supervise the flows of the optimizable Gaussians with the ground truth optical flows of the input frames. We use RAFT [56] to obtain ground truth flow for training views: forward flow f_{fwd} and backward flow f_{bwd} between two adjacent frames. The flow loss $\mathcal{L}_{\text{flow}}$ takes the L1 loss between the ground truth flows and the optical flow of the Gaussian for both directions of the flows. The flow loss gives our method spatial-temporal consistency without any additional computation cost in rendering. We combine the flow loss $\mathcal{L}_{\text{flow}}$ with the reconstruction loss that compares the rendered and training views:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \lambda_{\text{flow}} \mathcal{L}_{\text{flow}}(\hat{F}, F), \quad (6)$$

where $F = \{f_{\text{fwd}}, f_{\text{bwd}}\}$ and \hat{F} are the ground truth flow and the flow of the Gaussians, respectively, and λ_{flow} is a balancing hyperparameter for the flow term. Instead of applying an optical flow algorithm for rendering, we create pseudo optical flow from a Gaussian representation. Scene motion is represented solely by the 3D mean coefficients: $w_{x,1 \leq i}, w_{y,1 \leq i}, w_{z,1 \leq i}$. Scene flow in 3D space can be computed by

$$\begin{aligned} \hat{f}_{\text{fwd}}^x &= x(t + \Delta t) - x(t), & \hat{f}_{\text{bwd}}^x &= x(t) - x(t - \Delta t), \\ \hat{f}_{\text{fwd}}^y &= y(t + \Delta t) - y(t), & \hat{f}_{\text{bwd}}^y &= y(t) - y(t - \Delta t), \\ \hat{f}_{\text{fwd}}^z &= z(t + \Delta t) - z(t), & \hat{f}_{\text{bwd}}^z &= z(t) - z(t - \Delta t), \end{aligned} \quad (7)$$

where Δt is the difference between the timesteps of the two image frames. The scene flow is projected into a 2D camera plane using

$$\hat{f}_{\{\text{fwd}, \text{bwd}\}}^{\text{xyz}} = \mathbf{J}[\hat{f}_{\{\text{fwd}, \text{bwd}\}}^x, \hat{f}_{\{\text{fwd}, \text{bwd}\}}^y, \hat{f}_{\{\text{fwd}, \text{bwd}\}}^z]^T, \quad (8)$$

where \mathbf{J} is the Jacobian of the affine approximation of the projective transformation at the Gaussian center μ (Eq. (4)). Regarding scene flows on the camera plane as RGB colors, point-based rendering can compute an optical flow of a pixel through α -blending:

$$\hat{f}_{\text{fwd}} = \sum_{i=1}^N \hat{f}_{\text{fwd}, i}^{\text{xyz}} \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \quad (9)$$

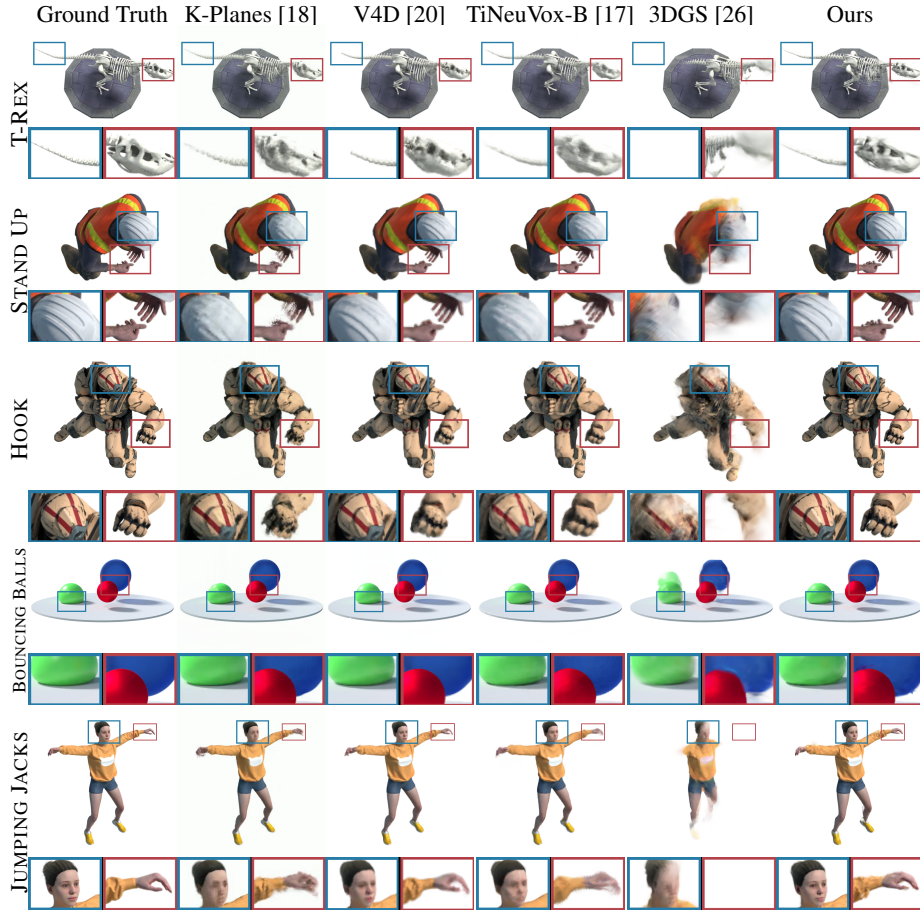


Fig. 3: Qualitative comparison on D-NeRF [45]. We highlight the differences by zoom view. Our method achieves competitive visual quality with strong baselines. While our method successfully reconstructs intricate details like hands, it causes a blurred sphere shape.

The backward flow is calculated in the same way. The optical flow \hat{F} consists of the forward flows \hat{f}_{fwd} and backward flows \hat{f}_{bwd} for all pixels. We exclude the flow loss for the D-NeRF dataset because the teleport of the cameras between adjacent frames causes difficulties in calculating ground truth flows.

4 Experiment

4.1 Evaluation data

We evaluate our compact dynamic Gaussian representation using dynamic scene datasets: a synthetic one D-NeRF [45] and two real ones, *i.e.*, DyNeRF [31] and HyperNeRF [44].

Table 1: Quantitative results on the D-NeRF dataset [45]. Our method performs competitively against NeRF approaches in terms of visual quality and achieves the fastest rendering speed among the highest-performing methods. Results except the FPS of [17, 18, 20] are adopted from the original papers. The **best** and second best scores among competing methods are highlighted.

	PSNR \uparrow	MS-SSIM \uparrow	LPIPS \downarrow	FPS \uparrow	Train Time \downarrow	Mem \downarrow
TiNeuVox-S [17]	30.75	0.96	0.07	0.32	8 mins	8MB
TiNeuVox-B [17]	<u>32.67</u>	<u>0.97</u>	<u>0.04</u>	0.13	<u>28 mins</u>	<u>48MB</u>
K-Planes [18]	31.61	<u>0.97</u>	-	0.54	52 mins	\sim 497MB
V4D [20]	33.72	0.98	0.02	<u>1.47</u>	6.9 hrs	1.2GB
3DGS [26]	20.51	0.89	0.07	170	6 mins	\sim 50MB
D-3DGS	17.22	0.81	0.13	173	15 mins	\sim 913MB
Ours	32.19	<u>0.97</u>	<u>0.04</u>	150	8 mins	\sim 159MB

Table 2: Quantitative results on the DyNeRF datasets [31]. Results excepting FPS of [18, 20] are adopted from the original papers. The **best** and second best scores among competing methods (excepting 3DGS) are highlighted. While our method matches NeRFs in terms of rendering quality, our method matches 3DGS in terms of rendering speed. Besides, our method is 20 times more compact than Dynamic3DGaussians.

	PSNR \uparrow	MS-SSIM \uparrow	LPIPS \downarrow	FPS \uparrow	Train Time \downarrow	Mem \downarrow
K-Planes [18]	31.63	0.964	-	0.31	1.8 hrs	\sim 309MB
V4D	28.96	0.937	<u>0.17</u>	0.11	4 hrs	1.2GB
3DGS [26]	20.94	0.800	<u>0.29</u>	109	20 mins	\sim 198MB
D-3DGS	24.36	0.834	0.25	119	51 mins	\sim 2.3GB
Dynamic3DGaussians [38]	27.79	0.869	0.23	51	2.1 hrs	\sim 6.6GB
Ours	<u>30.46</u>	<u>0.955</u>	0.15	<u>118</u>	<u>1 hrs</u>	<u>\sim338MB</u>

D-NeRF dataset [45]. This dataset comprises eight videos of varying lengths, ranging from 50 to 200 frames per video. The camera setup is designed to mimic a monocular camera setting by teleporting between adjacent timesteps. The test views are from novel camera positions. We train and render at the resolution of 800×800 .

DyNeRF dataset [31]. The multi-camera dataset includes six 10-second videos captured at 30 FPS using 15–20 synchronized fixed cameras. For evaluation, a central camera is used, while training utilizes frames from the other cameras. The training and rendering resolution is set at $1,352 \times 1,014$.

HyperNeRF dataset [44]. This dataset encompasses videos ranging from 8 to 15 s, captured at 15 FPS using two Pixel 3 phones. The training and rendering processes are conducted at a resolution of 540×960 .

4.2 Implementation details

We adhere to the experimental setup in the 3DGS paper [26]. The number of approximation terms of the Gaussian centers L is set to 2 for the D-NeRF dataset. For the other datasets, L is set to 5 from preliminary experiments. Our two-stage optimization process begins with an initial fitting of parameters, excluding the coefficients for Gaussian center and rotation. This initial stage spans 3,000 iterations and utilizes all training views in a static setting. Subsequently, we engage in a dynamic stage, adjusting all Gaussian

Table 3: Quantitative results on the HyperNeRF dataset [44]. Our method demonstrates competitive performance in rendering quality across all scenes, surpassing the compared methods in rendering speed. Furthermore, our method is not inferior to the compared methods in training time and memory size.

	FPS \uparrow	Train Time \downarrow	Mem \downarrow	BROOM		3D PRINTER		CHICKEN		PEEL BANANA		Mean	
				PSNR \uparrow	SSIM \uparrow	PSNR \uparrow	SSIM \uparrow	PSNR \uparrow	SSIM \uparrow	PSNR \uparrow	SSIM \uparrow	PSNR \uparrow	SSIM \uparrow
HyperNeRF [44]	<u>0.36</u>	48 hrs \dagger	<u>15MB</u>	19.3	0.591	20.0	0.821	26.9	0.948	23.3	<u>0.896</u>	22.2	0.811
TiNeuVox-B [17]	0.14	30 mins	48MB	<u>21.5</u>	0.686	22.8	<u>0.841</u>	28.3	0.947	24.4	0.873	24.3	<u>0.837</u>
V4D [20]	0.15	7 hrs	1.2GB	22.1	0.669	<u>23.2</u>	0.835	28.4	0.929	<u>25.2</u>	0.873	<u>24.7</u>	0.827
Ours	188	<u>1 hrs</u>	\sim 720MB	22.1	0.789	25.5	0.919	<u>28.3</u>	0.934	26.6	0.920	25.6	0.890

\dagger Train time of HyperNeRF [44] is estimated from their paper’s descriptions. Originally reported as 8 hours on 4 TPU v4s [25], the TPU v4 is slightly faster than the A100 GPU, and the A100 GPU is at least 1.5 times faster than the A6000 GPU.

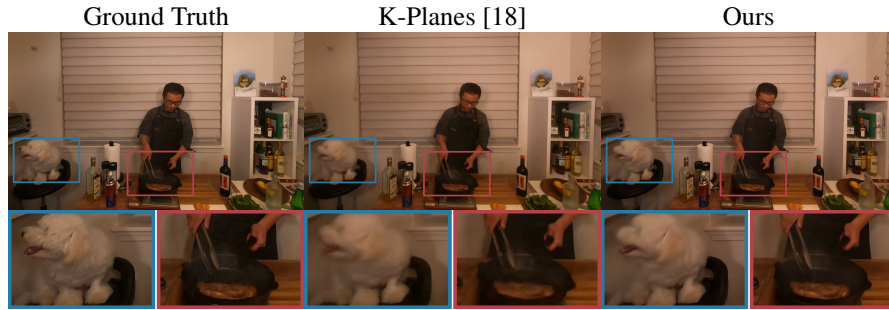


Fig. 4: Qualitative comparison on the DyNeRF dataset [31]. The differences are zoomed in.

parameters in 27,000 iterations. The entire optimization process encompasses 30,000 iterations. Following [26], λ is set to 0.2. We set the flow loss weight λ_{flow} to 1,000 and acquire ground truth flow through the RAFT pretrained on the Sintel dataset [7]. All experiments are conducted on a single RTX A6000 GPU.

4.3 Evaluation setup

Compared methods. We benchmark our method against the following baseline methods: TiNeuVox [17], K-Planes [18], V4D [20], HyperNeRF [44], 3D Gaussian Splatting (3DGS) [26], Dynamic3DGaussians [38], and a D-3DGS baseline. D-3DGS is the dynamic extension of 3DGS, which stores both position and rotation for each timestep.

Evaluation metrics. We assess the methods using various metrics, including PSNR [24], SSIM [63], LPIPS [70], FPS, Training time, and memory used to store optimized parameters. Memory consumption includes the 3D Gaussian parameters, voxel/plane representation, and neural network parameters.

4.4 Experimental results

Quantitative results. The quantitative results on the D-NeRF dataset are detailed in Tab. 1. Our method demonstrates a performance comparable to TiNeuVox and K-Planes

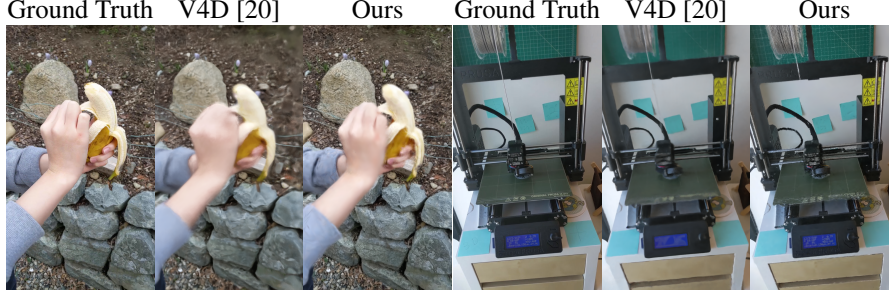


Fig. 5: Qualitative comparison on HyperNeRF [44]. Our method offers sharp results.

Table 4: Per-scene quantitative comparison on D-NeRF scenes of different L , which stands for the number of harmonic terms in the Fourier approximation, and other design choices. The highest mean score is achieved with $L = 2$, but increasing the complexity L (the number of coefficients) improves visual quality in some scenes (JUMPING JACKS and T-REX). The spline approximations bring marginal improvements in some scenes but slower rendering. The time-varying scale (the last row) also provides minor gains in some cases and increases the memory size. The setting reported in Fig. 3 is highlighted with a gray background.

	STAND UP		JACKS		BALLS		LEGO		WARRIOR		HOOK		T-REX		MUTANT		Mean	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
$L = 1$	40.21	0.994	27.22	0.952	30.27	0.972	24.26	0.940	32.42	0.937	32.84	<u>0.980</u>	25.15	0.957	38.04	0.994	31.30	0.965
$L = 2$	<u>39.10</u>	<u>0.993</u>	30.95	0.980	33.29	0.984	23.15	0.922	34.15	0.956	<u>33.19</u>	0.981	26.22	<u>0.962</u>	<u>37.45</u>	<u>0.993</u>	32.19	<u>0.971</u>
$L = 3$	38.09	0.990	32.78	0.984	32.54	0.979	22.12	0.881	35.36	0.955	30.23	0.967	27.73	0.956	35.06	0.985	31.74	0.962
$L = 4$	35.83	0.984	32.93	0.982	30.39	0.969	21.06	0.855	34.38	0.947	27.74	0.946	28.17	0.952	32.58	0.974	30.39	0.951
$L = 5$	32.89	0.976	30.71	0.977	27.68	0.959	20.20	0.825	32.64	0.933	25.43	0.923	26.11	0.927	29.09	0.960	28.10	0.934
Linear	27.77	0.973	23.10	0.921	26.68	0.959	22.27	0.922	17.39	0.869	24.98	0.946	26.82	0.955	33.98	<u>0.993</u>	25.37	0.942
Quadratic	29.40	0.978	23.44	0.926	27.51	0.963	22.45	0.924	17.70	0.876	25.70	0.950	26.93	0.957	33.20	0.992	25.79	0.946
Cubic	29.98	0.979	23.71	0.928	27.76	0.964	22.37	0.921	18.04	0.884	25.96	0.951	26.25	0.954	33.51	0.992	25.95	0.947
Spline (5)	38.87	<u>0.993</u>	31.96	<u>0.983</u>	<u>32.96</u>	<u>0.980</u>	23.09	0.918	34.46	0.959	31.69	0.978	26.68	0.970	37.07	<u>0.993</u>	<u>32.10</u>	0.972
Spline (6)	38.00	0.992	31.84	0.984	32.81	<u>0.980</u>	22.25	0.903	<u>35.24</u>	0.965	30.87	0.974	26.98	0.974	35.94	0.991	31.74	0.970
Linear (scale)	38.32	<u>0.993</u>	30.91	0.980	32.55	0.984	<u>23.87</u>	<u>0.930</u>	34.43	<u>0.956</u>	33.43	0.981	25.45	0.961	36.58	<u>0.993</u>	31.94	0.972

in terms of visual quality as measured by PSNR, SSIM, and LPIPS. Notably, it excels in training time, FPS, and memory size, achieving a rendering speed that is $300\times$ faster than that of K-Planes. Furthermore, our method surpasses both 3DGS and D-3DGS in terms of visual quality without compromising rendering speed. In the DyNeRF scenes experiment, detailed in Tab. 2, while our method does not exceed the baseline in reconstruction quality, it shows a substantial improvement in FPS. Since the DyNeRF scenes contain multi-view data, the D-3DGS baseline naturally improves static 3DGS, unlike monocular scenes. Our method even attains rendering speeds that exceed real-time performance at a high resolution of $1,354\times 1,014$. For the challenging HyperNeRF dataset, which is captured by only two moving cameras and referenced in Tab. 3, our method not only demonstrates rapid rendering speeds but also achieves higher average PSNR and SSIM scores than the compared methods.

Qualitative results. Figures 3 to 5 show that our method yields faithful reconstruction for the dynamic scenes. Unlike the structured representation, which has a fixed size of grids, the unstructured nature of 3D Gaussians enables adaptive control of the expres-



Fig. 6: Qualitative comparison of disabled and enabled flow loss on DyNeRF. We highlight the difference by zoom view.



Fig. 7: Composition of two scenes. Our method allows for the addition of adding 3D objects represented 3D Gaussians into a 3D Gaussian scene. We highlight the added object.

siveness of the representation, delivering sharper renderings. As seen with the results for BOUNCING BALLS, since our method has discrete primitives, it sometimes fails to reproduce smooth boundaries.

Effect of the number of parameters L . Table 4 shows per-scene PSNR and SSIM scores of K-Planes and our method with the different L (Eq. (2)). It is observed that the optimal L for novel view synthesis varies from scene to scene, highlighting the necessity for complex approximations to capture intricate motions effectively.

Effect of flow loss. Additionally, visual comparisons drawn from our method without and with the flow loss (Fig. 6) reveal that incorporating the flow loss mitigates ghostly artifacts and significantly enhances the accuracy of color reconstruction.

Design choice. Our method is very flexible and allows for the use of arbitrary approximation functions and the choice of time-varying parameters. Table 4 also shows the experimental results of other options for the design of the model to facilitate future dynamic scene reconstruction. The linear, quadratic, and cubic baselines approximate time-varying 3D positions with polynomials of degrees one, two, and three, respectively. The Spline (5) and Spline (6) baselines approximate 3D positions with spline approximations of five and six points, respectively. The linear (scale) baseline approximates time-varying scales with the linear approximation in addition to positions and rotations. Although a Spline baseline gives minor performance gains in some cases, it

achieves 91 FPS for rendering, showing slower rendering than the proposed method. The linear (scale) baseline does not show additional parameters that would result in performance improvements. For time-varying 3D rotation, we also consider the approximation with slerp. Since it does not offer performance gains while causing numerical instability for static Gaussians, we use linear approximation for rotation. For faster rendering and compact representation, we use the Fourier approximation for 3D positions and model 3D positions and rotations as time-varying parameters.

Scene composition. Since our dynamic 3D Gaussian representation still uses pure 3D Gaussian representation, the learned representation facilitates straightforward editing of Gaussians. We demonstrate the composition of two scenes with our representation. Figure 7 illustrates this by combining the MUTANT scene from the D-NeRF dataset with the SEARED STEAK scene from the DyNeRF dataset. This demonstrates the capability of our method in editing dynamic 3D scenes.

5 Discussion and Conclusion

Limitations and future directions. Our dynamic Gaussians are defined through all times of the dynamic scene. This representation implicitly assumes Gaussians exist over all times of the scene. It enables us to naturally model the rigid and non-rigid deformation in the scene. On the other hand, for modeling the change in topology, the occurrence and extinction of Gaussians (*e.g.*, fluid) is tough. Static colors cause difficulty in modeling changes in illumination and color. The reconstruction capability of the method depends on the number of parameters, so that the scene representation is compact but results in poor rendering quality for very long sequences, requiring additional memory consumption for proper reconstruction. To overcome these limitations, considering the lifetime of Gaussians, such as adding start and end time parameters, will allow for the modeling of changes in scene topology, and the adaptive decision of flexibility will lead to better trade-offs between quality and memory size.

Our Gaussian representation sacrifices the continuity and smoothness inherent in neural field-based volume rendering. Distilling NeRFs into our proposed representation in a manner similar to PlenOctree [69] is a potential extension of our method, promising to enhance rendering quality while maintaining fast rendering advantage.

Conclusion. We present a compact dynamic 3D Gaussian representation enabling faithful reconstruction and real-time rendering of dynamic scenes. We propose a representation for the position and rotation of 3D Gaussians as a function of time for modeling the motion of the scene. The parameterized functions of time introduce memory efficiency and robustness to the number of views per timestep. Furthermore, we introduce the flow loss constraining the scene flow of the learned Gaussian representation with the ground truth flow. Our experiments on synthetic and real datasets show that the proposed method achieves real-time dynamic scene rendering even at high resolutions.

Acknowledgements This study was supported by JSPS/MEXT KAKENHI Grant Numbers JP24K20830, JP23KJ0381, JP23K28139, and JP22H05015, ROIS NII Open Collaborative Research 2024-24S1201, and the Institute of AI and Beyond of the University of Tokyo. The authors would like to thank D. Horita for carefully proofreading the manuscript and N. Umetani for providing helpful advice on the method’s limitations.

References

1. Abdal, R., Yifan, W., Shi, Z., Xu, Y., Po, R., Kuang, Z., Chen, Q., Yeung, D.Y., Wetzstein, G.: Gaussian Shell Maps for Efficient 3D Human Generation. In: CVPR. pp. 9441–9451 (2024)
2. Akhter, I., Sheikh, Y., Khan, S., Kanade, T.: Nonrigid Structure from Motion in Trajectory Space. In: NeurIPS (2008)
3. An, S., Xu, H., Shi, Y., Song, G., Ogras, U.Y., Luo, L.: PanoHead: Geometry-Aware 3D Full-Head Synthesis in 360 degree. In: CVPR. pp. 20950–20959 (2023)
4. Attal, B., Huang, J.B., Richardt, C., Zollhöfer, M., Kopf, J., O’Toole, M., Kim, C.: Hyper-Reel: High-Fidelity 6-DoF Video with Ray-Conditioned Sampling. In: CVPR. pp. 16610–16620 (2023)
5. Bae, J., Kim, S., Yun, Y., Lee, H., Bang, G., Uh, Y.: Per-Gaussian Embedding-Based Deformation for Deformable 3D Gaussian Splatting. arXiv preprint arXiv:2404.03613 (2024)
6. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In: CVPR. pp. 5855–5864 (2021)
7. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A Naturalistic Open Source Movie for Optical Flow Evaluation. In: ECCV. pp. 611–625 (2012)
8. Cabral, B., Max, N., Springmeyer, R.: Bidirectional Reflection Functions from Surface Bump Maps. SIGGRAPH **21**(4), 273–281 (1987)
9. Cao, A., Johnson, J.: HexPlane: A Fast Representation for Dynamic Scenes. In: CVPR. pp. 130–141 (2023)
10. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L.J., Tremblay, J., Khamis, S., et al.: Efficient Geometry-Aware 3D Generative Adversarial Networks. In: CVPR. pp. 16123–16133 (2022)
11. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: TensorRF: Tensorial Radiance Fields. In: ECCV. pp. 333–350 (2022)
12. Chen, Z., Wang, F., Liu, H.: Text-to-3D Using Gaussian Splatting. arXiv preprint arXiv:2309.16585 (2023)
13. Das, D., Wewer, C., Yunus, R., Ilg, E., Lenssen, J.E.: Neural Parametric Gaussians for Monocular Non-Rigid Object Reconstruction. In: CVPR. pp. 10715–10725 (2024)
14. Dong, Z., Chen, X., Yang, J., Black, M.J., Hilliges, O., Geiger, A.: AG3D: Learning to Generate 3D Avatars from 2D Image Collections. In: ICCV. pp. 14916–14927 (2023)
15. Duan, Y., Wei, F., Dai, Q., He, Y., Chen, W., Chen, B.: 4D-Rotor Gaussian Splatting: Towards Efficient Novel View Synthesis for Dynamic Scenes. ACM TOG (2024)
16. Duisterhof, B.P., Mandi, Z., Yao, Y., Liu, J.W., Shou, M.Z., Song, S., Ichnowski, J.: MD-Splatting: Learning Metric Deformation from 4D Gaussians in Highly Deformable Scenes. arXiv preprint arXiv:2312.00583 (2023)
17. Fang, J., Yi, T., Wang, X., Xie, L., Zhang, X., Liu, W., Nießner, M., Tian, Q.: Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. In: SIGGRAPH Asia (2022)
18. Fridovich-Keil, S., Meanti, G., Warburg, F.R., Recht, B., Kanazawa, A.: K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. In: CVPR. pp. 12479–12488 (2023)
19. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance Fields Without Neural Networks. In: CVPR. pp. 5501–5510 (2022)
20. Gan, W., Xu, H., Huang, Y., Chen, S., Yokoya, N.: V4D: Voxel for 4D Novel View Synthesis. IEEE TVCG **30**(2), 1579–1591 (2024)
21. Gao, C., Saraf, A., Kopf, J., Huang, J.B.: Dynamic View Synthesis from Dynamic Monocular Video. In: ICCV. pp. 5712–5721 (2021)
22. Guo, Z., Zhou, W., Li, L., Wang, M., Li, H.: Motion-Aware 3D Gaussian Splatting for Efficient Dynamic Scene Reconstruction. arXiv preprint arXiv:2403.11447 (2024)

23. He, H., Yang, Z., Li, S., Dai, B., Wu, W.: OrthoPlanes: A Novel Representation for Better 3D-Awareness of GANs. In: ICCV. pp. 22996–23007 (2023)
24. Huynh-Thu, Q., Ghanbari, M.: Scope of Validity of PSNR in Image/Video Quality Assessment. *Electronics Letters* **44**(13), 800–801 (2008)
25. Jouppi, N., Kurian, G., Li, S., Ma, P., Nagarajan, R., Nai, L., Patil, N., Subramanian, S., Swing, A., Towles, B., Young, C., Zhou, X., Zhou, Z., Patterson, D.A.: TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings. In: Proceedings of the 50th Annual International Symposium on Computer Architecture. pp. 1–14 (2023)
26. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG* **42**(4), 1–14 (2023)
27. Keselman, L., Hebert, M.: Approximate Differentiable Rendering with Algebraic Surfaces. In: ECCV. pp. 596–614 (2022)
28. Kopanas, G., Leimkühler, T., Rainer, G., Jambon, C., Drettakis, G.: Neural Point Catacaustics for Novel-View Synthesis of Reflections. *ACM TOG* **41**(6), 1–15 (2022)
29. Kopanas, G., Philip, J., Leimkühler, T., Drettakis, G.: Point-Based Neural Rendering with Per-View Optimization. In: *Comput. Graph. Forum.* vol. 40, pp. 29–43 (2021)
30. Lei, J., Wang, Y., Pavlakos, G., Liu, L., Daniilidis, K.: GART: Gaussian Articulated Template Models. In: CVPR. pp. 19876–19887 (2024)
31. Li, T., Slavcheva, M., Zollhöfer, M., Green, S., Lassner, C., Kim, C., Schmidt, T., Lovegrove, S., Goesele, M., Newcombe, R., et al.: Neural 3D Video Synthesis from Multi-View Video. In: CVPR. pp. 5521–5531 (2022)
32. Li, Z., Chen, Z., Li, Z., Xu, Y.: Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. *arXiv preprint arXiv:2312.16812* (2023)
33. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. In: CVPR. pp. 6498–6508 (2021)
34. Li, Z., Wang, Q., Cole, F., Tucker, R., Snavely, N.: DynIBaR: Neural Dynamic Image-Based Rendering. In: CVPR. pp. 4273–4284 (2023)
35. Liang, Y., Khan, N., Li, Z., Nguyen-Phuoc, T., Lanman, D., Tompkin, J., Xiao, L.: Gaufre: Gaussian Deformation Fields for Real-Time Dynamic Novel View Synthesis. *arXiv preprint arXiv:2312.11458* (2023)
36. Lu, Z., Guo, X., Hui, L., Chen, T., Yang, M., Tang, X., Zhu, F., Dai, Y.: 3D Geometry-Aware Deformable Gaussian Splatting for Dynamic View Synthesis. In: CVPR. pp. 8900–8910 (2024)
37. Lucas, B.D., Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. In: IJCAI. pp. 674–679 (1981)
38. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. In: 3DV (2024)
39. Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In: CVPR. pp. 4040–4048 (2016)
40. Menze, M., Geiger, A.: Object Scene Flow for Autonomous Vehicles. In: CVPR. pp. 3061–3070 (2015)
41. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In: ECCV (2020)
42. Müller, T., Evans, A., Schied, C., Keller, A.: Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM TOG* **41**(4), 1–15 (2022)
43. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable Neural Radiance Fields. In: ICCV. pp. 5865–5874 (2021)

44. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: HyperNeRF: a Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM TOG* **40**(6), 1–12 (2021)
45. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-NeRF: Neural Radiance Fields for Dynamic Scenes. In: *CVPR*. pp. 10318–10327 (2021)
46. Qian, S., Kirschstein, T., Schoneveld, L., Davoli, D., Giebenhain, S., Nießner, M.: GaussianAvatars: Photorealistic Head Avatars with Rigged 3D Gaussians. In: *CVPR*. pp. 20299–20309 (2024)
47. Qian, Z., Wang, S., Mihajlovic, M., Geiger, A., Tang, S.: 3DGS-Avatar: Animatable Avatars via Deformable 3D Gaussian Splatting. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 5020–5030 (2024)
48. Reiser, C., Peng, S., Liao, Y., Geiger, A.: KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs. In: *ICCV*. pp. 14335–14345 (2021)
49. Reiser, C., Szeliski, R., Verbin, D., Srinivasan, P., Mildenhall, B., Geiger, A., Barron, J., Hedman, P.: MERF: Memory-Efficient Radiance Fields for Real-Time View Synthesis in Unbounded Scenes. *ACM TOG* **42**(4), 1–12 (2023)
50. Schönberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M.: Pixelwise View Selection for Unstructured Multi-View Stereo. In: *ECCV*. pp. 501–518 (2016)
51. Schönberger, J.L., Frahm, J.M.: Structure-from-Motion Revisited. In: *CVPR* (2016)
52. Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In: *CVPR*. pp. 519–528 (2006)
53. Shao, R., Zheng, Z., Tu, H., Liu, B., Zhang, H., Liu, Y.: Tensor4D: Efficient Neural 4D Decomposition for High-Fidelity Dynamic Reconstruction and Rendering. In: *CVPR*. pp. 16632–16642 (2023)
54. Song, L., Chen, A., Li, Z., Chen, Z., Chen, L., Yuan, J., Xu, Y., Geiger, A.: NeRFPlayer: A Streamable Dynamic Scene Representation with Decomposed Neural Radiance Fields. *IEEE TVCG* **29**(5), 2732–2742 (2023)
55. Tang, J., Ren, J., Zhou, H., Liu, Z., Zeng, G.: DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation. *arXiv preprint arXiv:2309.16653* (2023)
56. Teed, Z., Deng, J.: RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In: *ECCV*. pp. 402–419 (2020)
57. Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P., Tretschk, E., Wang, Y., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S., Simon, T., Theobalt, C., Nießner, M., Barron, J.T., Wetzstein, G., Zollhöfer, M., Golyanik, V.: Advances in Neural Rendering. In: *Comput. Graph. Forum*. vol. 41, pp. 703–735 (2022)
58. Tian, F., Du, S., Duan, Y.: MonoNeRF: Learning a Generalizable Dynamic Radiance Field from Monocular Videos. In: *ICCV*. pp. 17903–17913 (2023)
59. Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., Theobalt, C.: Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene From Monocular Video. In: *ICCV* (2021)
60. Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., Theobalt, C.: Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene from Monocular Video. In: *ICCV*. pp. 12959–12970 (2021)
61. Vedula, S., Baker, S., Rander, P., Collins, R., Kanade, T.: Three-Dimensional Scene Flow. In: *CVPR*. pp. 722–729 (1999)
62. Vedula, S., Rander, P., Collins, R., Kanade, T.: Three-Dimensional Scene Flow. *IEEE TPAMI* **27**(3), 475–480 (2005)
63. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE TIP* **13**(4), 600–612 (2004)

64. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. arXiv preprint arXiv:2310.08528 (2023)
65. Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., Sridhar, S.: Neural Fields in Visual Computing and Beyond. *Comput. Graph. Forum* (2022)
66. Xu, D., Yuan, Y., Mardani, M., Liu, S., Song, J., Wang, Z., Vahdat, A.: AGG: Amortized Generative 3D Gaussians for Single Image to 3D. arXiv preprint arXiv:2401.04099 (2024)
67. Yang, Z., Yang, H., Pan, Z., Zhu, X., Zhang, L.: Real-Time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. arXiv preprint arXiv:2310.10642 (2023)
68. Yi, T., Fang, J., Wu, G., Xie, L., Zhang, X., Liu, W., Tian, Q., Wang, X.: GaussianDreamer: Fast Generation from Text to 3D Gaussian Splatting with Point Cloud Priors. arXiv preprint arXiv:2310.08529 (2023)
69. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: PlenOctrees for Real-Time Rendering of Neural Radiance Fields. In: *ICCV*. pp. 5752–5761 (2021)
70. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In: *CVPR*. pp. 586–595 (2018)
71. Zheng, E., Ji, D., Dunn, E., Frahm, J.M.: Sparse Dynamic 3D Reconstruction from Unsynchronized Videos. In: *CVPR*. pp. 4435–4443 (2015)
72. Zielonka, W., Bagautdinov, T., Saito, S., Zollhöfer, M., Thies, J., Romero, J.: Drivable 3D Gaussian Avatars. arXiv preprint arXiv:2311.08581 (2023)
73. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: EWA Splatting. *IEEE TVCG* **8**(3), 223–238 (2002)