7DGS: Unified Spatial-Temporal-Angular Gaussian Splatting

Zhongpai Gao*, Benjamin Planche, Meng Zheng, Anwesa Choudhuri, Terrence Chen, Ziyan Wu
United Imaging Intelligence, Boston MA, USA
{first.last}@uii-ai.com

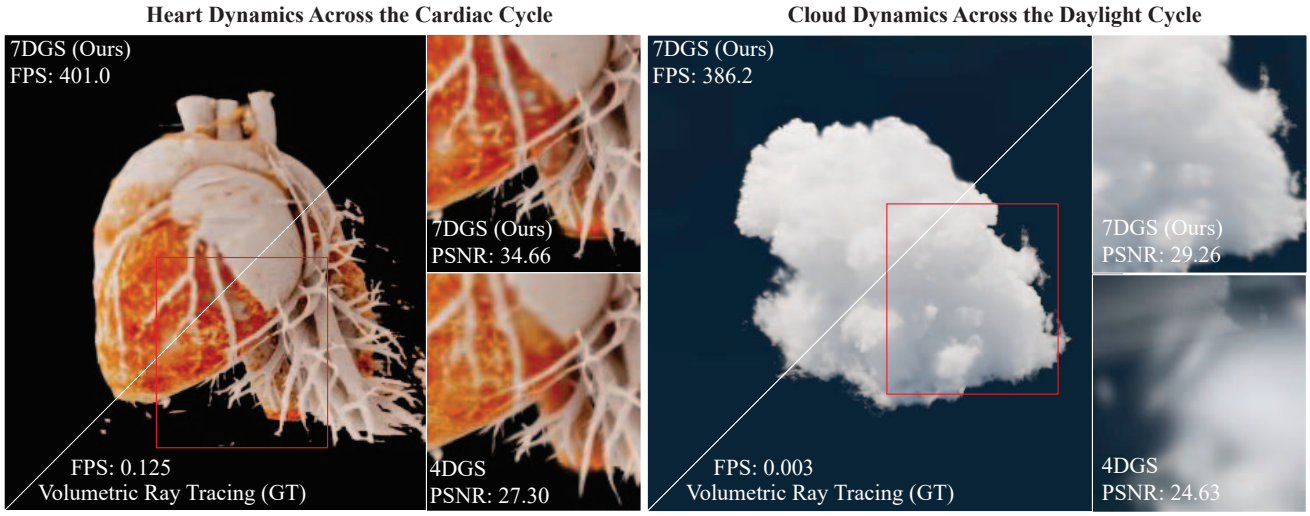

Figure 1. Visualization of volumetric rendering for dynamic scenes. **Top-left:** Our 7DGS rendering. **Bottom-left:** Physically-based rendering via ray/path tracing (note: floating artifacts in the `heart` scene are caused by incomplete segmentation in CT scans and are not rendering artifacts). **Right:** Comparison between our method and 4DGS in highlighted red regions.

Abstract

Real-time rendering of dynamic scenes with view-dependent effects remains a fundamental challenge in computer graphics. While recent advances in Gaussian Splatting have shown promising results separately handling dynamic scenes (4DGS) and view-dependent effects (6DGS), no existing method unifies these capabilities while maintaining real-time performance. We present 7D Gaussian Splatting (7DGS), a unified framework representing scene elements as seven-dimensional Gaussians spanning position (3D), time (1D), and viewing direction (3D). Our key contribution is an efficient conditional slicing mechanism that transforms 7D Gaussians into view- and time-conditioned 3D Gaussians, maintaining compatibility with existing 3D Gaussian Splatting pipelines while enabling joint optimization. Experiments demonstrate that 7DGS outperforms prior methods by up to 7.36 dB in PSNR while achieving real-time rendering (401 FPS) on challenging dynamic scenes with complex view-dependent effects. The project page is: gaozhongpai.github.io/7dgs/.

1. Introduction

Photorealistic rendering of dynamic scenes with complex view-dependent effects remains challenging in computer vision and graphics. Examples include dynamic heartbeat visualization from real CT scans and clouds transitioning across daylight with absorption and scattering effects (Figure 1). The ability to synthesize novel views of dynamic scenes is crucial for numerous applications, including virtual reality, augmented reality, content creation, and digital twins. While significant progress has been made in static scene reconstruction and rendering through Neural Radiance Fields (NeRF) [23] and more recently through 3D Gaussian Splatting (3DGS) [12], achieving high-quality, real-time rendering of dynamic scenes with view-dependent effects presents substantial computational and representational challenges.

The core difficulty lies in simultaneously modeling three fundamental aspects: 1) spatial geometry, 2) temporal dynamics, and 3) view-dependent appearance. Each of these dimensions introduces unique challenges. Spatial modeling must capture intricate scene geometry at varying scales.

Temporal modeling must represent both rigid and non-rigid motions with potentially complex deformations. View-dependent modeling needs to capture sophisticated light transport effects such as scattering, anisotropic reflections, and translucency. When considered together, these challenges become significantly more complex due to their interdependencies—for instance, specular highlights on moving objects change their appearance based on both viewing direction and object position over time.

Recent advances have addressed these challenges in isolation. 3DGS [12] introduced a breakthrough in static scene rendering by representing scenes as collections of 3D Gaussian primitives, enabling real-time rendering rates while maintaining high visual fidelity. However, this approach is inherently limited to static scenes. Two recent extensions have independently addressed different limitations of 3DGS: 4D Gaussian Splatting (4DGS) [38] incorporates temporal dynamics by extending the representation to 4D (space+time), while 6D Gaussian Splatting (6DGS) [7] models view-dependent effects by adding directional dimensions (space+direction). Despite their success in their respective domains, neither approach provides a comprehensive solution for dynamic scenes with view-dependent effects, as they address only subsets of the challenge.

In this paper, we present 7D Gaussian Splatting (7DGS), a unified framework for real-time rendering of dynamic scenes with view-dependent effects. Our key insight is to model scene elements as 7-dimensional Gaussians spanning spatial position (3D), time (1D), and viewing direction (3D). This high-dimensional representation naturally captures the interdependencies between geometry, dynamics, and appearance, enabling more accurate modeling of complex phenomena such as moving specular highlights and time-varying anisotropic reflections.

The primary technical challenge in our approach is efficiently handling 7D Gaussians while maintaining real-time performance. To address this, we introduce a principled conditional slicing mechanism that transforms 7D Gaussians into time- and view-conditioned 3D Gaussians compatible with existing real-time rendering pipelines. This operation preserves the computational efficiency of 3DGS while incorporating the rich representational capacity of our 7D model. Furthermore, we develop an adaptive Gaussian refinement technique that dynamically adjusts Gaussian parameters via neural network-predicted residuals, enabling more accurate modeling of complex non-rigid deformations and time-varying appearance.

We evaluate 7DGS on two public datasets: D-NeRF [27] (synthetic monocular videos) and Technicolor [29] (in-the-wild multi-view videos), and a custom dataset 7DGS-PBR with dynamic scenes featuring complex motions and view-dependent effects. Our results demonstrate that 7DGS consistently outperforms existing methods in terms of both rendering quality and computational efficiency. Our contributions can be summarized as follows:

- **Unified High-Dimensional Representation:** We introduce a novel 7D Gaussian model that jointly encodes spatial structure, temporal evolution, and view-dependent appearance. Furthermore, an adaptive Gaussian refinement technique is developed to enable more accurate modeling of complex deformations and time-varying appearance.
- **Efficient Conditional Slicing:** By deriving a principled conditional slicing mechanism, our method projects high-dimensional Gaussians into 3D counterparts that are compatible with existing real-time rendering pipelines, ensuring both efficiency and fidelity.
- **Validation:** Extensive experiments demonstrate that 7DGS outperforms the prior method 4DGS by up to 7.36 dB in PSNR while maintaining real-time rendering speeds (exceeding 401 FPS) on challenging dynamic scenes exhibiting complex view-dependent effects.

2. Related Work

Dynamic Neural Radiance Fields. NeRF [23] revolutionized novel view synthesis by representing scenes as continuous volumetric functions parameterized by neural networks. While the original NeRF focused on static scenes, numerous extensions [4, 9, 11, 18, 19, 30, 33, 36] have emerged for dynamic scene modeling. D-NeRF [27], Nerfies [25], and HyperNeRF [26] condition on time and learn deformation fields that warp points from canonical space to each time step. DyNeRF [15] represents scene dynamics using compact latent codes with a time-conditioned neural radiance field. To improve efficiency, HexPlane [2] accelerated dynamic NeRF rendering through hybrid representations. Despite these advances, NeRF-based methods generally struggle to achieve real-time performance when modeling complex dynamics and view-dependent effects.

Dynamic 3D Gaussian Splatting. 3DGS [12] represents scenes as collections of 3D Gaussians with learnable parameters, enabling high-quality rendering at real-time rates through efficient rasterization. Building on this foundation, several works [10, 17, 20, 28, 37] have extended 3DGS for dynamic scenes. 4DGS [38] incorporates temporal dynamics by extending Gaussians to a 4D (space+time) representation. Dynamic 3D Gaussians [21] and 4D Gaussians [35] jointly optimize Gaussians in canonical space alongside a deformation field to model scene geometry and dynamics. Ex4DGS [14] explicitly models the motions of 3D Gaussians using keyframe interpolation. While these approaches successfully address temporal aspects of dynamic scene modeling, they do not fully account for view-dependent effects within a unified framework.

View-dependent Rendering. For view-dependent effects, various methods have incorporated sophisticated

physically-based reflectance models into neural rendering pipelines. NeRV [31] introduced neural reflectance and visibility fields to capture view-dependent appearance. LFNR [32] proposed light field neural rendering for realistic view synthesis, while PhySG [39] incorporated physically-based BRDF models. In parallel, 6DGS [7] extended 3DGS to capture rich angular variations through 6D (space+direction) Gaussians. Recent work [1, 3, 8, 22, 24, 41] has also focused on integrating Gaussian primitives with ray tracing for more accurate light transport.

Our 7DGS method builds upon these prior works by unifying spatial, temporal, and angular dimensions into a single coherent framework. Unlike previous approaches that address temporal dynamics and view-dependent effects separately, 7DGS jointly models these dimensions through a unified 7D Gaussian representation, capturing their interdependencies while maintaining real-time performance.

3. Preliminary

In this section, we review two foundational methods that form the basis of our 7D Gaussian Splatting (7DGS) framework: 3D Gaussian Splatting (3DGS) [12] for static scene rendering, and its extension, 6D Gaussian Splatting (6DGS) [7], which incorporates view-dependent effects.

3D Gaussian Splatting. 3DGS represents a scene as a collection of anisotropic 3D Gaussians. Each Gaussian is defined by a mean vector $\mu \in \mathbb{R}^3$, which specifies its spatial position, and a covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$, which encodes the extent, shape, and orientation of the Gaussian. In practice, the covariance is factorized as

$$\Sigma = R\,S\,R^\top, \tag{1}$$

where $S = \mathrm{diag}(s_x, s_y, s_z)$ is a diagonal scaling matrix and R is a rotation matrix that aligns the Gaussian with the global coordinate system. This factorization provides an intuitive and compact way to represent local geometry.

In addition to geometry, each Gaussian carries an opacity α and view-dependent color information. The color is modeled via spherical harmonics:

$$c(d) = \sum_{\ell=0}^{N} \sum_{m=-\ell}^{\ell} \beta_{\ell m} Y_{\ell m}(d), \tag{2}$$

where N is the harmonics order ($N = 3$ typically), d denotes the viewing direction, $\beta_{\ell m}$ are learnable coefficients, and $Y_{\ell m}(d)$ are the spherical harmonic basis functions. This representation enables the model to capture complex appearance variations under different viewing angles while maintaining real-time rendering capabilities through efficient rasterization.

6D Gaussian Splatting. While 3DGS excels at static scene rendering, it does not account for the appearance changes induced by view-dependent effects. To overcome this limitation, 6D Gaussian Splatting extends the 3D representation by incorporating directional information. In 6DGS, each scene element is modeled as a 6D Gaussian defined over a joint space:

$$X = \begin{pmatrix} X_p \\ X_d \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_p \\ \mu_d \end{pmatrix}, \begin{pmatrix} \Sigma_p & \Sigma_{pd} \\ \Sigma_{pd}^\top & \Sigma_d \end{pmatrix} \right). \tag{3}$$

Here, $X_p \in \mathbb{R}^3$ represents the spatial coordinates with mean μ_p and covariance Σ_p, while $X_d \in \mathbb{R}^3$ encodes the directional component with mean μ_d and covariance Σ_d. The cross-covariance Σ_{pd} captures correlations between position and direction, allowing the Gaussian to encode view-dependent appearance variations.

For numerical stability and to guarantee positive definiteness, the full 6D covariance is parameterized via a Cholesky decomposition:

$$\Sigma = LL^\top, \tag{4}$$

with L being a lower-triangular matrix whose diagonal entries are enforced to be positive. To render an image for a given viewing direction d, the 6D Gaussian is conditioned on $X_d = d$, yielding a conditional 3D Gaussian for the spatial component. Specifically, the conditional distribution is given by:

$$p(X_p \mid X_d = d) \sim \mathcal{N}(\mu_{\text{cond}}, \Sigma_{\text{cond}}), \tag{5}$$

with

$$\mu_{\text{cond}} = \mu_p + \Sigma_{pd}\,\Sigma_d^{-1}(d - \mu_d), \tag{6}$$

$$\Sigma_{\text{cond}} = \Sigma_p - \Sigma_{pd}\,\Sigma_d^{-1}\,\Sigma_{pd}^\top. \tag{7}$$

Moreover, the opacity of each Gaussian is modulated to reflect the alignment between the current view direction and the Gaussian's preferred direction:

$$f_{\text{cond}} = \exp\left(-\lambda\,(d - \mu_d)^\top \Sigma_d^{-1}(d - \mu_d)\right), \tag{8}$$

$$\alpha_{\text{cond}} = \alpha \cdot f_{\text{cond}}, \tag{9}$$

where λ is a positive scaling parameter controlling the sensitivity of the modulation. This mechanism enhances the model's ability to capture view-dependent effects such as specular highlights and anisotropic reflections. However, note that both 3DGS and 6DGS are inherently designed for static scenes, as they do not incorporate temporal dynamics.

4. Our Approach

We introduce 7D Gaussian Splatting (7DGS), a unified framework that jointly models spatial, temporal, and angular dimensions. In 7DGS, each scene element is represented as a 7D Gaussian that naturally captures scene geometry, dynamics, and view-dependent appearance. By extending the Gaussian representation with an additional temporal dimension, 7DGS seamlessly integrates spatial, temporal, and angular variations, preserving the advantages of efficient real-time rendering and accurate view-dependent effects while robustly handling dynamic scenes.

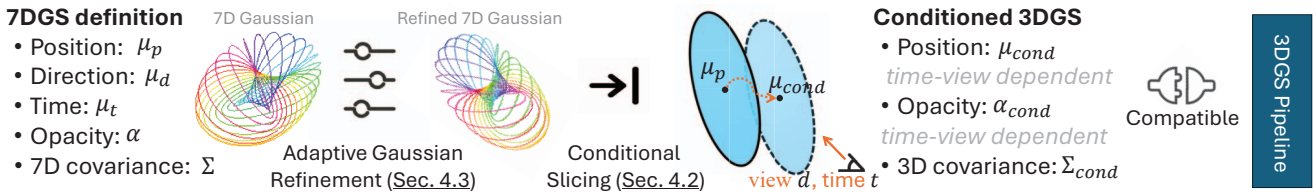

Figure 2. Proposed 7DGS compatible with the existing 3DGS pipeline.

4.1. 7D Gaussian Representation

In 7DGS, each scene element is modeled as a 7D Gaussian random variable that jointly encodes its spatial, temporal, and directional properties. This unified representation naturally captures not only the geometry of the scene but also its dynamics and view-dependent appearance. Formally, we define the 7D Gaussian as follows:

$$X = \begin{pmatrix} X_p \\ X_t \\ X_d \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_p \\ \mu_t \\ \mu_d \end{pmatrix}, \begin{pmatrix} \Sigma_p & \Sigma_{pt} & \Sigma_{pd} \\ \Sigma_{pt}^\top & \Sigma_t & \Sigma_{td} \\ \Sigma_{pd}^\top & \Sigma_{td}^\top & \Sigma_d \end{pmatrix} \right), \quad (10)$$

where:

- $X_p \in \mathbb{R}^3$ represents the spatial coordinates, with mean μ_p and covariance Σ_p that model the local geometric shape.
- $X_t \in \mathbb{R}$ is a scalar capturing the temporal coordinate, with mean μ_t and variance Σ_t. This component accounts for the dynamic evolution of scene elements.
- $X_d \in \mathbb{R}^3$ encodes the directional (angular) information, with mean μ_d and covariance Σ_d, which is critical for modeling view-dependent effects.

The off-diagonal blocks Σ_{pt}, Σ_{pd}, and Σ_{td} capture the correlations among the spatial, temporal, and directional components, enabling the Gaussian to model complex interdependencies across these dimensions.

Inspired by 6DGS, we parameterize the full 7D covariance matrix using a Cholesky decomposition:

$$\Sigma = LL^\top, \quad (11)$$

where L is a lower-triangular matrix with positive diagonal entries. This reparameterization not only guarantees a valid covariance matrix during optimization but also facilitates efficient computation.

For the color representation, we continue to adopt the view-dependent spherical harmonics formulation from 3DGS without introducing additional temporal dependencies, as the dynamic information is already encoded within the Gaussian parameters.

4.2. Conditional Slicing Mechanism

To render an image at a specified time t and from a given view direction d, we condition each 7D Gaussian on the observed temporal and angular values. This operation "slices" the full 7D Gaussian to yield a conditional 3D Gaussian that solely governs the spatial component. Such conditioning is critical because it allows us to efficiently integrate the temporal dynamics and view-dependent effects into the traditional 3D rendering pipeline.

We begin by partitioning the covariance matrix into two parts: $\Sigma_{(t,d)}$, corresponds to the temporal and directional dimensions, while the other, $\Sigma_{p,(t,d)}$, links the spatial dimension with the combined temporal-directional space:

$$\Sigma_{(t,d)} = \begin{pmatrix} \Sigma_t & \Sigma_{td} \\ \Sigma_{td}^\top & \Sigma_d \end{pmatrix}, \quad \text{and} \quad \Sigma_{p,(t,d)} = \begin{bmatrix} \Sigma_{pt} & \Sigma_{pd} \end{bmatrix}.$$

Here, Σ_t and Σ_d are the covariance matrices associated with the temporal and directional components, respectively, and Σ_{td} captures their mutual correlation. Similarly, Σ_{pt} and Σ_{pd} encode how the spatial component correlates with time and view direction.

Using the standard properties of multivariate Gaussian distributions, the conditional distribution of the spatial component X_p given $X_t = t$ and $X_d = d$ is also Gaussian:

$$p(X_p \mid X_t = t,\, X_d = d) \sim \mathcal{N}(\mu_{\text{cond}}, \Sigma_{\text{cond}}), \quad (12)$$

with conditional mean and covariance given by

$$\mu_{\text{cond}} = \mu_p + \Sigma_{p,(t,d)}\, \Sigma_{(t,d)}^{-1} \begin{pmatrix} t - \mu_t \\ d - \mu_d \end{pmatrix}, \quad (13)$$

$$\Sigma_{\text{cond}} = \Sigma_p - \Sigma_{p,(t,d)}\, \Sigma_{(t,d)}^{-1}\, \Sigma_{p,(t,d)}^\top. \quad (14)$$

In Equation (13), the term $\Sigma_{p,(t,d)}\, \Sigma_{(t,d)}^{-1} \begin{pmatrix} t - \mu_t \\ d - \mu_d \end{pmatrix}$ serves as a correction that shifts the spatial mean μ_p in accordance with deviations in time and view direction from their expected values μ_t and μ_d. Equation (14) similarly adjusts the spatial uncertainty Σ_p by removing the part of the variance explained by the temporal and directional components.

To further refine the rendering, we modulate the contribution of each Gaussian based on how much the observed time t and view direction d deviate from the Gaussian's expected values. We define two separate modulation factors:

$$f_{\text{temp}} = \exp\left(-\frac{1}{2}\, \lambda_t\, (t - \mu_t)^2\, \Sigma_t^{-1} \right), \quad (15)$$

$$f_{\text{dir}} = \exp\left(-\frac{1}{2}\, \lambda_d\, (d - \mu_d)^\top \Sigma_d^{-1} (d - \mu_d) \right), \quad (16)$$

where λ_t and λ_d are positive scalar parameters that control the sensitivity of the temporal and directional modulation, respectively. The factor f_{temp} decays exponentially as the observed time t diverges from the expected time μ_t, with the decay rate governed by λ_t. Similarly, the factor f_{dir} decreases as the view direction d moves away from the preferred direction μ_d.

The final conditional opacity for the Gaussian is then computed by combining the base opacity α with both modulation factors:

$$\alpha_{\text{cond}} = \alpha \cdot f_{\text{temp}} \cdot f_{\text{dir}}. \tag{17}$$

This formulation ensures that Gaussians contribute less to the rendered image when the current time or view direction is far from their expected values, thereby effectively integrating temporal dynamics and view-dependent appearance into the rendering process.

4.3. Adaptive Gaussian Refinement

While the conditional slicing mechanism in 7DGS adjusts the spatial mean μ_{cond} and modulates the opacity α_{cond} based on the current time t and view direction d, the intrinsic shape of each Gaussian—determined by its covariance—remains static over time. This limitation can hinder representing complex dynamic behaviors such as non-rigid deformations or motion-induced shape changes. To address this, we introduce an *Adaptive Gaussian Refinement* that dynamically updates the Gaussian parameters via residual corrections computed by lightweight neural networks.

Specifically, we first construct a comprehensive feature vector f that encapsulates the geometric and temporal context of each Gaussian. This feature vector is formed by concatenating the spatial mean μ_p, the temporal coordinate μ_t, the directional mean μ_d, and a high-frequency temporal encoding $\gamma(t)$:

$$f = \mu_p \oplus \mu_t \oplus \mu_d \oplus \gamma(t), \tag{18}$$

where $\oplus$ denotes vector concatenation. The temporal encoding $\gamma(t)$ is defined as

$$\gamma(t) = \Big(\sin(2^0\pi t),\, \cos(2^0\pi t),\, \ldots,\, \sin(2^{K-1}\pi t),\, \cos(2^{K-1}\pi t)\Big),$$

with $K = 10$. This multi-frequency encoding, inspired by positional encodings in [23], provides a rich representation of time that captures both low-frequency trends and high-frequency details.

Next, we employ a set of small two-layer multilayer perceptrons (MLPs) with architecture $C_{\text{in}} \times 64 \times C_{\text{out}}$ to predict residual adjustments for the key Gaussian parameters. These residuals are added to the original parameters to yield refined estimates:

$$\begin{aligned} \hat{\mu}_p &= \mu_p + \phi_p(f), \quad \hat{\mu}_t = \mu_t + \phi_t(f), \\ \hat{\mu}_d &= \mu_d + \phi_d(f), \quad \hat{l} = l + \phi_l(f). \end{aligned} \tag{19}$$

Here, l represents the vectorized lower-triangular elements of the 7D covariance matrix, and $\phi_p(f)$, $\phi_t(f)$, $\phi_d(f)$, and $\phi_l(f)$ are the residuals predicted by the respective MLPs. These updates allow the spatial position, temporal coordinate, directional mean, and covariance (which controls rotation and shape) to be dynamically adjusted as functions of the observed time.

This refinement module is applied before the conditional slicing step (Section 4.2). By dynamically adapting the 7D Gaussian parameters, the subsequent conditioning produces a 3D Gaussian whose spatial attributes—including its shape and orientation—more accurately reflect the evolving scene dynamics and view-dependent variations. This leads to improved modeling of complex motions and a more faithful reconstruction of dynamic scenes.

Algorithm 1 Slice 7DGS to Conditional 3DGS

Input: Lower-triangular L, μ_p, μ_t, μ_d, base opacity α, scaling factors λ_t, λ_d, view direction d, observed time t

Output: Conditional μ_{cond}, Σ_{cond}, α_{cond} (optionally, scale S and rotation R are required for densification steps)

1: Compute feature: $f = \text{concat}\big(\mu_p,\, \mu_t,\, \mu_d,\, \gamma(t)\big)$.

2: Adaptive Gaussian refinement:

$$\begin{aligned} \hat{\mu}_p &= \mu_p + \phi_p(f), \quad \hat{\mu}_t = \mu_t + \phi_t(f), \\ \hat{\mu}_d &= \mu_d + \phi_d(f), \quad \hat{l} = l + \phi_l(f). \end{aligned}$$

3: Reconstruct refined covariance: $\hat{\Sigma} = \hat{L}\hat{L}^\top$.

4: Partition $\hat{\Sigma}$ into blocks:

$$\hat{\Sigma} = \begin{pmatrix} \hat{\Sigma}_p & \hat{\Sigma}_{p,(t,d)} \\ \hat{\Sigma}_{p,(t,d)}^\top & \hat{\Sigma}_{(t,d)} \end{pmatrix}, \text{with } \hat{\Sigma}_{(t,d)} = \begin{pmatrix} \hat{\Sigma}_t & \hat{\Sigma}_{td} \\ \hat{\Sigma}_{td}^\top & \hat{\Sigma}_d \end{pmatrix}.$$

5: Compute conditional statistics:

$$\begin{aligned} \Sigma_{\text{cond}} &= \hat{\Sigma}_p - \hat{\Sigma}_{p,(t,d)}\, \hat{\Sigma}_{(t,d)}^{-1}\, \hat{\Sigma}_{p,(t,d)}^\top, \\ \mu_{\text{cond}} &= \hat{\mu}_p + \hat{\Sigma}_{p,(t,d)}\, \hat{\Sigma}_{(t,d)}^{-1} \begin{pmatrix} t - \hat{\mu}_t \\ d - \hat{\mu}_d \end{pmatrix}. \end{aligned}$$

6: Compute conditional opacity:

$$\begin{aligned} f_{\text{temp}} &= \exp\Big(-\tfrac{1}{2}\,\lambda_t\,(t - \hat{\mu}_t)^2\,\hat{\Sigma}_t^{-1}\Big), \\ f_{\text{dir}} &= \exp\Big(-\tfrac{1}{2}\,\lambda_d\,(d - \hat{\mu}_d)^\top \hat{\Sigma}_d^{-1}(d - \hat{\mu}_d)\Big), \\ \alpha_{\text{cond}} &= \alpha \cdot f_{\text{temp}} \cdot f_{\text{dir}} \end{aligned}$$

7: **Optional:** Perform SVD on $\Sigma_{\text{cond}} = UDU^\top$ to extract scale $S = \sqrt{\text{diag}(D)}$ and rotation $R = U$ (adjust R to ensure $\det(R) > 0$) on desification steps.

4.4. Optimization and Rendering Pipeline

Our optimization strategy extends the adaptive Gaussian densification framework of 3DGS to the enriched spatio-temporal-angular domain of 7DGS. In our method, each Gaussian is dynamically adjusted via cloning and splitting operations, ensuring comprehensive coverage across spatial, temporal, and directional dimensions.

To guide these refinement operations, we first extract scale and rotation information from the conditional covariance matrix Σ_{cond} (obtained after conditioning on the observed time t and view direction d). We perform a Singular Value Decomposition: $\Sigma_{\text{cond}} = U\,D\,U^\top$, where U is an orthogonal matrix and D is a diagonal matrix containing the singular values. We then define the rotation matrix as $R = U$ and compute the scale vector as $S = \sqrt{\text{diag}(D)}$. To ensure that R represents a right-handed coordinate system, we

adjust its last column as follows: $R_{:,3} = R_{:,3} \cdot \text{sign}(\det(R))$.

For Gaussian splitting, 7DGS leverages temporal cues in addition to spatial gradients. We quantify the spatial-temporal correlation using the magnitude of the off-diagonal block Σ_{pt}, which captures the interaction between spatial and temporal components. When this correlation exceeds a threshold of 0.05 (relative to the screen extent) and the normalized temporal scale (derived from Σ_t) is larger than 0.25, the corresponding Gaussians are split. This criterion ensures that regions with significant motion dynamics are densely represented.

The rendering pipeline remains fully compatible with 3DGS. In our approach, the 7DGS representation is first converted into a 3DGS-compatible format via the conditional slicing mechanism (see Section 4.2 and Algorithm 1). The resulting conditional 3D Gaussian's mean and covariance are then projected onto the image plane using standard perspective projection, yielding a set of 2D Gaussians. These 2D Gaussians are subsequently splatted onto the image canvas using a differentiable rasterization routine, and the final pixel colors are computed by aggregating the contributions of all Gaussians in a depth-aware, opacity-blended manner.

Importantly, our 7DGS framework integrates seamlessly with the existing 3DGS training pipeline. We employ the same loss functions, optimizers, and hyperparameter settings—with the only modification being an increased minimum opacity threshold ($\tau_{\text{min}} = 0.01$) for pruning, which compensates for the modulation of the conditional opacity α_{cond} by time and view direction. By converting our 7D representation into a conditional 3D format, we fully leverage the adaptive density control and efficient rasterization techniques of 3DGS, thereby achieving enhanced performance with minimal modifications.

5. Experiments

5.1. Experimental Protocol

Datasets. We evaluate 7DGS on three distinct datasets:

- **D-NeRF** [27]: A synthetic monocular video dataset containing eight scenes at a resolution of 800×800.
- **Technicolor** [29]: An in-the-wild dataset composed of video recordings captured by a synchronized 4×4 camera array at 2048×1088 resolution.
- **7DGS-PBR**: Our custom dataset, rendered using physically-based techniques, consists of six dynamic scenes exhibiting complex view-dependent effects:
  - `heart1` and `heart2`: Derived from real CT scans, these scenes capture cardiac cycles over 15 timestamps.
  - `cloud`: Based on the Walt Disney Animation Studios volumetric cloud dataset[1], this scene features a complete daylight cycle spanning 60 timestamps.
  - `dust` and `flame`: Sourced from Blender Market[2], these scenes present dynamic volumetric effects over 79 and 101 timestamps, respectively.
  - `suzanne`: the standard Blender test mesh rendered with a translucent "*Glass BSDF*" material, showing jelly-like deformations across 60 timestamps.

For each timestamp, we sampled 300, 60, 20, 10, 10, and 10 views for `heart1`, `heart2`, `cloud`, `dust`, `flame`, and `suzanne`, respectively, following a 9:1 train-test split.

All scenes were rendered using Blender's Cycles engine at a resolution of 1000×1000 for `heart1` and 1600×1600 for the remaining scenes. Average rendering times per view on an NVIDIA Tesla V100 GPU were 8 seconds for `heart1`, 18 seconds for `heart2`, 311 seconds for `cloud`, 16 seconds for `dust`, 28 seconds for `flame`, and 26 seconds for `suzanne`. We will make the 7DGS-PBR dataset publicly available.

Evaluation Metrics. We evaluate our method using three image quality metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM) [34], and LPIPS [40]. For efficiency, we report the number of Gaussian points, rendering speed (FPS), and training time (minutes).

Implementation Details. We adopt the 4DGS configuration with a batch size of 4 and downscaling factor of 2, except for Technicolor [29] where we use no downscaling and reduce batch size to 1 for 7DGS experiments. The directional and temporal modulation parameters (λ_d and λ_t) are initially set to 0.5 for 7DGS-PBR, 0.05 for D-NeRF, and 0.1 for Technicolor, based on their respective directional and temporal dependencies, and become trainable after 15,000 iterations. Point clouds for `heart1` and `heart2` are initialized using marching cubes following DDGS [6], while other scenes use 100,000 randomly initialized points within a bounding cube. For Technicolor, we initialize from COLMAP sparse reconstructions.

All experiments are conducted on a single NVIDIA Tesla V100 GPU (16GB memory) using the Adam optimizer [13]. We employ distinct learning rates for different parameter groups: 2.5×10^{-2} for temporal and directional means (μ_t, μ_d), 5×10^{-2} for covariance diagonals, 1×10^{-2} for lower triangular covariance elements, and 2×10^{-4} for adaptive Gaussian network parameters (which become trainable after 3,000 iterations). All remaining parameters follow the default 3DGS learning rates.

5.2. Comparison with Baseline

Table 1 presents a comprehensive comparison between our 7DGS framework and the state-of-the-art 4DGS method across all three datasets. We evaluate both the full 7DGS implementation and a variant without the adaptive Gaussian refinement (AGR) component to isolate the contribution of our core 7D representation.

[1] Disney Animation Studios cloud dataset

[2] Animated VDB pack dataset

Dataset	Scene	4DGS						7DGS (**Ours**)						7DGS (**Ours**, w/o AGR)				
		PSNR↑	SSIM↑	LPIPS↓	Train↓	FPS↑	# points↓	PSNR↑	SSIM↑	LPIPS↓	Train↓	FPS↑	# points↓	PSNR↑	SSIM↑	LPIPS↓	FPS↑	# points↓
7DGS-PBR	heart1	27.30	0.949	0.046	**103.0**	186.9	694,006	**35.48**	**0.986**	**0.020**	114.2	155.7	82,813	34.66	0.983	0.023	**401.0**	**82,412**
	heart2	25.13	0.920	0.084	**103.4**	160.4	869,245	**31.80**	**0.964**	**0.051**	145.9	139.5	98,458	30.99	0.959	0.057	**384.6**	**101,503**
	cloud	24.63	0.938	0.100	123.7	219.0	216,878	**29.60**	**0.955**	**0.075**	**102.6**	199.2	44,858	29.29	0.955	0.075	**386.2**	**44,175**
	dust	35.88	0.954	0.037	97.0	296.1	357,744	**37.30**	**0.956**	**0.037**	**69.8**	243.9	11,253	36.87	0.955	0.038	**394.8**	**10,924**
	flame	29.34	0.928	0.067	113.7	151.2	947,786	**32.53**	**0.940**	**0.059**	**74.1**	247.2	16,544	31.67	0.937	0.062	**371.6**	**15,060**
	suzanne	24.45	0.917	0.141	222.5	141.8	766,098	**28.26**	**0.949**	**0.062**	**193.9**	62.5	336,713	27.14	0.942	0.074	**317.9**	**276,281**
	avg	27.79	0.934	0.079	127.2	192.6	641,960	**32.50**	**0.958**	**0.051**	**116.7**	174.7	98,440	31.77	0.955	0.055	**376.0**	**88,393**
D-NeRF	b.balls	33.24	0.982	0.025	**50.7**	219.9	276,073	**35.10**	**0.984**	**0.019**	86.6	104.3	129,791	34.05	0.982	0.025	**213.1**	**127,395**
	h.warrior	**34.10**	**0.949**	**0.067**	35.3	299.4	298,391	32.96	0.935	0.084	**31.3**	238.3	9,569	32.78	0.934	0.090	**431.5**	**8,693**
	hook	**32.93**	**0.970**	**0.034**	38.0	325.1	174,720	31.57	0.962	0.040	**35.7**	233.1	24,700	30.95	0.958	0.045	**432.8**	**21,662**
	j.jacks	31.14	0.970	0.044	66.9	366.0	143,665	**33.57**	**0.977**	**0.027**	**34.1**	243.2	18,784	31.37	0.967	0.042	**432.4**	**15,779**
	lego	25.58	0.917	0.077	**55.4**	320.0	186,165	**28.86**	**0.947**	**0.051**	78.5	160.2	74,884	28.72	0.947	0.051	**365.0**	**68,552**
	mutant	39.01	0.991	0.009	**39.1**	341.6	138,691	**41.36**	**0.995**	**0.005**	42.5	193.7	37,706	39.59	0.993	0.007	**395.8**	**33,868**
	standup	39.75	0.991	0.008	34.4	330.4	142,468	**40.60**	**0.992**	**0.008**	**33.5**	224.0	15,598	38.45	0.988	0.014	**399.2**	**12,688**
	trex	29.89	0.979	0.021	100.7	169.2	682,378	**30.72**	**0.980**	**0.018**	**63.4**	156.5	67,994	30.13	0.979	0.021	**352.4**	**61,946**
	avg	33.21	0.969	0.036	52.6	296.4	255,319	**34.34**	**0.972**	**0.032**	**50.7**	194.2	47,378	33.26	0.969	0.037	**377.8**	**43,823**
Technicolor	birthday	31.28	0.922	0.153	370.8	69.6	842,491	**32.31**	**0.940**	**0.111**	**117.0**	39.2	589,128	32.01	0.937	0.116	**237.1**	**622,437**
	fabien	**35.48**	**0.894**	**0.297**	332.8	110.5	705,106	34.87	0.885	0.317	**73.0**	131.3	107,240	34.53	0.876	0.336	**354.7**	**91,237**
	painter	35.09	0.905	0.238	361.9	99.4	287,036	**36.54**	**0.919**	**0.208**	**98.1**	113.1	144,226	36.46	0.914	0.216	**364.0**	**119,654**
	theater	**31.84**	0.871	0.291	383.7	83.0	946,909	31.54	**0.876**	**0.265**	**87.3**	94.0	197,713	31.09	0.873	0.271	**333.2**	**185,330**
	train	32.58	0.932	0.102	345.3	58.3	1,412,917	**32.64**	**0.940**	**0.089**	**185.2**	18.6	1,043,645	32.43	0.938	0.090	**100.7**	**1,014,529**
	avg	33.25	0.905	0.216	358.9	84.2	838,892	**33.58**	**0.912**	**0.198**	**112.1**	79.2	416,390	33.30	0.908	0.206	**278.0**	**406,637**

Table 1. Comparison with 4DGS [38] on 7DGS-PBR, D-NeRF [27], and Technicolor [29].'*Train*' means training time in minutes.

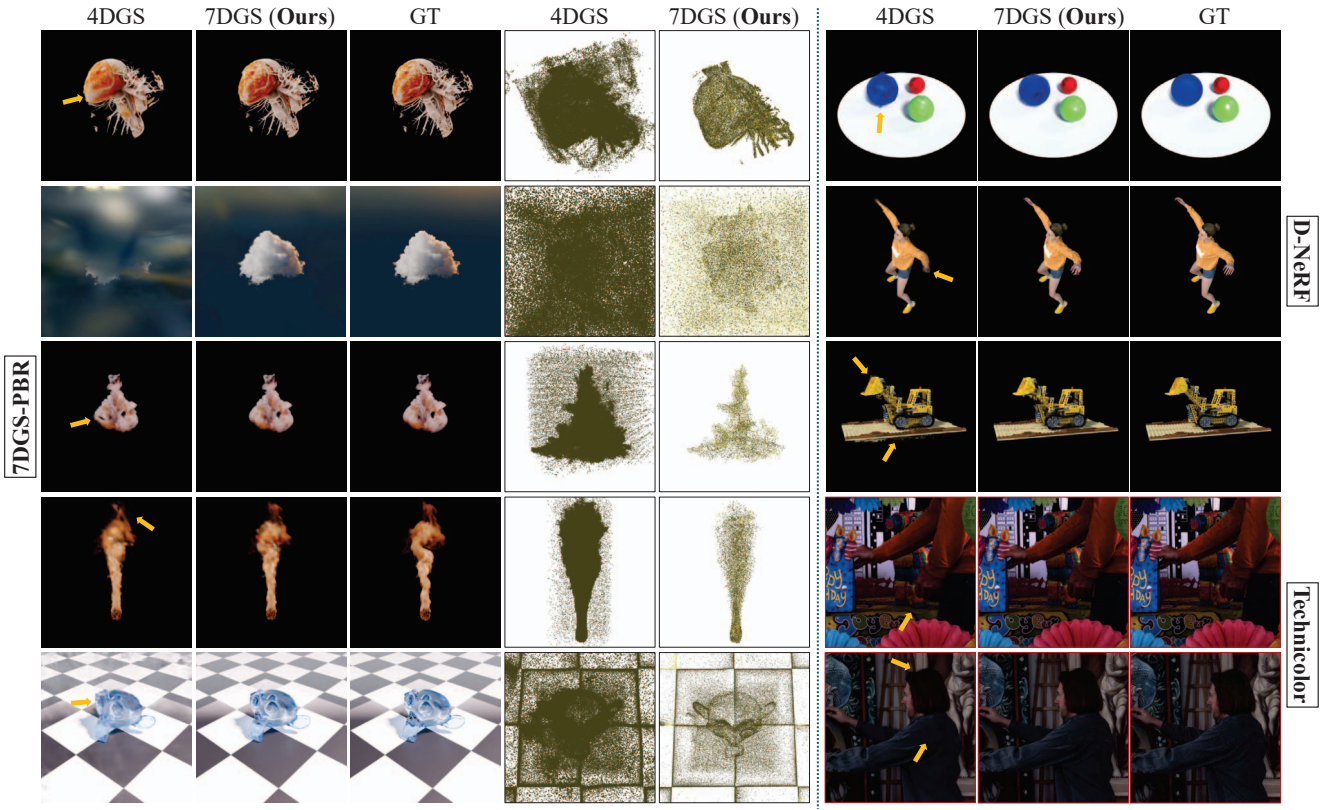

Figure 3. Qualitative comparison of methods on the 7DGS-PBR, D-NeRF [27], and Technicolor [29] datasets (zoom in for details).

Our 7DGS consistently outperforms 4DGS across all evaluation metrics and datasets. On 7DGS-PBR, which specifically targets complex view-dependent effects, our method achieves remarkable improvements with an average PSNR gain of +4.71 dB (from 27.79 dB to 32.50 dB) while utilizing only 15.3% of the Gaussian points required by 4DGS (98,440 vs. 641,960). The most substantial improvement is observed on the `heart1` scene, where 7DGS delivers an impressive +8.18 dB PSNR increase while requiring only 11.9% of the Gaussian points used by 4DGS. In addition, our method reduces the training time by an average of 8.3% and can even further speed up by implement-

ing our 7DGS slicing in Algorithm 1 in CUDA.

On the D-NeRF dataset [27], 7DGS maintains its superior performance with an average PSNR improvement of +1.13 dB while using only 18.6% of the Gaussian points. Similarly, on the challenging in-the-wild Technicolor dataset [29], 7DGS delivers superior results with an average PSNR gain of +0.33 dB while requiring approximately half the number of Gaussian points.

Notably, even without the adaptive Gaussian refinement (AGR), our 7DGS (w/o AGR) variant still outperforms 4DGS with an average PSNR gain of +3.98 dB on 7DGS-PBR, +0.05 dB on D-NeRF, and +0.05 dB on Technicolor, while using significantly fewer Gaussian points (13.8%, 17.2%, and 48.5% respectively). Additionally, the removal of AGR substantially accelerates rendering speed, achieving an average of 376.0 FPS, 377.8 FPS, and 278.0 FPS on the three datasets—approximately twice the rendering speed of the full 7DGS implementation and substantially faster than 4DGS.

Figure 3 provides visual comparisons of novel view renderings alongside visualizations of the reconstructed point clouds. The qualitative results reveal that 4DGS exhibits more pronounced artifacts, particularly for scenes with complex view-dependent effects. Furthermores, 7DGS produces cleaner, more faithful geometric reconstructions with superior handling of temporal dynamics and view-dependent appearance variations. The improvement is especially noticeable in scenes with complex lighting interactions, such as the translucent `suzanne` and the volumetric `cloud` scenes, where our unified spatio-temporal-angular representation effectively captures the interdependence between geometry, motion, and appearance.

5.3. Comparison with State-of-the-Art

Table 2 presents a comprehensive comparison between our method and other state-of-the-art approaches on the D-NeRF [27] and Technicolor [29] datasets. On the D-NeRF dataset, 7DGS substantially outperforms all existing methods in terms of PSNR, achieving a score of 34.34 dB, which represents a +1.04 dB improvement over 4DGaussians [35] and +1.13 dB over 4DGS [38]. While 7DGS achieves a competitive SSIM of 0.97 (matching 4DGS and DaReNeRF), 4DGaussians slightly leads with 0.98. For LPIPS, our method ties for best performance with DaReNeRF and 4DGaussians at 0.03.

For the challenging Technicolor dataset, which features in-the-wild multi-view videos, our method achieves state-of-the-art results with a PSNR of 33.58 dB. In terms of SSIM, our score of 0.912 is competitive with the best-performing Ex4DGS (0.917) and matches STG exactly. While our LPIPS score of 0.101 is slightly higher (worse) than STG's leading 0.085, it represents an improvement over several other methods including 4DGS (0.110) and 4DGaussians (0.178).

	Method	PSNR↑	SSIM↑	LPIPS↓
D-NeRF	D-NeRF [27]	29.67	0.95	0.07
	HexPlane [2]	31.05	0.97	0.04
	K-Planes [5]	31.61	0.97	-
	DaReNeRF [19]	31.95	0.97	**0.03**
	4DGS* [38]	33.21	0.97	0.04
	4DGaussians [35]	33.30	**0.98**	**0.03**
	7DGS (**Ours**)	**34.34**	0.97	**0.03**
	Method	PSNR↑	SSIM↑	LPIPS-Alex↓
Technicolor	DyNeRF [15]	31.80	-	0.142
	HyperReel [26]	32.73	0.906	0.109
	4DGaussians [35]	30.79	0.843	0.178
	STG [16]	33.23	0.912	**0.085**
	4DGS* [38]	33.25	0.905	0.110
	Ex4DGS* [14]	33.49	**0.917**	0.094
	7DGS (**Ours**)	**33.58**	0.912	0.101

Table 2. Comparison with SOTA methods on benchmarks. Methods with '*' are reproduced results with the official codes. Note, SOTA methods only have 2-digit precision for LPIPS in D-NeRF.

The performance advantages across both datasets demonstrate the effectiveness of our unified 7D representation in handling diverse dynamic scenes. Furthermore, our 7DGS is inherently flexible and can be integrated with complementary techniques to further enhance performance. For instance, while Ex4DGS [14] employs keyframe interpolation to explicitly model large-scale motion, similar strategies could be incorporated into 7DGS as future work. The modular nature of our approach allows for such extensions without compromising its core unified representation of spatial, temporal, and angular dimensions. This versatility positions 7DGS as not just a standalone improvement but as a fundamental advancement that can serve as a foundation for future research in dynamic scene rendering.

6. Conclusion

We present 7DGS, a novel framework that unifies spatial, temporal, and angular dimensions into a single 7D Gaussian representation for dynamic scene rendering. Our conditional slicing mechanism efficiently projects 7D Gaussians into renderable 3D Gaussians, enabling both high-quality results and real-time performance. Experiments across three datasets demonstrate that 7DGS outperforms state-of-the-art methods by up to 7.36 dB PSNR while using significantly fewer Gaussian points and maintaining render speeds of over 400 FPS (without adaptive refinement). Our approach excels particularly on scenes with complex view-dependent effects, advancing the field toward unified and efficient dynamic scene representations.

References

[1] Hugo Blanc, Jean-Emmanuel Deschaud, and Alexis Paljic. Raygauss: Volumetric gaussian-based ray casting for photorealistic novel view synthesis. *arXiv preprint arXiv:2408.03356*, 2024. 3

[2] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 2, 8

[3] Jorge Condor, Sebastien Speierer, Lukas Bode, Aljaz Bozic, Simon Green, Piotr Didyk, and Adrian Jarabo. Don't splat your gaussians: Volumetric ray-traced primitives for modeling and rendering scattering and emissive media, 2024. 3

[4] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 2

[5] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 8

[6] Zhongpai Gao, Benjamin Planche, Meng Zheng, Xiao Chen, Terrence Chen, and Ziyan Wu. Ddgs-ct: Direction-disentangled gaussian splatting for realistic volume rendering. *Advances in Neural Information Processing Systems*, 2024. 6

[7] Zhongpai Gao, Benjamin Planche, Meng Zheng, Anwesa Choudhuri, Terrence Chen, and Ziyan Wu. 6dgs: Enhanced direction-aware gaussian splatting for volumetric rendering. *arXiv preprint arXiv:2410.04974*, 2024. 2, 3

[8] Shrisudhan Govindarajan, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Radiant foam: Real-time differentiable ray tracing. *arXiv preprint arXiv:2502.01157*, 2025. 3

[9] Xiang Guo, Jiadai Sun, Yuchao Dai, Guanying Chen, Xiaoqing Ye, Xiao Tan, Errui Ding, Yumeng Zhang, and Jingdong Wang. Forward flow for novel view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16022–16033, 2023. 2

[10] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4220–4230, 2024. 2

[11] Erik Johnson, Marc Habermann, Soshi Shimada, Vladislav Golyanik, and Christian Theobalt. Unbiased 4d: Monocular 4d reconstruction with a neural deformation model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6598–6607, 2023. 2

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 2, 3

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[14] Junoh Lee, ChangYeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. Fully explicit dynamic gaussian splatting. *Advances in Neural Information Processing Systems*, 37:5384–5409, 2024. 2, 8

[15] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5521–5531, 2022. 2, 8

[16] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8508–8520, 2024. 8

[17] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21136–21145, 2024. 2

[18] Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13–23, 2023. 2

[19] Ange Lou, Benjamin Planche, Zhongpai Gao, Yamin Li, Tianyu Luan, Hao Ding, Terrence Chen, Jack Noble, and Ziyan Wu. Darenerf: Direction-aware representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5031–5042, 2024. 2, 8

[20] Zhicheng Lu, Xiang Guo, Le Hui, Tianrui Chen, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3d geometry-aware deformable gaussian splatting for dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8900–8910, 2024. 2

[21] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 2

[22] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T Barron, and Yinda Zhang. Ever: Exact volumetric ellipsoid rendering for real-time view synthesis. *arXiv preprint arXiv:2410.01804*, 2024. 3

[23] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Computer Vision–ECCV 2020*, pages 405–421, 2020. 1, 2, 5

[24] Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *arXiv preprint arXiv:2407.07090*, 2024. 3

[25] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo

Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5865–5874, 2021. 2

[26] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 2, 8

[27] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10318–10327, 2021. 2, 6, 7, 8

[28] Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5020–5030, 2024. 2

[29] Neus Sabater, Guillaume Boisson, Benoit Vandame, Paul Kerbiriou, Frederic Babon, Matthieu Hog, Remy Gendrot, Tristan Langlois, Olivier Bureller, Arno Schubert, et al. Dataset and pipeline for multi-view light-field video. In *Proceedings of the IEEE conference on computer vision and pattern recognition Workshops*, pages 30–40, 2017. 2, 6, 7, 8

[30] Liangchen Song, Xuan Gong, Benjamin Planche, Meng Zheng, David Doermann, Junsong Yuan, Terrence Chen, and Ziyan Wu. Pref: Predictability regularized neural motion fields. In *European Conference on Computer Vision*, pages 664–681. Springer, 2022. 2

[31] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7495–7504, 2021. 3

[32] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Light field neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8269–8279, 2022. 3

[33] Chaoyang Wang, Lachlan Ewen MacDonald, Laszlo A Jeni, and Simon Lucey. Flow supervision for deformable nerf. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21128–21137, 2023. 2

[34] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 6

[35] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20310–20320, 2024. 2, 8

[36] Zhiwen Yan, Chen Li, and Gim Hee Lee. Nerf-ds: Neural radiance fields for dynamic specular objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8285–8295, 2023. 2

[37] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20331–20341, 2024. 2

[38] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *International Conference on Learning Representations (ICLR)*, 2024. 2, 7, 8

[39] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5453–5462, 2021. 3

[40] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

[41] Yang Zhou, Songyin Wu, and Ling-Qi Yan. Unified gaussian primitives for scene representation and rendering. *arXiv preprint arXiv:2406.09733*, 2024. 3