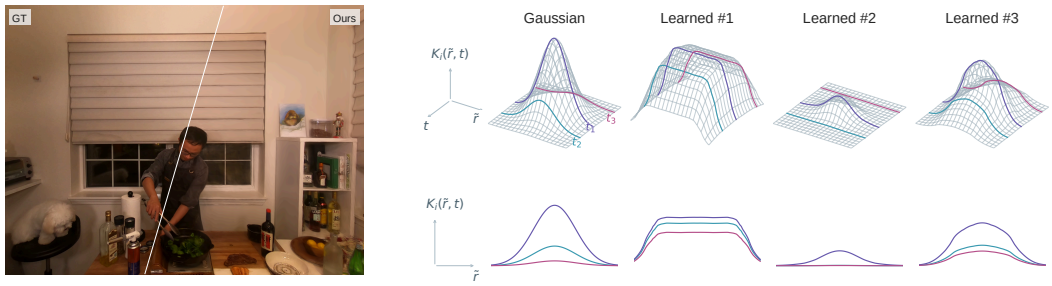
LEARNING 4D SPLATTING KERNELS

Anonymous authors
Paper under double-blind review



(a) Rendering results (b) Learned kernels

Figure 1: We present a differentiable framework to learn spatial and temporal kernels in a 4D splatting-based pipeline for dynamic novel-view synthesis. (a) Ground truth and our rendering of a representative dynamic scene. (b) For each primitive, we denote the spatio-temporal kernel by $K_i(\tilde{r}, t)$ (Eq. 13), where $\tilde{r}$ is the rescaled spatial Mahalanobis distance and t is the query time. The top row visualizes an analytical Gaussian reference and three representative learned kernels as wireframe surfaces of $K_i(\tilde{r}, t)$. The highlighted curves mark three temporal slices, with the corresponding 1D profiles along $\tilde{r}$ shown in the bottom row.

ABSTRACT

We present a differentiable framework for learning spatial and temporal kernels directly from data in a 4D splatting-based pipeline for dynamic novel-view synthesis. Each primitive is equipped with spatial and temporal latents. The spatial latent and time-conditioned 3D attributes are mapped to a radially symmetric 2D kernel through neural projection and decoding, while a separate decoder maps the temporal latent to a temporal kernel. The networks are jointly optimized with 4D primitive attributes, including the latents. The learned kernels adapt to scene structure and dynamics, exhibiting diverse profiles beyond Gaussian forms. The effectiveness of our approach is demonstrated on standard dynamic novel-view synthesis benchmarks, comparing favorably against state-of-the-art techniques.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) has become an efficient explicit representation for novel-view synthesis through differentiable splatting. Recent methods extend 3DGS to dynamic scenes through deformation-based representations (Yang et al., 2024b; Wu et al., 2024), explicit motion modeling (Luiten et al., 2024; Li et al., 2024), or native 4D Gaussian primitives (Yang et al., 2024a; Duan et al., 2024). In particular, 4D Gaussian Splatting (4DGS) (Yang et al., 2024a) represents space and time jointly with anisotropic 4D Gaussians and derives time-conditioned 3D primitives for rendering. This native formulation provides a unified and extensible basis for spatio-temporal scene modeling, with subsequent work exploring higher-dimensional Gaussian representations (Gao et al., 2025), sparse-view dynamic reconstruction (Zhou et al., 2026), and compact modeling of long dynamic scenes (Xu et al., 2026). However, the kernel shapes in these methods remain restricted to the Gaussian form, limiting per-primitive flexibility in space and time.

Recent work has explored more expressive splatting kernels. Analytical approaches introduce alternative parametric kernel families (Hamdi et al., 2024; Liu et al., 2025a). Universal Beta Splat-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ting (UBS) (Liu et al., 2026) further extends Beta kernels to a unified high-dimensional formulation spanning spatial, temporal, and angular dimensions. However, these methods remain restricted to predefined functional forms. Learned approaches instead optimize kernel shapes directly from data (Huang et al., 2025; Zhou et al., 2025). Learning View-Dependent Splatting Kernels (LVDSK) (Ding et al., 2026) further learns view-dependent projected spatial kernels from per-primitive latent representations and shared neural networks. However, it focuses on spatial kernel modeling in static scenes, leaving temporal kernel learning for dynamic representations largely unexplored.

We propose to learn spatial and temporal kernels jointly within a native 4D Gaussian representation. Building on the parameterization and analytical conditioning of 4DGS (Yang et al., 2024a), we replace its prescribed spatial and temporal Gaussian kernels with primitive-specific learned kernels. At each timestamp, the time-conditioned 3D attributes and spatial latent are mapped to a view-dependent radial spatial kernel through a shared projection network and decoder. In parallel, a temporal decoder maps scale-normalized relative time and the temporal latent to a primitive-specific temporal kernel. We fix the temporal kernel value at each primitive's temporal center, allowing the decoder to focus on relative temporal variation rather than overall magnitude. The kernel networks and 4D primitive attributes, including the spatial and temporal latents, are jointly optimized. The learned spatial and temporal kernels are compatible with the standard splatting pipeline.

Experiments on standard dynamic novel-view synthesis benchmarks demonstrate the effectiveness of our method. Compared with state-of-the-art techniques, our approach achieves the best average reconstruction quality across all evaluated metrics. Our learned kernels also provide higher representation efficiency while preserving efficient splatting-based rendering. Visualizations further show that the learned kernels adapt to scene structure and dynamics. Our code and data are publicly available at `https://******`.

2 Related Work

We focus our discussion on two aspects most relevant to our method: dynamic Gaussian splatting and generalized splatting kernels. For broader developments in 3D Gaussian splatting and dynamic scene reconstruction, we refer readers to recent surveys (Bao et al., 2025; Chen & Wang, 2026; Han et al., 2026).

2.1 Dynamic Gaussian Splatting

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) has emerged as an efficient explicit representation for novel view synthesis, achieving high-quality rendering with real-time performance. Recent studies have explored extending 3D Gaussian Splatting to dynamic scenes. Some approaches (Yang et al., 2024b; Wu et al., 2024) predict time-dependent changes in Gaussian attributes using MLPs or HexPlane-based spatio-temporal representations. Other works explicitly model Gaussian motion trajectories. STGS (Li et al., 2024) parameterizes primitive motion and rotation with polynomial functions, while FreeTimeGS (Wang et al., 2025) models the local motion of each Gaussian with a linear motion function. SharpTimeGS (Liao et al., 2026) further introduces lifespan-aware motion modulation for dynamic Gaussians. In contrast to the above approaches, 4DGS (Yang et al., 2024a) and 4D-RotorGS (Duan et al., 2024) extend Gaussian primitives from 3D to 4D space-time by explicitly incorporating the temporal dimension. They directly model scene dynamics with anisotropic 4D Gaussians, from which time-specific 3D primitives are obtained through temporal conditioning for rendering. 7DGS (Gao et al., 2025) further generalizes this high-dimensional formulation by jointly modeling spatial, temporal, and angular dimensions within a unified Gaussian representation. 4C4D (Zhou et al., 2026) builds upon native 4D Gaussian primitives for sparse-view dynamic reconstruction and introduces neural opacity modulation to improve optimization under limited observations. These methods improve temporal modeling through deformation, explicit motion, native 4D Gaussian distributions, and predefined temporal visibility modulation. However, their spatial splatting kernels remain Gaussian, while the temporal dependence of visibility follows predefined forms.

2.2 Generalized Splatting Kernels

In 3D splatting, kernel generalization has been explored through Gaussian variants and alternative analytical families. Some methods modify Gaussian-based kernels through generalized decay profiles (Hamdi et al., 2024), truncated or asymmetric support (Liu et al., 2025b; Li et al., 2025), and skewed Gaussian profiles (Wu et al., 2026). Other methods adopt alternative analytical kernel families. 3D Student Splatting (Zhu et al., 2025) uses Student's t distributions, while Deformable Beta Splatting (Liu et al., 2025a) introduces bounded Beta kernels with learnable shape parameters. Universal Beta Splatting (Liu et al., 2026) further unifies spatial, temporal, and angular dimensions with high-dimensional Beta kernels. These methods broaden the analytical kernel family, but still predefine its functional form.

Beyond analytical kernels, several works learn primitive geometry or density directly from data. 3D Convex Splatting (Held et al., 2025) optimizes general convex primitive shapes, while Splat the Net (Zhou et al., 2025) represents each primitive with a neural density field. Related planar methods (Held et al., 2026; Huang et al., 2025; Jiménez-Ayguadé & Agudo, 2026) explore alternative or learnable primitive shapes. LVDSK (Ding et al., 2026) directly learns view-dependent projected kernels from latent primitive representations, but focuses on spatial kernels in static scenes. Our formulation learns both spatial and temporal kernels within a native 4D Gaussian representation.

3 Preliminaries

This section briefly reviews the formulations most relevant to our method, including 3D Gaussian splatting, view-dependent neural kernels, and native 4D Gaussian representations.

3.1 3D Gaussian Splatting

3DGS (Kerbl et al., 2023) represents a scene with anisotropic Gaussians, each defined by a center $\boldsymbol{\mu}_i$, covariance $\boldsymbol{\Sigma}_i = \mathbf{R}_i\mathbf{S}_i\mathbf{S}_i^\mathsf{T}\mathbf{R}_i^\mathsf{T}$, opacity α_i, and view-dependent color $\mathbf{c}_i$. Here, $\mathbf{R}_i$ and $\mathbf{S}_i$ are the rotation and diagonal scale matrices. Local perspective projection gives a screen-space center $\boldsymbol{\mu}_i^{\mathrm{2D}}$ and covariance $\boldsymbol{\Sigma}_i^{\mathrm{2D}} = \mathbf{JW}\boldsymbol{\Sigma}_i\mathbf{W}^\mathsf{T}\mathbf{J}^\mathsf{T}$, where $\mathbf{W}$ is the camera rotation and $\mathbf{J}$ is the 2×3 projection Jacobian at the Gaussian center. The Gaussian response in screen space can be defined as:

$$G_i(\mathbf{u}) = \exp\left(-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu}_i^{\mathrm{2D}})^\mathsf{T}(\boldsymbol{\Sigma}_i^{\mathrm{2D}})^{-1}(\mathbf{u}-\boldsymbol{\mu}_i^{\mathrm{2D}})\right) = \exp\left(-\frac{1}{2}r_i^2(\mathbf{u})\right). \tag{1}$$

Here $\mathbf{u}$ is the coordinate of a pixel and $r_i(\mathbf{u})$ is the Mahalanobis distance from $\mathbf{u}$ to the projected Gaussian center under covariance $\boldsymbol{\Sigma}_i^{\mathrm{2D}}$. Given Gaussians sorted in front-to-back order, the pixel color is composited as:

$$\mathbf{C}(\mathbf{u}) = \sum_i \alpha_i G_i(\mathbf{u})\mathbf{c}_i \prod_{j<i}\left(1-\alpha_j G_j(\mathbf{u})\right). \tag{2}$$

3.2 View-Dependent Neural Kernel

LVDSK (Ding et al., 2026) replaces G_i with a learned radial profile. Each primitive carries a bounding ellipsoid and a latent $\mathbf{z}_i^{\mathrm{s}}$. A projection network maps this latent and camera-space geometry to a 2D-kernel latent, which a decoder converts into the splatting response:

$$\begin{aligned}\mathbf{z}_i^{\mathrm{proj}} &= \Phi_{\mathrm{proj}}(\mathbf{z}_i^{\mathrm{s}}, \boldsymbol{\mu}_i^{\mathrm{cam}}, \mathbf{s}_i, \mathbf{R}_i^{\mathrm{cam}}),\\ d_i(r) &= \Phi_{\mathrm{dec}}\left(r^2, \mathbf{z}_i^{\mathrm{proj}}\right).\end{aligned} \tag{3}$$

Here, $\mathbf{s}_i$ and $\mathbf{R}_i^{\mathrm{cam}}$ specify the ellipsoid's scale and camera-relative orientation. The learned response replaces the Gaussian profile in compositing while retaining the projected geometry.

3.3 4D Gaussian Representation

4DGS (Yang et al., 2024a) extends Gaussian splatting to dynamic scenes by defining anisotropic Gaussian primitives jointly over space and time. For the i-th primitive, let the joint spatiotemporal

mean and covariance be $\boldsymbol{\mu}_i^{4\mathrm{D}} \in \mathbb{R}^4$ and $\boldsymbol{\Sigma}_i^{4\mathrm{D}} \in \mathbb{R}^{4\times4}$, respectively. We partition them as

$$\boldsymbol{\mu}_i^{4\mathrm{D}} = \begin{bmatrix} \boldsymbol{\mu}_i^{\mathrm{s}} \\ \tau_i \end{bmatrix}, \qquad \boldsymbol{\Sigma}_i^{4\mathrm{D}} = \begin{bmatrix} \boldsymbol{\Sigma}_i^{\mathrm{ss}} & \mathbf{q}_i \\ \mathbf{q}_i^{\mathsf{T}} & (\sigma_i^{\mathrm{t}})^2 \end{bmatrix}, \tag{4}$$

where τ_i and $(\sigma_i^{\mathrm{t}})^2$ denote the temporal center and variance, and $\mathbf{q}_i$ captures the space–time correlation.

Conditioning the joint Gaussian on a query time t yields a 3D Gaussian with mean $\boldsymbol{\mu}_{i,t}$ and covariance $\boldsymbol{\Sigma}_{i,t}$:

$$\boldsymbol{\mu}_{i,t} = \boldsymbol{\mu}_i^{\mathrm{s}} + \frac{\mathbf{q}_i}{(\sigma_i^{\mathrm{t}})^2}(t-\tau_i), \qquad \boldsymbol{\Sigma}_{i,t} = \boldsymbol{\Sigma}_i^{\mathrm{ss}} - \frac{\mathbf{q}_i\mathbf{q}_i^{\mathsf{T}}}{(\sigma_i^{\mathrm{t}})^2}. \tag{5}$$

For a fixed 4D primitive, the conditional covariance is independent of the query time t. The superscript t denotes the temporal dimension, and σ_i^{t} is the temporal standard deviation.

The corresponding temporal response is

$$T_i(t) = \exp\left(-\frac{(t-\tau_i)^2}{2(\sigma_i^{\mathrm{t}})^2}\right). \tag{6}$$

After projecting the conditioned 3D Gaussian onto the image plane, let $G_i(\mathbf{u} \mid t)$ denote its screen-space response, defined analogously to the static Gaussian response above. The rendered pixel color is then obtained by front-to-back compositing:

$$\mathbf{C}(\mathbf{u}, t) = \sum_i \alpha_i T_i(t) G_i(\mathbf{u} \mid t)\mathbf{c}_i \prod_{j<i} \left(1 - \alpha_j T_j(t) G_j(\mathbf{u} \mid t)\right). \tag{7}$$

4 Overview

Our method retains the native 4D primitive representation of 4DGS (Yang et al., 2024a) and directly learns kernels from data. Figure 2 summarizes the overall framework. At a query time t, each 4D primitive is first conditioned to obtain the 3D mean and covariance. Each primitive is equipped with separate spatial and temporal latents. The spatial latent, together with the conditioned 3D attributes, is mapped through neural projection and radial decoding to a view-dependent, radially symmetric 2D kernel (Sec. 5.1). In parallel, the temporal latent is decoded into a primitive-specific temporal kernel, whose response is anchored at the primitive's temporal center to separate kernel-shape variation from base opacity (Sec. 5.2). The spatial and temporal kernels jointly determine the primitive's spatio-temporal response, which is then passed to the splatting pipeline for color evaluation and alpha compositing (Sec. 5.3).

5 Method

5.1 Time-Conditioned Spatial Kernel

Following LVDSK (Ding et al., 2026), we obtain the screen-space spatial kernel through neural projection and radial decoding. In our dynamic setting, the projection is conditioned on the 3D geometry derived from the 4D primitive at the query time. As a result, time also implicitly influences the predicted spatial kernel through the conditioned geometry. For notational simplicity, we omit the dependence of the projected latent, radial coordinates, and spatial kernel on the query time.

For each camera view, we first transform the conditioned center and covariance into camera coordinates. The projection network maps the spatial latent and the camera-space geometry to a 2D kernel latent:

$$\mathbf{z}_i^{\mathrm{proj}} = \Phi_{\mathrm{proj}}\left(\mathbf{z}_i^{\mathrm{s}},\, \boldsymbol{\mu}_{i,t}^{\mathrm{cam}},\, \boldsymbol{\Sigma}_{i,t}^{\mathrm{cam}}\right), \tag{8}$$

Here, $\boldsymbol{\mu}_{i,t}^{\mathrm{cam}}$ and $\boldsymbol{\Sigma}_{i,t}^{\mathrm{cam}}$ are the camera-space forms of the time-conditioned 3D mean and covariance. Unlike Eq. 3, which uses scale and rotation as geometric inputs, our projection network directly uses the time-conditioned covariance. This mapping implicitly projects the primitive-level kernel representation into the current view, while the Gaussian center and covariance are still projected analytically.

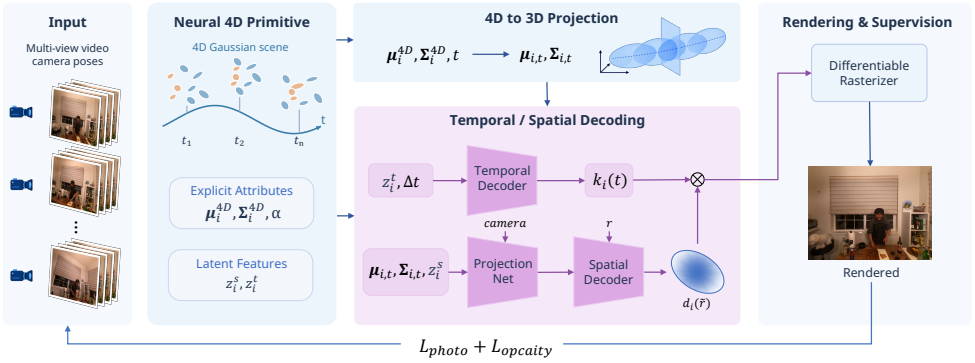


Figure 2: Overview of our framework for learning spatial and temporal splatting kernels. At each query time, we condition the 4D primitives to obtain their 3D geometry. A projection network and spatial decoder map this geometry and the spatial latent to a view-dependent screen-space kernel $d_i(\tilde{r})$ (Eq. 9). In parallel, a temporal decoder uses the temporal latent and normalized relative time to predict a center-anchored temporal kernel $k_i(t)$ (Eq. 12). The two responses jointly modulate each primitive's opacity for differentiable splatting and alpha blending. The primitive attributes, latent features, and shared networks are jointly optimized under image reconstruction supervision.

A shared spatial decoder Φ_{dec} decodes a radially symmetric screen-space kernel from the projected latent $\mathbf{z}_i^{\text{proj}}$. Let $r_i(\mathbf{u})$ denote the Mahalanobis distance defined in Eq. 1, computed using the time-conditioned projected center and covariance. We restrict the spatial kernel to the three-standard-deviation region $r_i(\mathbf{u}) \leq 3$. Within this region, we rescale the Mahalanobis distance as $\tilde{r}_i(\mathbf{u}) = r_i(\mathbf{u})/3$, so that $\tilde{r}_i(\mathbf{u}) \in [0, 1]$. For a rescaled radial coordinate $\tilde{r} \in [0, 1]$, the kernel response is

$$d_i(\tilde{r}) = \Phi_{\text{dec}}\left(\tilde{r}^2, \mathbf{z}_i^{\text{proj}}\right), \qquad d_i(\tilde{r}) \in (0, 1). \tag{9}$$

The spatial response is set to zero for $r_i(\mathbf{u}) > 3$.

5.2 Center-Anchored Temporal Kernel

Instead of the analytical temporal Gaussian in Eq. 6, we learn a primitive-specific temporal response from scale-normalized relative time. We define the temporal coordinate as

$$\rho_i(t) = \frac{t - \tau_i}{3\sigma_i^{\text{t}}}, \qquad u_i(t) = \frac{\rho_i^2(t)}{1 + \rho_i^2(t)}. \tag{10}$$

The normalization expresses temporal distance relative to the extent of each 4D primitive. Squaring the coordinate gives a symmetric kernel around τ_i, while the bounded mapping compresses the unbounded distance to $[0, 1)$ without introducing a hard temporal cutoff.

Given the temporal latent $\mathbf{z}_i^{\text{t}}$, a shared decoder Φ_{t} predicts a relative neural energy and the corresponding temporal kernel:

$$E_i(t) = \Phi_{\text{t}}\left(0, \mathbf{z}_i^{\text{t}}\right) - \Phi_{\text{t}}\left(u_i(t), \mathbf{z}_i^{\text{t}}\right), \tag{11}$$

$$k_i(t) = \text{sigmoid}(\text{logit}(\kappa) - E_i(t)), \qquad \kappa = 0.9. \tag{12}$$

At the temporal center, $u_i(\tau_i) = 0$, which gives $E_i(\tau_i) = 0$ and $k_i(\tau_i) = \kappa$ by construction. This center anchor removes the free variation of the kernel response at the primitive center and reduces its amplitude ambiguity with the base opacity. The decoder therefore focuses on learning the relative temporal profile around each primitive center. Apart from this center anchor, we impose no additional monotonicity or compact-support constraints.

5.3 Efficient Splatting with Learned Kernels

Evaluating the spatial kernel response at every covered pixel would require a large number of network evaluations. Following LVDSK (Ding et al., 2026), we instead pre-sample each primitive's

one-dimensional radial profile at S uniformly spaced radii in $[0, 1]$ and linearly interpolate the response at $\tilde{r}_i(\mathbf{u})$ during rasterization. This decouples decoder evaluations from the number of covered pixels. Unless otherwise stated, we use $S = 2$ radial samples.

In contrast, the temporal kernel $k_i(t)$ depends only on the primitive and the query time. It is therefore evaluated once per primitive at each timestamp and shared across all covered pixels, requiring no additional interpolation.

We denote the resulting spatio-temporal kernel by

$$K_i(\tilde{r}, t) = k_i(t)\, d_i(\tilde{r})\,. \tag{13}$$

Replacing the analytical spatio-temporal Gaussian kernel in Eq. 7 with $K_i(\tilde{r}_i(\mathbf{u}), t)$ gives our rendering formulation:

$$\mathbf{C}(\mathbf{u}, t) = \sum_{i=1}^{N} \alpha_i K_i(\tilde{r}_i(\mathbf{u}), t)\mathbf{c}_i \prod_{j=1}^{i-1} \left(1 - \alpha_j K_j(\tilde{r}_j(\mathbf{u}), t)\right). \tag{14}$$

5.4 Implementation Details

Network architecture. The projection network and kernel decoders are shared across primitives, while the latent codes are optimized independently for each primitive. We use a 10D spatial latent $\mathbf{z}_i^{\mathrm{s}}$, a 5D temporal latent $\mathbf{z}_i^{\mathrm{t}}$, and a 5D projected spatial latent $\mathbf{z}_i^{\mathrm{proj}}$. The projection network contains 4 hidden layers of width 64. Both kernel decoders contain 4 hidden layers of width 8 with LeakyReLU activations. The spatial decoder produces a sigmoid kernel response, while the temporal decoder produces the scalar used in Eq. 12.

Kernel initialization. As in LVDSK (Ding et al., 2026), we pre-train the neural kernel networks with Gaussian-shaped profiles before scene reconstruction. This initialization provides a well-behaved kernel response before the neural kernels are jointly optimized with scene geometry and appearance. The temporal pre-training follows the same center anchoring with $\kappa = 0.9$ used in Eq. 12. The pre-training is performed on randomly sampled geometric configurations and does not optimize any scene-specific primitives.

Scene optimization. For scene reconstruction, we jointly optimize the 4D primitive parameters, spatial and temporal latents, and neural kernel networks using Adam. The appearance is represented by the 4D spherical harmonics of 4DGS (Yang et al., 2024a).

We retain the densification strategy of 4DGS (Yang et al., 2024a), but disable its periodic opacity reset. In our formulation, the effective opacity is jointly determined by the base opacity and the learned spatial and temporal kernel responses. A hard opacity reset can therefore strongly suppress primitives with small kernel responses and make their contributions difficult to recover. We instead regularize the base opacity continuously and optimize

$$\mathcal{L} = (1 - \lambda_{\mathrm{ssim}})\mathcal{L}_1 + \lambda_{\mathrm{ssim}}\mathcal{L}_{\mathrm{SSIM}} + \frac{\lambda_{\mathrm{opacity}}}{N} \sum_{i=1}^{N} \alpha_i. \tag{15}$$

Here, N is the number of primitives. The regularization provides a continuous constraint on base opacity without the abrupt global perturbation introduced by periodic resetting.

6 Experiments

Our evaluation covers 6 real-world multi-view scenes from N3V (Li et al., 2022), downsampled by a factor of 4, and 8 synthetic monocular scenes from D-NeRF (Pumarola et al., 2021) at the original resolution. Reconstruction quality is measured using PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018). All experiments are conducted on a workstation equipped with dual AMD EPYC 7763 CPUs, 768 GB RAM, and an NVIDIA RTX 4090 GPU. For a fair comparison, we rerun 4DGS (Yang et al., 2024a) and UBS (Liu et al., 2026) using their official public implementations under the same hardware and evaluation settings. All methods are trained for 30K iterations and evaluated at the final iteration.

Table 1: Quantitative comparison with 4DGS (Yang et al., 2024a) and UBS (Liu et al., 2026) on N3V (Li et al., 2022) and D-NeRF (Pumarola et al., 2021) datasets. Best results are highlighted in bold.

Dataset	Scene	4DGS			UBS			**Ours**		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
N3V	Coffee Martini	28.46	0.921	0.111	28.27	0.919	0.111	**29.95**	**0.935**	**0.091**
	Cook Spinach	33.07	0.951	0.102	31.74	0.946	0.120	**33.34**	**0.959**	**0.091**
	Cut Roasted Beef	33.61	0.956	0.100	33.10	0.950	0.120	**34.02**	**0.962**	**0.089**
	Flame Salmon	29.15	0.925	0.106	26.92	0.911	0.121	**30.17**	**0.933**	**0.096**
	Flame Steak	33.65	0.960	0.098	31.56	0.952	0.115	**33.68**	**0.966**	**0.089**
	Sear Steak	33.59	0.961	0.099	32.12	0.954	0.114	**34.02**	**0.967**	**0.087**
	Average	31.92	0.946	0.103	30.62	0.939	0.117	**32.53**	**0.954**	**0.091**
D-NeRF	bouncingballs	32.79	0.982	0.030	32.85	0.984	0.026	**34.64**	**0.987**	**0.024**
	hellwarrior	33.39	0.945	0.074	32.75	0.930	0.095	**33.75**	**0.953**	**0.072**
	hook	31.33	0.956	0.045	28.59	0.938	0.062	**31.90**	**0.960**	**0.044**
	jumpingjacks	30.69	0.966	0.038	30.05	0.964	0.040	**31.71**	**0.971**	**0.034**
	lego	24.81	0.906	0.087	26.53	**0.927**	**0.071**	**26.54**	0.912	0.076
	mutant	36.39	0.981	0.020	**37.66**	**0.988**	**0.013**	36.54	0.983	0.023
	standup	**38.24**	**0.985**	**0.017**	36.59	0.984	0.018	37.98	0.985	0.020
	trex	28.79	0.972	0.030	29.10	0.971	0.030	**30.74**	**0.978**	**0.026**
	Average	32.05	0.962	0.043	31.77	0.961	0.044	**32.98**	**0.966**	**0.040**

Table 2: Comparison of representation efficiency and rendering performance on the N3V dataset. Results are reported at three similar memory footprints, including all serialized primitive attributes and neural parameters. Num. denotes the number of primitives.

Method	∼10 MB			∼25 MB			∼50 MB		
	PSNR↑	Num.↓	FPS↑	PSNR↑	Num.↓	FPS↑	PSNR↑	Num.↓	FPS↑
4DGS (Yang et al., 2024a)	30.23	133.6K	1556.09	30.98	312.7K	1377.74	31.21	620.7K	1168.21
UBS (Liu et al., 2026)	29.85	59.6K	279.76	30.19	148.9K	132.43	30.50	297.9K	75.31
Ours	**31.39**	80.6K	295.19	**32.04**	191.4K	202.78	**32.39**	376.2K	134.37

6.1 Comparison

Reconstruction quality. Table 1 summarizes reconstruction quality. Our method achieves the best average performance across all three metrics on both benchmarks. On N3V, it reaches 32.53 dB PSNR, outperforming 4DGS and UBS by 0.61 dB and 1.91 dB, respectively. It achieves the best PSNR, SSIM, and LPIPS on all six scenes. On D-NeRF, our method improves the average PSNR from 32.05 dB for 4DGS and 31.77 dB for UBS to 32.98 dB. Figure 3 further demonstrates the improved visual quality of our method over the baselines.

Representation efficiency. Table 2 evaluates representation efficiency by comparing our method with 4DGS and UBS at similar representation sizes. Following LVDSK (Ding et al., 2026), we disable all view-dependent color components to focus on the impact of kernel geometry. Across all three memory budgets, our method consistently achieves the highest reconstruction quality while maintaining competitive rendering speed. These results demonstrate improved representation efficiency from the learned kernels.

Kernel visualization. Figure 4 visualizes representative learned temporal kernels and compares them with the Gaussian profile. The learned kernels exhibit diverse primitive-specific responses beyond the fixed Gaussian shape.

We further analyze the distributions of spatial kernel center responses across scenes and viewpoints in Figure 5. The distributions vary with both scene and view, demonstrating the adaptive geometry of our learned spatial kernels.

6.2 Ablation Study

We conduct all ablation experiments on all six scenes of N3V to evaluate the main design choices of our spatial and temporal kernels. Results are averaged across these scenes. Unless otherwise

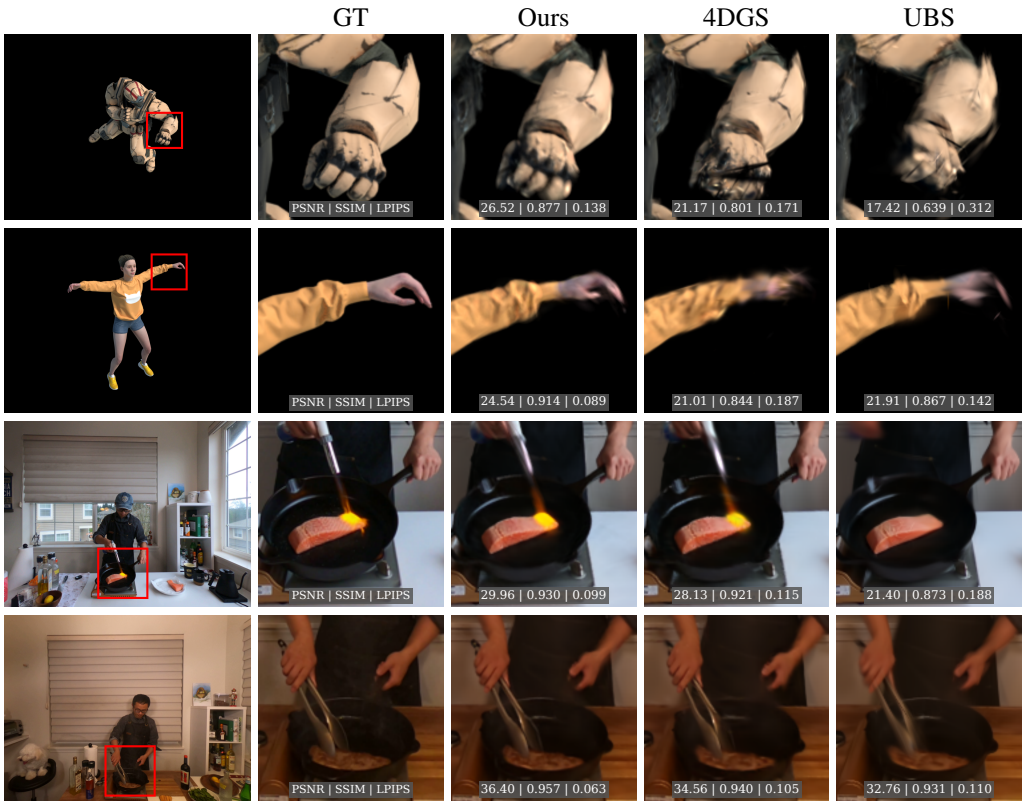


Figure 3: Qualitative comparison with 4DGS (Yang et al., 2024a) and UBS (Liu et al., 2026). PSNR, SSIM, and LPIPS are computed on the highlighted regions and reported at the bottom of each corresponding crop.

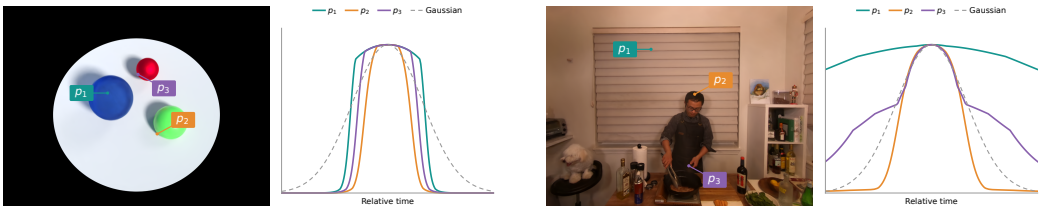


Figure 4: Representative learned temporal kernels. In each panel, colored markers p_1–p_3 indicate three primitives. Solid curves on the right show their corresponding learned kernel responses over normalized relative time, while the gray dashed curve shows a standard Gaussian rescaled to the fixed center response defined for our temporal kernels (Eq. 12) for visual comparison.

specified, only the component under study is changed while the remaining representation and optimization settings are fixed.

Spatial kernel. The spatial-kernel ablations in Table 3 evaluate three design choices: the kernel representation, predictor inputs, and pre-training. First, we replace the learned spatial kernel with the analytical Gaussian kernel while retaining the learned temporal kernel. Second, we retain the learned spatial kernel and shared latent code, but replace the time-conditioned 3D attributes with the native 4D Gaussian attributes $\boldsymbol{\mu}_i^{4D}, \boldsymbol{\Sigma}_i^{4D}$ as spatial projection inputs (4D Attr.). Finally, we disable spatial-kernel pre-training while retaining temporal-kernel pre-training. Learning spatial kernels from time-conditioned geometry gives the best reconstruction quality. The substantial degradation without pre-training further highlights the importance of kernel initialization.



Figure 5: Spatial-kernel distributions across scenes and viewpoints. Columns show different viewpoints at a fixed time. Insets show histograms of the learned center response $d_i(0)$ (Eq. 9), with red dashed lines indicating the mean. The horizontal axis spans $[0, 1]$.

Temporal kernel. The temporal-kernel ablations in Table 3 evaluate the kernel representation, decoder factorization, output parameterization, temporal support, and pre-training. To assess the kernel representation, we replace the learned temporal kernel with the analytical Gaussian kernel while retaining the learned spatial kernel. To examine decoder factorization, we replace the separate spatial and temporal decoders with a parameter-matched joint decoder for the combined spatio-temporal kernel (Joint Dec.). For output parameterization, we replace the center-anchored relative-energy formulation with direct temporal-kernel prediction. We further compare our bounded coordinate mapping with finite temporal support (Finite Supp.) that uses $\rho_i^2(t)$ and sets the temporal response to zero when $|\rho_i(t)| > 1$. Finally, we disable temporal-kernel pre-training while retaining spatial-kernel pre-training. Overall, our default design achieves the best reconstruction quality among all temporal-kernel variants. The substantial degradation without temporal pre-training further highlights the importance of pre-training for optimization.

Table 3: Ablation of spatial and temporal kernel designs on the six N3V scenes. *4D Attr.*: native 4D attributes as spatial projection inputs; *Joint Dec.*: joint spatial-temporal decoder; *Direct*: direct temporal-kernel prediction; *Finite Supp.*: finite temporal support; *w/o Pre.*: without pre-training. *Ours* uses all default settings. Best results, determined before rounding, are shown in bold.

Metric	Spatial kernel			Temporal kernel					**Ours**
	Gaussian	4D Attr.	w/o Pre.	Gaussian	Joint Dec.	Direct	Finite Supp.	w/o Pre.	
PSNR↑	32.24	32.30	17.60	32.35	31.89	32.36	32.32	30.91	**32.53**
SSIM↑	0.950	0.952	0.627	0.952	0.952	0.953	0.950	0.944	**0.954**
LPIPS↓	0.094	0.092	0.466	0.091	0.093	0.092	0.094	0.103	**0.091**

7 Conclusion

We presented a framework for learning spatial and temporal kernels in a 4D splatting-based pipeline for dynamic novel-view synthesis. The learned kernels adapt their profiles to scene structure and dynamics, providing flexible spatial and temporal responses beyond predefined Gaussian forms. Experiments on standard dynamic novel-view synthesis benchmarks demonstrate improved reconstruction quality and representation efficiency, while ablations validate the key spatial and temporal kernel designs. Overall, our results highlight kernel learning as a promising modeling dimension for more expressive dynamic splatting representations.

AI Use Statement

We used generative AI tools to provide feedback on research methodology and experimental design, assist with method implementation and debugging, and discuss experimental observations. We also

used these tools to assist with literature search, scientific figure design, LaTeX formatting, and drafting and editing the manuscript for clarity and readability. Generative AI was not used to generate experimental data, results or references.

All AI-assisted suggestions, code, and scientific claims were reviewed and verified by the authors. The authors made the final decisions regarding the method, experiments, analysis, and presentation, and take full responsibility for the content of this work.

Ethics Statement

This work studies dynamic scene reconstruction and novel-view synthesis using publicly available benchmark datasets. We do not collect new personal data or conduct studies with human participants or animals. Unlike the synthetic scenes in D-NeRF, the Neural 3D Video dataset contains recordings of people performing everyday activities. We use these recordings solely for reconstruction and rendering evaluation, without attempting to identify individuals or infer sensitive attributes. As with other image synthesis techniques, the resulting models could potentially be misused to create misleading visual content. We encourage responsible use that respects privacy and applicable dataset usage conditions.

Reproducibility Statement

We have made every effort to ensure that our results are reproducible. Our method is described in Sec. 5, and implementation details and key hyperparameters are given in Sec. 5.4. The experimental setup is described in Sec. 6. Our code is included in the submitted supplementary material for review. All evaluations use the publicly available D-NeRF and Neural 3D Video datasets.

References

Yanqi Bao, Tianyu Ding, Jing Huo, Yaoli Liu, Yuxin Li, Wenbin Li, Yang Gao, and Jiebo Luo. 3d gaussian splatting: Survey, technologies, challenges, and opportunities. *IEEE Transactions on Circuits and Systems for Video Technology*, 35(7):6832–6852, 2025.

Guikun Chen and Wenguan Wang. A survey on 3d gaussian splatting. *ACM Computing Surveys*, 58 (12):1–39, 2026.

Huakeng Ding, Zhanpeng Liu, Fan Pei, Kun Zhou, and Hongzhi Wu. Learning view-dependent splatting kernels. *arXiv preprint arXiv:2605.25426*, 2026.

Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d-rotor gaussian splatting: towards efficient novel view synthesis for dynamic scenes. In *ACM SIGGRAPH 2024 conference papers*, pp. 1–11, 2024.

Zhongpai Gao, Benjamin Planche, Meng Zheng, Anwesa Choudhuri, Terrence Chen, and Ziyan Wu. 7dgs: Unified spatial-temporal-angular gaussian splatting. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 26316–26325. IEEE, 2025.

Abdullah Hamdi, Luke Melas-Kyriazi, Jinjie Mai, Guocheng Qian, Ruoshi Liu, Carl Vondrick, Bernard Ghanem, and Andrea Vedaldi. Ges: Generalized exponential splatting for efficient radiance field rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19812–19822. IEEE, 2024.

Dong Han, Cheng-Ye Su, Fan-Yi Zeng, Fang-Lue Zhang, and Miao Wang. Dynamic scene representation in the era of neural rendering: from nerfs to 3dgss. *Frontiers of Computer Science*, 20 (11):2011708, 2026.

Jan Held, Renaud Vandeghen, Abdullah Hamdi, Adrien Deliege, Anthony Cioppa, Silvio Giancola, Andrea Vedaldi, Bernard Ghanem, and Marc Van Droogenbroeck. 3d convex splatting: Radiance field rendering with 3d smooth convexes. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24360–21369. IEEE, 2025.

Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Silvio Giancola, Daniel Rebain, Anthony Cioppa, Bernard Ghanem, Andrea Vedaldi, Andrea Tagliasacchi, et al. Triangle splatting for real-time radiance field rendering. In *2026 International Conference on 3D Vision (3DV)*, pp. 1248–1257. IEEE, 2026.

Yi-Hua Huang, Ming-Xian Lin, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Deformable radial kernel splatting. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21513–21523. IEEE, 2025.

Oriol Jiménez-Ayguadé and Antonio Agudo. Deformable triangle splatting: Flexible primitives for real-time radiance field rendering. *arXiv preprint arXiv:2607.22446*, 2026.

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

Haolin Li, Jinyang Liu, Mario Sznaier, and Octavia Camps. 3d-hgs: 3d half-gaussian splatting. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10996–11005. IEEE, 2025.

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 *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5511–5521. IEEE, 2022.

Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8508–8520. IEEE, 2024.

Zhanfeng Liao, Jiajun Zhang, Hanzhang Tu, Zhixi Wang, Yunqi Gao, Hongwen Zhang, and Yebin Liu. Sharptimegs: Sharp and stable dynamic gaussian splatting via lifespan modulation. *arXiv preprint arXiv:2602.02989*, 2026.

Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. Deformable beta splatting. In *Proceedings of the special interest group on computer graphics and interactive techniques conference conference papers*, pp. 1–11, 2025a.

Rong Liu, Zhongpai Gao, Benjamin Planche, Meida Chen, Van Nguyen, Meng Zheng, Anwesa Choudhuri, Terrence Chen, Yue Wang, Andrew Feng, et al. Universal beta splatting. In *International Conference on Learning Representations*, volume 2026, pp. 101275–101294, 2026.

Xiaofeng Liu, Guanchen Meng, Chongyang Feng, Risheng Liu, Zhongxuan Luo, and Xin Fan. Tntgs: Truncated and tailored gaussian splatting. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 573–581, 2025b.

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pp. 800–809. IEEE, 2024.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10313–10322. IEEE, 2021.

Yifan Wang, Peishan Yang, Zhen Xu, Jiaming Sun, Zhanhua Zhang, Yong Chen, Hujun Bao, Sida Peng, and Xiaowei Zhou. Freetimegs: Free gaussian primitives at anytime anywhere for dynamic scene reconstruction. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21750–21760. IEEE, 2025.

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.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20310–20320. IEEE, 2024.

Xiangru Wu, Ke Fan, and Yanwei Fu. 3d skew-normal splatting. *arXiv preprint arXiv:2605.15010*, 2026.

Hanjie Xu, Yuanxing Duan, Qiyu Dai, Ge Li, Baoquan Chen, and He Wang. Layered 4d-rotor gaussian splatting: A compressed representation for long dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18958–18967, 2026.

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *International Conference on Learning Representations*, volume 2024, pp. 9142–9159, 2024a.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20331–20341. IEEE, 2024b.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Junsheng Zhou, Zhifan Yang, Liang Han, Wenyuan Zhang, Kanle Shi, Shenkun Xu, and Yu-Shen Liu. 4c4d: 4 camera 4d gaussian splatting. *arXiv preprint arXiv:2604.04063*, 2026.

Xilong Zhou, Bao-Huy Nguyen, Loïc Magne, Vladislav Golyanik, Thomas Leimkühler, and Christian Theobalt. Splat the net: Radiance fields with splattable neural primitives. *arXiv preprint arXiv:2510.08491*, 2025.

Jialin Zhu, Jiangbei Yue, Feixiang He, and He Wang. 3d student splatting and scooping. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21045–21054. IEEE, 2025.