POINTSPLATV2: QUERY-BASED FEED-FORWARD 4D GAUSSIAN RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

On-the-fly dynamic reconstruction is crucial for real-time spatial computing and immersive applications, requiring 4D representations that render stably from free viewpoints. Existing feed-forward methods rapidly reconstruct scenes by predicting Gaussian representations aligned with individual input observations, such as views or timesteps. This observation-wise discrete prediction prevents them from directly consolidating multi-view and temporal observations into a shared continuous, compact 4D representation, resulting in temporal flickering and redundant Gaussian primitives. Conversely, optimization-based methods achieve spatiotemporal consistency and compactness, but their costly per-scene optimization limits low-latency applications. To bridge this gap, we introduce PointSplatV2, a query-based feed-forward framework for reconstructing 4D Gaussians. Given multi-view temporal observations, PointSplatV2 first uses a multi-view image encoder to construct a latent field from the input frames. It then builds key-spatiotemporal queries that sparsely yet completely cover the spatial projection of the 4D scene at a key timestep. Each query then aggregates features from this latent field via cross-attention and directly decodes a 4D Gaussian representation. This simple yet effective formulation gathers temporal observations onto shared queries to avoid dense per-frame discrete prediction, and lets the continuous representation render stably across time. To extend this formulation to longer sequences, we introduce the Boundary-Aware Sequential Training (BAST) strategy, which enforces temporal coherence at sequence interfaces. Experiments show that PointSplatV2 outperforms the evaluated feed-forward 4D baseline in quality and compactness at low latency.

1 INTRODUCTION

Efficient dynamic scene reconstruction is a critical step toward real-time spatial computing, enabling systems to reconstruct and render dynamic environments with low latency. Central to this goal is the creation of compact and temporally consistent scene representations, which support applications ranging from immersive telepresence and volumetric video to free-viewpoint content creation.

To meet these rigorous requirements, particularly for capturing complex dynamic variations, offline 4D reconstruction methods Yang et al. (2024); Xu et al. (2024b); Wang et al. (2025), such as FreeTimeGS (Wang et al., 2025), construct continuous and temporally coherent 4D scene representations by per-scene optimization. However, this paradigm inherently requires hours of computation, making it unsuitable for low-latency, on-the-fly reconstruction.

Driven by strict low-latency demands, recent feed-forward 3D Gaussian reconstruction methods Charatan et al. (2024); Chen et al. (2024); Xu et al. (2025a); Zheng et al. (2024b); Zhang et al. (2024); Xu et al. (2024a); Tang et al. (2024) bypass costly per-scene optimization by predicting explicit 3D representations directly from images. Among them, PointSplat Guo et al. (2026) performs feed-forward prediction in 3D space, but still processes dynamic inputs frame by frame. Feed-forward 4D methods Ren et al. (2024); Xu et al. (2025b); Lin et al. (2025) further extend this paradigm to dynamic scenes. However, most methods predict Gaussian representations still aligned with individual input observations, such as a particular timestep or viewpoint. This observation-wise discrete prediction prevents direct consolidation of repeated multi-view and temporal observations into a shared continuous 4D representation, causing temporal flickering and redundant primitives.

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.

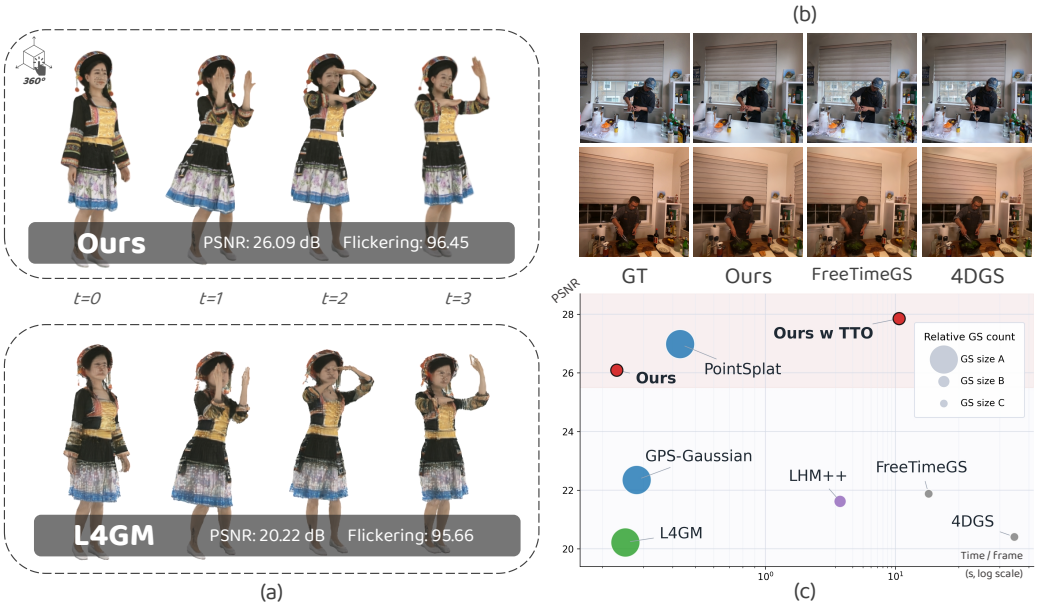

Figure 1: **Comparisons with Gaussian reconstruction baselines.** (a) Feed-forward 4D comparison with L4GM. (b) Qualitative comparison with optimization-based methods. (c) Runtime–quality–compactness trade-off across baseline families.

In this work, we present **PointSplatV2**, a query-based feed-forward framework for sparse-view 4D Gaussian reconstruction (Fig. 1). Our key idea is to query the 4D representation through its spatial projection at a key timestep. Given a synchronized multi-view video chunk, we first encode all input frames and views into a 3D latent field. We then construct key-spatiotemporal queries that are sparse yet complete, using a 3D visual hull at the middle timestep and DA3 depth at the first timestep. These queries describe the complete projection of the 4D scene, rather than a union of projections from each input frame. Each query then aggregates multi-frame, multi-view features from the latent field using a simple cross-attention mechanism, gathering temporal observations onto the shared support before prediction. Finally, a 4D Gaussian decoder directly predicts 4D Gaussian primitives parameterized following FreeTimeGS (Wang et al., 2025), representing a continuous 4D scene. This query-based formulation offers two key advantages. First, sparse queries replace redundant observation-wise Gaussian predictions and concentrate computation on geometrically supported locations for efficient reconstruction. Second, the shared 4D prediction is rendered at each target time, allowing for temporal consistency.

However, applying the short-chunk predictor independently across a long sequence can still yield incompatible renderings at chunk boundaries. We therefore introduce *Boundary-Aware Sequential Training* (BAST), which jointly trains consecutive chunk predictions and enforces rendering consistency on their overlapping frames. BAST exposes the model to the boundary conditions encountered during long-sequence prediction while preserving feed-forward inference.

We evaluate PointSplatV2 on DNA-Rendering and Neural3DV, covering both human-centric dynamic captures and general dynamic multi-view scenes. Compared with the strongest feed-forward 4D baseline, PointSplatV2 achieves superior quality and compactness while preserving low latency; it also attains comparable rendering quality with feed-forward 3D methods while markedly improving compactness and speed. These results demonstrate a favorable trade-off among reconstruction quality, efficiency, and representation compactness.

In summary, our main contributions are as follows:

- We introduce PointSplatV2, a query-based feed-forward framework that reconstructs a continuous and compact 4D Gaussian representation from sparse multi-view videos, replacing redundant discrete predictions per source view or timestep with key-spatiotemporal queries.

- We propose Boundary-Aware Sequential Training (BAST), a training strategy that jointly supervises consecutive chunks and uses overlap-rendering consistency to stabilize long-sequence prediction.
- Extensive experiments on human-centric and general dynamic scenes show that PointSplatV2 outperforms the evaluated feed-forward 4D baseline in quality and compactness, achieves rendering quality comparable to that of strong feed-forward 3D reconstruction methods, and substantially improves their compactness and inference speed.

2 Related Work

2.1 Optimization-based Dynamic Reconstruction

Optimization-based dynamic reconstruction fits time-varying representations from dense input views. NeRF-based methods Park et al. (2021a;b); Li et al. (2022); Peng et al. (2023) optimize neural or radiance representations per sequence, while plane-based methods Fridovich-Keil et al. (2023); Cao & Johnson (2023) reduce training and inference cost. Recent explicit 4D Gaussian methods Yang et al. (2024); Li et al. (2024); Wu et al. (2024); Xu et al. (2024b); Wang et al. (2025); Wu et al. (2026); Cho et al. (2026) improve rendering and temporal modeling, but their sequence-specific fitting still requires hours and is unsuitable for low-latency applications. We adopt FreeTimeGS's 4D Gaussian parameterization as an output representation, while replacing sequence-specific fitting with a feed-forward framework.

2.2 Feed-forward 3D Reconstruction

Feed-forward 3D reconstruction predicts an explicit scene representation directly from images, avoiding per-scene optimization. Early generalizable view synthesis methods Wang et al. (2021); Lin et al. (2022; 2023) use radiance-field or image-based rendering networks to aggregate multi-view observations, but their implicit representations remain expensive to render or adapt for compact transmission. The emergence of 3D Gaussian Splatting Kerbl et al. (2023) has shifted this line of work toward explicit Gaussian prediction. Recent methods Charatan et al. (2024); Zheng et al. (2024b); Chen et al. (2024); Xu et al. (2025a); Zhang et al. (2024); Xu et al. (2024a); Tang et al. (2024); Chen et al. (2025) regress 3D Gaussian maps from sparse views in a single forward pass. PointSplat Guo et al. (2026) performs reconstruction directly in 3D space, but processes input videos frame by frame, leaving temporal observations separate. For human-centric applications, LHM methods Qiu et al. (2025a;b) further use parametric body priors. These methods either target static scenes or, in dynamic settings, predict Gaussian sets aligned with individual observations at discrete timesteps. In contrast, PointSplatV2 aggregates temporal observations with sparse queries and decodes a continuous 4D representation.

2.3 Feed-forward 4D Reconstruction

Feed-forward 4D reconstruction predicts time-varying representations from video without per-scene optimization. 4DGT Xu et al. (2025b) and DGS-LRM Lin et al. (2025) demonstrate generalizable dynamic reconstruction from monocular videos, but extending such architectures to multi-view inputs faces substantial memory costs. Most closely related, L4GM Ren et al. (2024) predicts a separate 3D Gaussian map for each frame of synchronized sparse-view video. This observation-aligned discrete output does not directly consolidate temporal observations onto shared 4D support. In contrast, PointSplatV2 uses sparse query tokens to decode a continuous, compact 4D representation.

3 Methodology

We consider feed-forward 4D reconstruction from calibrated, synchronized sparse-view videos. Let $\mathcal{C}_k = \{I^v_{k,t} \mid t = 1, \ldots, T,\ v = 1, \ldots, V\}$ be a temporal chunk with V input views, and let $\bar{t}_k$ denote its middle timestep. The overview of PointSplatV2 is illustrated in Figure 2. We introduce the output parameterization in Sec. 3.1, then describe the latent field Sec. 3.2, key-spatiotemporal query construction Sec. 3.3, decoder Sec. 3.4, and Boundary-Aware Sequential Training in Sec. 3.5.

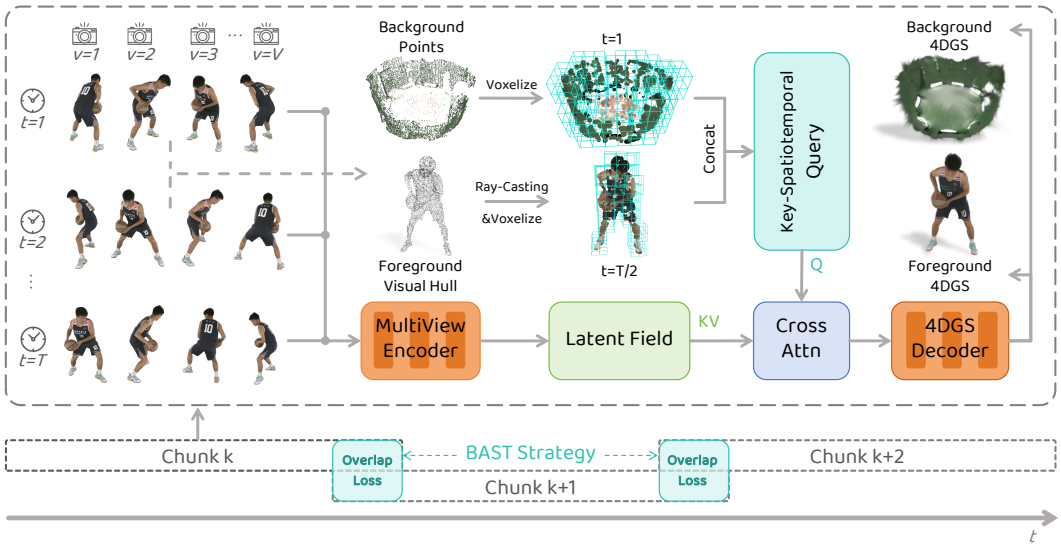

Figure 2: **Overview of PointSplatV2.** Given a synchronized multi-view video chunk, we first encode all frames into a latent field. We then construct key-spatiotemporal queries that are sparse yet complete, using ray-casted and voxelized foreground visual hull and voxelized background points. Each query then aggregates features from the latent field using a cross-attention mechanism. Finally, a 4D Gaussian decoder directly predicts 4DGS primitives using separate heads. BAST supervises overlapping renderings from consecutive chunks to stabilize long-sequence prediction.

3.1 Preliminary

FreeTimeGS represents dynamic scenes as a collection of 4D Gaussians. Specifically, each primitive is defined by a 4D mean $\mu \in \mathbb{R}^4$, a 3D scale $s \in \mathbb{R}^3$, a rotation quaternion $r \in \mathbb{R}^4$, an opacity $\alpha \in \mathbb{R}$, and SH coefficients $h \in \mathbb{R}^m$, where m denotes the number of SH bases. Each Gaussian is assigned a velocity vector $\mathbf{v} \in \mathbb{R}^3$ and a temporal variance $\sigma_t \in \mathbb{R}$ to model temporal dynamics and visibility. Given a target timestamp t, each primitive is marginalized to a 3D Gaussian before rasterization, and the 3D position of the Gaussian at time t is computed as:

$$\mu_x(t) = \mu_x + \mathbf{v} \cdot (t - \mu_t), \tag{1}$$

where μ_x and μ_t represent the 3D position and time mean of the Gaussian, respectively, and its effective opacity is modulated by a Gaussian temporal kernel:

$$\alpha(t) = \alpha \exp\left(-\frac{(t-\mu_t)^2}{2\sigma_t^2}\right). \tag{2}$$

3.2 Latent Field Construction

We first extract image features given multi-view frames to construct the latent field. For each timestep t, we use the Depth Anything 3 Lin et al. (2026) (DA3) feature encoder to obtain dense multi-view features and geometric predictions. Specifically, the backbone provides a list of multi-scale features extracted from N different layers, denoted as $\{\mathbf{h}_{k,t}^{v,(n)}\}_{n=1}^{N}$. These multi-scale features are upsampled to the input resolution, concatenated across layers, and projected into the transformer dimension through a gated linear unit (GLU) Dauphin et al. (2017), producing the mapped feature $\phi_{k,t,p}^{v}$ for patch p:

$$\phi_{k,t,p}^{v} = \mathrm{GLU}\left(\mathrm{Linear}\left(\mathrm{Concat}_{n=1}^{N}\left(\mathrm{Upsample}\left(\mathbf{h}_{k,t,p}^{v,(n)}\right)\right)\right)\right). \tag{3}$$

In parallel, we build a posed image token by concatenating the RGB patch and its Plücker ray:

$$\psi_{k,t,p}^{v} = \mathrm{Concat}(\mathrm{RGB}_{k,t,p}^{v}, \mathrm{Ray}_{k,t,p}^{v}). \tag{4}$$

We also encode the normalized timestamp $\tau_{k,t} \in [0,1]$ with a sinusoidal timestep embedding following Ho et al. (2020):

$$e_{k,t} = \mathrm{Sinusoidal}(\tau_{k,t}). \tag{5}$$

The posed image token and timestep embedding are projected to the same transformer dimension as $\phi^v_{k,t,p}$ before fusion. The final chunk-wise latent field is obtained by summing these aligned components and is used as the latent field for the decoder:

$$\mathbf{f}^v_{k,t,p} = \phi^v_{k,t,p} + P_{\text{pose}}\left(\psi^v_{k,t,p}\right) + P_{\text{time}}\left(e_{k,t}\right). \tag{6}$$

This construction of the latent field $\mathbf{F}_k$ keeps multi-view features, low-level appearance features, camera geometry, and temporal position in the same latent space before any key-spatiotemporal query is decoded.

3.3 Key-Spatiotemporal Query Construction

To avoid dense per-frame or per-view Gaussian prediction, we construct a sparse set of key-spatiotemporal queries guided by geometric proxies, which are subsequently decoded into the final FreeTimeGS representation. Specifically, we explicitly decouple the processing of foreground and background regions using estimated masks. A key-spatiotemporal query is defined as a spatial point together with its temporal anchor, i.e., $q^d_{k,i} = \left(\mathbf{x}^d_{k,i}, t^d_k\right)$ with $d \in \{\text{fg}, \text{bg}\}$. We use the middle timestep for the foreground and the initial timestep (relative time 1) for the background. Under the indexing $t = 1, \ldots, T$, their local and chunk-level temporal anchors are

$$\begin{aligned} t_{\text{fg}} &= \left\lceil \frac{T}{2} \right\rceil, \quad t_{\text{bg}} = 1, \\ t^{\text{fg}}_k &= t_{k,t_{\text{fg}}}, \quad t^{\text{bg}}_k = t_{k,t_{\text{bg}}}. \end{aligned} \tag{7}$$

For the foreground, we construct a dense visual hull from calibrated multi-view masks at t_{fg}. We then cast rays through foreground pixels from all views and use a voxel-accelerated ray query to retain their nearest valid intersections with the hull, following Guo et al. (2026):

$$\begin{aligned} \mathcal{V}^{\text{fg}}_k &= \text{VisualHull}\left(\{M^v_{k,t_{\text{fg}}}, \Pi^v\}_{v=1}^V\right), \quad \mathcal{R}^{\text{fg}}_k = \left\{\mathbf{r}^v_{k,t_{\text{fg}},p} \mid M^v_{k,t_{\text{fg}}}(p) = 1,\ v = 1, \ldots, V\right\}, \\ \mathcal{P}^{\text{fg}}_k &= \text{Ray-Casting}\left(\mathcal{V}^{\text{fg}}_k, \mathcal{R}^{\text{fg}}_k\right), \qquad\qquad \mathcal{Q}^{\text{fg}}_k = \left\{(\mathbf{x}, t^{\text{fg}}_k) \mid \mathbf{x} \in \text{Voxelize}\left(\mathcal{P}^{\text{fg}}_k\right)\right\}. \end{aligned} \tag{8}$$

Here, $\mathbf{r}^v_{k,t_{\text{fg}},p}$ denotes the camera ray of foreground pixel p in view v. For the background, we leverage the DA3 depth maps at the initial timestep. By unprojecting and voxelizing them, we establish a coarse geometric prior. Background queries are uniformly sampled from this prior using voxelization to ensure sparsity while maintaining coverage of the background region:

$$\mathcal{P}^{\text{bg}}_k = \text{Voxelize}\left(\bigcup_{v=1}^{V} \text{Unproject}(D^v_{k,t_{\text{bg}}}, \Pi^v)\right), \qquad \mathcal{Q}^{\text{bg}}_k = \text{Sample}\left\{(\mathbf{x}, t^{\text{bg}}_k) \mid \mathbf{x} \in \mathcal{P}^{\text{bg}}_k\right\}. \tag{9}$$

These two domain-specific sequences are then concatenated to form a unified key-spatiotemporal query sequence for the k-th chunk:

$$\mathcal{Q}_k = \text{Concat}\left(\mathcal{Q}^{\text{fg}}_k, \mathcal{Q}^{\text{bg}}_k\right). \tag{10}$$

This design makes the query set sparse yet complete: the foreground visual hull covers the dynamic subject, while DA3 depth provides geometric support for background regions. It also allows foreground and background query tokens to be processed efficiently while sharing a unified latent space during the decoding stage.

3.4 4DGS Decoder

The 4DGS decoder turns the unified query sequence into a chunk-level 4DGS representation. Each query coordinate is lifted to the transformer dimension by a sinusoidal positional encoding followed by an MLP projection, and the resulting query token gathers evidence from the chunk-wise latent field $\mathbf{F}_k$ through cross-attention:

$$\mathcal{Q}^{\text{feat}}_k = \text{CrossAttn}(\mathcal{Q}_k, \mathbf{F}_k), \tag{11}$$

where $\mathcal{Q}_k^{\text{feat}}$ denotes the 'Query w/ Feature' sequence. This sequence is then processed by a 4-layer Transformer decoder to exchange information among queries:

$$\mathcal{Q}_k^{\text{dec}} = \text{Transformer}^{(4)}(\mathcal{Q}_k^{\text{feat}}), \tag{12}$$

For each decoded query feature $\mathcal{Q}_{k,i}^{\text{dec}}$, a 4DGS head predicts a residual 4D mean and the other attributes of K primitives:

$$\hat{g}_{i,j}^d = \text{Linear}^d\left(\mathcal{Q}_{k,i}^{\text{dec}}\right) = \left(\Delta\mu_{i,j}, s_{i,j}, r_{i,j}, \alpha_{i,j}, h_{i,j}, \mathbf{v}_{i,j}, \sigma_{i,j}\right), \quad j = 1, \ldots, K,\ d \in \{\text{fg}, \text{bg}\}. \tag{13}$$

where $\Delta\mu_{i,j} \in \mathbb{R}^4$ and the final 4D mean is obtained by adding the predicted residual to its query anchor, $\mu_{i,j}^d = q_{k,i}^d + \Delta\mu_{i,j}$, so the geometric support established by query construction is preserved during decoding. Here, $s_{i,j} \in \mathbb{R}^3$ and $r_{i,j} \in \mathbb{R}^4$ are the scale and rotation, $\alpha_{i,j} \in \mathbb{R}$ is the opacity, $h_{i,j} \in \mathbb{R}^m$ is the SH coefficient vector, and $\mathbf{v}_{i,j} \in \mathbb{R}^3$ and $\sigma_{i,j} \in \mathbb{R}$ are the linear velocity and temporal variance in FreeTimeGS. The foreground and background heads are separate heads with the same architecture, and they decode tokens from the corresponding query sets with different velocity ranges. Their outputs are merged to form the final 4D scene representation. Unlike a decoder that produces an independent 3D representation at every time, these heads directly predict a continuous 4D representation whose spatiotemporal attributes can be rendered at different times.

3.5 Boundary-Aware Sequential Training

Training proceeds in two stages. Stage 1 trains the feed-forward predictor on single chunks with direct supervision on rendered target views. Let $\hat{I}_t^v$ be the rendered image at target time t and view v. The reconstruction loss is

$$\mathcal{L}_{\text{rec}} = \lambda_1 \left\| \hat{I}_t^v - I_t^v \right\|_1 + \lambda_{\text{lpips}} \text{LPIPS}(\hat{I}_t^v, I_t^v). \tag{14}$$

For longer sequences, however, independent chunk supervision can still produce incompatible renderings at chunk boundaries. Stage 2 therefore introduces Boundary-Aware Sequential Training (BAST), which fine-tunes the Stage-1 model by supervising overlapping temporal regions covered by neighboring chunks. BAST trains on batches of M consecutive chunks $\{\mathcal{C}_k, \ldots, \mathcal{C}_{k+M-1}\}$. Let Ω_j denote the target times shared by chunks j and $j+1$. For the same time and view, both chunks should render compatible images:

$$\mathcal{L}_{\text{BAST}} = \lambda_{\text{BAST}} \sum_{j=k}^{k+M-2} \sum_{t \in \Omega_j, v} \left\| \hat{I}_{j,t}^v - \hat{I}_{j+1,t}^v \right\|_2^2. \tag{15}$$

BAST does not require the two chunks to predict identical Gaussian sets; it only encourages their rendered outputs to agree where they describe the same observed time. The final training objective is

$$\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{BAST}}. \tag{16}$$

Optional Test-Time Optimization. We optionally refine the predictions for long sequences at test time. Starting from the predicted cross-chunk 4DGS, test-time optimization (TTO) uses the same image reconstruction loss on input views and the same BAST consistency loss on shared target times between neighboring chunks. This refinement supervises the predicted 4DGS across chunks without using held-out novel-view targets.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our method on the DNA-Rendering Cheng et al. (2023) and Neural 3D Video (Neural3DV) Li et al. (2022) datasets. For DNA-Rendering, we construct two evaluation protocols: a dynamic-only region evaluation using foreground masks and a full-image evaluation incorporating background content. Following prior work Jin et al. (2025); Guo et al. (2026), we designate 16

Table 1: **Quantitative comparison on the DNA-Rendering test set.** Metrics are evaluated on the masked dynamic foreground using 8 input views at 512×512 resolution across all 150 frames. GS/chunk counts foreground primitives; Time/frame is the per-frame reconstruction latency in seconds. Green and yellow cells indicate the best and second-best results.

Method	Type	PSNR ↑	SSIM ↑	LPIPS ↓	Flickering ↑	GS/chunk ↓	Time/frame (s) ↓
LHM++ Qiu et al. (2025b)	Human-specific	21.62	0.791	0.179	95.83	160k	2.09
4DGS Yang et al. (2024)	Optimization-based	20.41	0.741	0.243	95.82	-	34.64
FreeTimeGS Wang et al. (2025)	Optimization-based	21.88	0.751	0.259	96.85	-	14.84
GPS-Gaussian Zheng et al. (2024b)	Feed-forward 3D	22.35	0.797	0.172	95.05	816k	0.28
PointSplat Guo et al. (2026)	Feed-forward 3D	26.98	0.885	**0.083**	96.10	284k	0.43
L4GM Ren et al. (2024)	Feed-forward 4D	20.22	0.758	0.256	95.66	524k	0.25
Ours	Feed-forward 4D	26.09	0.861	0.094	96.45	**77k**	**0.23**
Ours w TTO	Feed-forward 4D + TTO	**27.85**	**0.895**	0.096	**96.86**	**77k**	11.09

sequences as the test set. Neural3DV comprises general dynamic multi-view scenes and is evaluated exclusively under the full-image protocol. The train-test splits follow Wang et al. (2025). We train our model separately on each dataset, using its train-test split for evaluation.

Baselines. For DNA-Rendering dynamic regions, we compare with 4DGS Yang et al. (2024), FreeTimeGS Wang et al. (2025), GPS-Gaussian Zheng et al. (2024b), PointSplat Guo et al. (2026), L4GM Ren et al. (2024), and LHM++ Qiu et al. (2025b). For full-image rendering on DNA-Rendering and Neural3DV, we compare with FreeTimeGS, 4DGS, and L4GM; LHM++, GPS-Gaussian, and PointSplat are not designed for full-image general dynamic scenes. Since L4GM was originally trained with 4 views, 8 timesteps, and 256×256 images, we adapt it separately on each dataset under our training protocol. Other baselines use original implementations. In the dynamic-region protocol, all methods use the same BiRefNet Zheng et al. (2024a) masks.

Metrics. To assess rendering quality, we report PSNR, SSIM, and LPIPS. Moreover, we adopt the Temporal Flickering score in VBench Huang et al. (2024) to measure temporal stability of rendered videos (higher is better). To evaluate representation efficiency, we report the average number of Gaussian primitives required to represent a temporal chunk (GS/chunk) and reconstruction latency per frame, computed as the total wall-clock reconstruction time of a sequence divided by its number of frames (details in Sec. A.5). Runtimes are measured on NVIDIA H20 hardware. *Ours* runtime includes the full pipeline in Sec. 3.

Implementation Details. The encoder is initialized with the DA3-B backbone weights. The query number is adaptive to the occupied volume; on DNA-Rendering, the foreground visual hull yields approximately 5k queries per chunk on average, which decode into about 77k foreground Gaussians with $K = 16$ primitives per query. Training follows the two-stage schedule in Sec. 3.5. Full architecture, optimization, sequential-inference, and TTO settings are provided in Sec. A.2.

4.2 Dynamic-Region Results on DNA-Rendering

Table 1 presents a quantitative comparison of rendering quality, temporal stability, representation compactness, and runtime, focusing exclusively on the masked dynamic foreground in the DNA-Rendering test set. GS/chunk measures the number of foreground Gaussian primitives used for one temporal chunk, reflecting the compactness gained by sharing primitives across frames. Corresponding qualitative results are shown in Figure 3. For all methods, we use 8 input views at 512×512 resolution for reconstruction and evaluate novel view synthesis on the remaining 40 target views. We also report the optional test-time optimization results described in Sec. 3.5.

PointSplat is spatially comparable to PointSplatV2, whereas PointSplatV2 yields higher temporal stability (96.45 vs. 96.10), uses $3.7\times$ fewer foreground primitives, and has $1.9\times$ lower latency. Relative to L4GM, PointSplatV2 gains 5.87 dB PSNR, 0.103 SSIM, and 0.162 LPIPS, with lower flickering, fewer primitives, and lower latency: L4GM exchanges temporal features but still emits independent per-frame 3D Gaussian maps rather than shared 4D primitives. FreeTimeGS and 4DGS score high on flickering but are visibly blurrier.

4.3 Results on General Dynamic Scenes

Table 2 and Figure 4 report full-image quality and runtime on DNA-Rendering and Neural3DV. Optimization-based methods often overfit sparse inputs; their high Flickering scores can coincide with smooth, detail-loss renderings. Although adapted L4GM is fastest, it is less sharp and

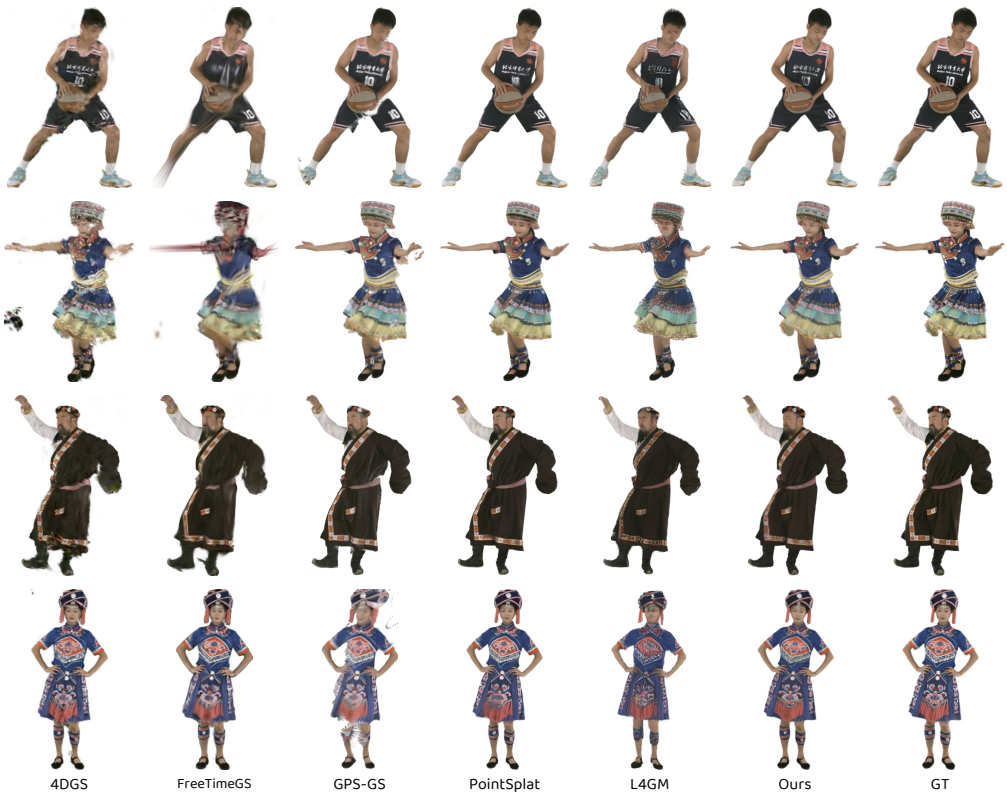

Figure 3: **Qualitative comparison of dynamic regions on DNA-Rendering.** To facilitate a clear evaluation of moving content, backgrounds are masked out. Our method recovers sharper high-frequency details and maintains better structural integrity under sparse-view constraints.

Table 2: **Full-image quantitative results on DNA-Rendering and Neural3DV.** Metrics are evaluated on full-frame images at 512×512 resolution using 8 input views. Time/frame denotes per-frame reconstruction latency.

	DNA-Rendering					Neural3DV				
Method	PSNR ↑	SSIM ↑	LPIPS ↓	Flickering ↑	Time/frame (s) ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Flickering ↑	Time/frame (s) ↓
FreeTimeGS	14.86	0.793	0.535	99.89	17.80	24.43	0.900	0.221	99.94	12.32
4DGS	16.46	0.750	0.487	99.55	89.68	26.34	0.887	0.177	99.87	42.36
L4GM	23.37	0.814	0.463	99.43	**0.25**	23.31	0.674	0.525	99.86	**0.25**
Ours	24.08	0.825	0.308	99.44	0.65	28.55	0.902	0.096	99.30	1.28
Ours w TTO	**25.44**	**0.852**	**0.269**	**99.90**	14.90	**31.03**	**0.937**	**0.086**	**99.98**	18.28

trails PointSplatV2 on all spatial metrics for both datasets. PointSplatV2 attains the best non-TTO quality with competitive temporal consistency and practical latency; TTO gives the best quality at optimization-comparable runtime.

4.4 Ablation Study

Table 3 compares the full query design with two query-construction ablations and one training ablation. The full model uses middle-frame foreground visual-hull queries and initial-frame background DA3-depth queries. *All-frame hull union* replaces the compact foreground queries with visual-hull points pooled across all input frames, testing temporal key selection. *DA3-depth-only queries* removes foreground/background query decoupling and constructs all queries by DA3-depth back-projection, testing domain-specific geometric support. *No BAST fine-tuning* evaluates the Stage-1 checkpoint under the same inference window, testing overlap consistency supervision. Both query ablations reduce quality and compactness, while removing BAST mainly reduces PSNR and Flickering, confirming the importance of key-spatiotemporal queries and boundary-aware supervision.

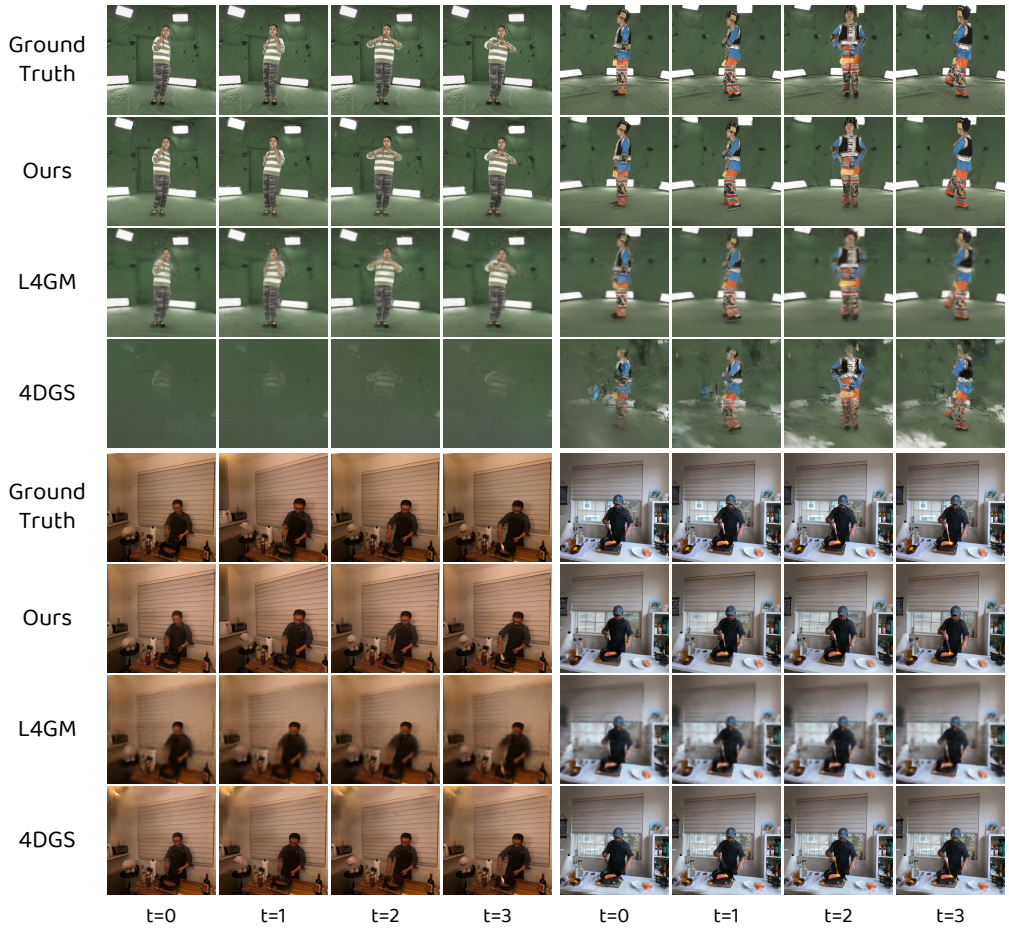

Figure 4: **Full-image comparisons on dynamic scenes.** PointSplatV2 preserves motion details and sharp structures better than over-smoothed L4GM and optimization-based baselines.

Table 3: **Ablation of proposed components on the foreground region of DNA-Rendering.**

Variant	PSNR ↑	LPIPS ↓	Flickering ↑	GS/chunk ↓
All-frame hull union	20.68	0.213	96.22	153k
DA3-depth-only queries	20.22	0.230	96.09	901k
No BAST fine-tuning	23.85	0.123	96.21	**77k**
Full model	**26.09**	**0.094**	**96.45**	**77k**

5 Conclusion and Limitations

We presented PointSplatV2, a query-based feed-forward framework that reconstructs continuous 4D Gaussians from sparse multi-view videos. Key-spatiotemporal queries aggregate temporal chunks into a compact, temporally consistent 4D representation, replacing separate per-timestep predictions, and Boundary-Aware Sequential Training ensures compatibility across overlapping chunks for stable long-sequence reconstruction.

Limitations. PointSplatV2 enables efficient feed-forward 4D reconstruction but faces two constraints: limited input resolution and generalization bounds inherent to feed-forward approaches trained on finite multi-view datasets. Future work addressing these resolution and data scalability issues will be crucial for robust real-world volumetric video systems.

AI USE STATEMENT

In preparing this manuscript, we used generative AI tools only for recommended-disclosure tasks such as improving readability and organizing the presentation. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, design or provide feedback on the methodology or experiments, implement methods, clean or reformat datasets, or interpret results. All AI-assisted material was reviewed and verified by the authors. We take responsibility for the final content of this submission.

REFERENCES

Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

David Charatan, Sizhe Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian Splats from Image Pairs for Scalable Generalizable 3D Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3D Gaussian Splatting from Sparse Multi-View Images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

Ziwen Chen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Wei Cheng, Ruixiang Chen, Wanqi Yin, Siming Fan, Keyu Chen, Honglin He, Huiwen Luo, Zhongang Cai, Jingbo Wang, Yang Gao, Zhengming Yu, Zhengyu Lin, Daxuan Ren, Lei Yang, Ziwei Liu, Chen Change Loy, Chen Qian, Wayne Wu, Dahua Lin, Bo Dai, and Kwan-Yee Lin. Dna-rendering: A diverse neural actor repository for high-fidelity human-centric rendering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2023.

Insung Cho, Jae Shin Yoon, Hyeonseop Jeon, and Kuk-Jin Yoon. 4d scaffold gaussian splatting with dynamic-aware anchor growing for efficient and high-fidelity dynamic scene reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 40, pp. 3363–3371, 2026. doi: 10.1609/aaai.v40i5.37332.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks, 2017.

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 (CVPR)*, 2023.

Yujie Guo, Yudong Jin, Lingteng Qiu, Zehong Shen, Zhen Xu, Jing Zhang, Xianchao Shen, Hujun Bao, Sida Peng, and Xiaowei Zhou. Pointsplat: Compact gaussian splatting via human-centric prediction. *arXiv preprint arXiv:2606.32036*, 2026.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Yudong Jin, Sida Peng, Xuan Wang, Tao Xie, Zhen Xu, Yifan Yang, Yujun Shen, Hujun Bao, and Xiaowei Zhou. Diffuman4d: 4d consistent human view synthesis from sparse-view videos with spatio-temporal diffusion models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2025.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4), 2023.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, and Zhaoyang Lv. Neural 3d video synthesis from multi-view video, 2022.

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 (CVPR)*, 2024.

Chieh Lin, Zhaoyang Lv, Songyin Wu, Zhen Xu, Thu Nguyen-Phuoc, Hung-Yu Tseng, Julian Straub, Numair Khan, Lei Xiao, Ming-Hsuan Yang, Yuheng Ren, Richard Newcombe, Zhao Dong, and Zhengqin Li. Dgs-lrm: Real-time deformable 3d gaussian reconstruction from monocular videos. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *Proceedings of the ACM SIGGRAPH Conference and Exhibition on Computer Graphics and Interactive Techniques in Asia (SIGGRAPH Asia)*, 2022.

Haotong Lin, Sida Peng, Zhen Xu, Tao Xie, Xingyi He, Hujun Bao, and Xiaowei Zhou. High-fidelity and real-time novel view synthesis for dynamic scenes. In *SIGGRAPH Asia Conference Proceedings*, 2023.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL `https://arxiv.org/abs/1711.05101`.

Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021a.

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. *ACM Transactions on Graphics (TOG)*, 40 (6), 2021b.

Sida Peng, Chen Geng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Xiaowei Zhou, and Hujun Bao. Implicit neural representations with structured latent codes for human body modeling. *TPAMI*, 2023.

Lingteng Qiu, Xiaodong Gu, Peihao Li, Qi Zuo, Weichao Shen, Junfei Zhang, Kejie Qiu, Weihao Yuan, Guanying Chen, Zilong Dong, and Liefeng Bo. Lhm: Large animatable human reconstruction model from a single image in seconds. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2025a.

Lingteng Qiu, Peihao Li, Heyuan Li, Qi Zuo, Xiaodong Gu, Yuan Dong, Weihao Yuan, Rui Peng, Siyu Zhu, Xiaoguang Han, Guanying Chen, and Zilong Dong. Lhm++: An efficient large human reconstruction model for pose-free images to 3d. *arXiv preprint arXiv:2503.10625*, 2025b.

Jiawei Ren, Kevin Xie, Ashkan Mirzaei, Hanxue Liang, Xiaohui Zeng, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung Wook Kim, and Huan Ling. L4GM: Large 4D Gaussian Reconstruction Model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

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 (CVPR)*, pp. 20310–20320, June 2024.

Jiahao Wu, Jie Peng, Die Zhang, Jiayu Zhai, Kaiqiang Xiong, Xiaoyun Zheng, Xiang Xiang, Chao Lin, and Ronggang Wang. Atgs: Anchored temporal gaussian splatting for long volumetric video representation. *ACM Transactions on Graphics (TOG)*, 45(4), 2026. doi: 10.1145/3811306.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arXiv preprint arXiv:2403.14621*, 2024a.

Zhen Xu, Yinghao Xu, Zhiyuan Yu, Sida Peng, Jiaming Sun, Hujun Bao, and Xiaowei Zhou. Representing long volumetric video with temporal gaussian hierarchy. *ACM Transactions on Graphics (TOG)*, 43(6), 2024b.

Zhen Xu, Zhengqin Li, Zhao Dong, Xiaowei Zhou, Richard Newcombe, and Zhaoyang Lv. 4dgt: Learning a 4d gaussian transformer using real-world monocular videos. In *Advances in Neural Information Processing Systems*, volume 38, 2025b.

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

Peng Zheng, Dehong Gao, Deng-Ping Fan, Li Liu, Jorma Laaksonen, Wanli Ouyang, and Nicu Sebe. Bilateral reference for high-resolution dichotomous image segmentation. *CAAI Artificial Intelligence Research*, 3:9150038, 2024a.

Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

A SUPPLEMENTARY MATERIAL

REPRODUCIBILITY STATEMENT

The paper provides the model formulation, training objective, evaluation protocols, and implementation details. The supplement adds qualitative results, ablations, robustness analysis, and training details needed to reproduce the reported experiments.

A.1 ADDITIONAL QUALITATIVE RESULTS

Fig. 5 provides additional masked dynamic-region comparisons on DNA-Rendering, where PointSplatV2 better preserves subject contours and local appearance details under sparse views. Fig. 6 shows full-image results on Neural3DV, confirming coherent subject appearance and scene structure on general dynamic scenes.

A.2 ADDITIONAL IMPLEMENTATION AND TRAINING DETAILS

The encoder is initialized with DA3-B and projects its multi-scale features to 1024 hidden dimensions. The visual-hull grid resolution is 0.003, and points are voxelized with a voxel size of 0.006. We set the velocity range to 0.5 for the foreground head and 0.005 for the background head.

Stage 1 samples 8 context views spanning 4 frames and trains for 200k steps. Stage 2 applies BAST to $M = 3$ consecutive chunks with one-frame overlap between neighboring chunks, corresponding to 10 frames per batch, for 100k steps; the DA3 encoder is frozen and the 4DGS decoder is updated. We use $\lambda_1 = 1.0$ and $\lambda_{lpips} = 0.8$ throughout and $\lambda_{\text{BAST}} = 1.0$ in Stage 2, with the AdamW Loshchilov & Hutter (2019) optimizer, a learning rate of 1×10^{-4}, and a total batch size of 32 on 32 NVIDIA H20 GPUs.

At test time, sequential inference processes a long sequence chunk by chunk with window length $W = 4$, overlap $O = 1$, and stride $S = 3$. Each chunk renders the frames it covers, and the shared boundary frame is rendered by the earlier chunk; this hard handoff uses neither image-space blending nor primitive merging. We run optional TTO for 120 iterations with a learning rate of 1×10^{-3}.

A.3 ADDITIONAL ARCHITECTURE DETAILS

Background queries are sampled from the DA3 depth maps at the first timestep $t_{\text{bg}} = 1$ of each chunk, so regions that are occluded initially but become disoccluded later lack direct query support. The short temporal window ($T = 4$) bounds the extent of such disocclusion, and the residual offsets predicted relative to query anchors allow nearby background primitives to adjust to small newly



Figure 5: **More qualitative comparisons on the DNA-Rendering dynamic region.**



Figure 6: **More qualitative comparisons on Neural3DV.** We show visual results on general dynamic scenes under 8-view inputs. Our method achieves competitive rendering quality relative to existing methods while reducing computational overhead and reconstruction time per frame.

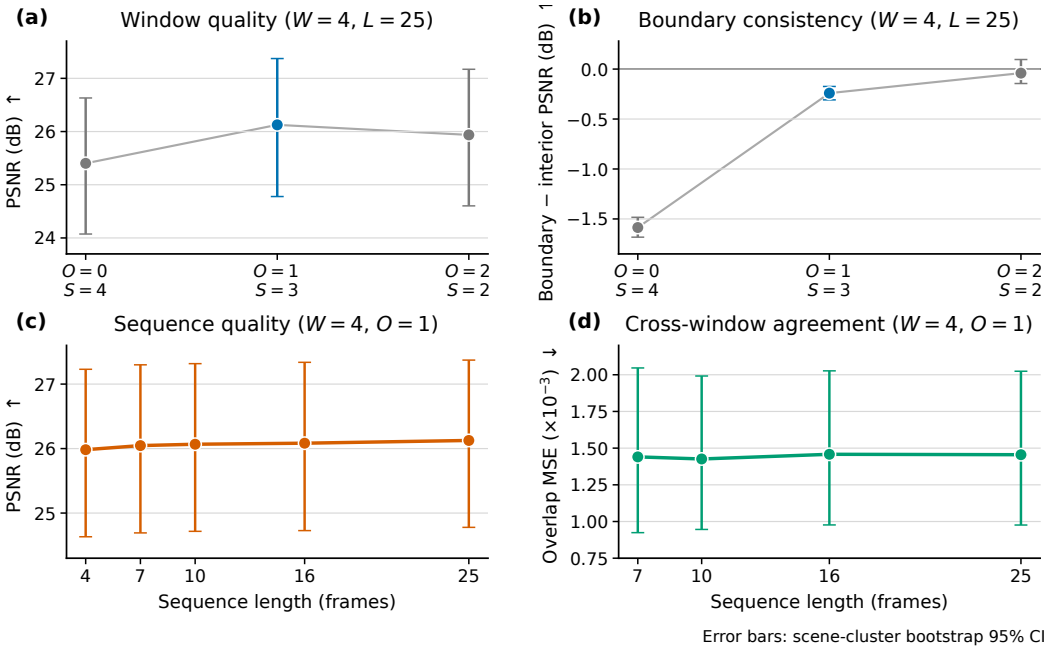

Figure 7: **Analysis of Boundary-Aware Sequential Training.** (a–b) For a four-frame window at sequence length 25, one-frame overlap gives the highest mean PSNR and reduces the boundary penalty. (c–d) Under the default $W = 4$, $O = 1$, $S = 3$ setting, reconstruction quality and cross-window agreement remain stable as sequence length increases. Error bars show scene-cluster bootstrap 95% confidence intervals.

visible regions. Pooling background depth maps across chunk timesteps could further improve geometric completeness at the cost of additional queries.

A.4 Analysis of Boundary-Aware Sequential Training

While Tab. 3 shows the overall benefit of BAST, we analyze its boundary behavior on the DNA-Rendering test set using the same evaluation protocol as Sec. 4.2. We denote the window length, overlap, and stride by W, O, and S, respectively, and compute 95% confidence intervals using 10,000 scene-cluster bootstrap samples. All configurations in Fig. 7(a–b) are evaluated with the same BAST-trained model by only changing the inference-time overlap and stride; no separate models are trained per configuration.

Table 4: **Robustness to geometry-prior perturbations.** We use the same full-image DNA-Rendering protocol as Tab. 2. Representative perturbations are reported here; Fig. 8 shows the complete sweep.

Setting	PSNR ↑	SSIM ↑	LPIPS ↓	Flickering ↑
Clean geometry	24.08	0.825	0.308	99.44
DA3 depth noise (2%)	22.24	0.810	0.341	99.44
Context pose (0.5°, 0.5%)	22.00	0.794	0.336	99.47
Mask boundary noise (4 px)	23.06	0.809	0.327	99.46

With window length W and stride S, chunk j covers frames $[jS + 1, jS + W]$; we call frame f a boundary frame if it is covered by two consecutive chunks ($f = jS + W$ for some chunk j), and an interior frame otherwise. Following the hard-handoff inference protocol, boundary PSNR is computed on boundary frames using the earlier chunk's rendering, and the boundary-minus-interior gap subtracts the mean interior PSNR from the mean boundary PSNR. Overlap MSE is the mean squared error between the two chunks' renderings of each boundary frame.

Figure 7 shows that removing overlap at $W = 4$ gives a -1.586 dB boundary-minus-interior PSNR gap, whereas one-frame overlap reduces this gap to -0.241 dB and gives the highest mean PSNR of 26.126 dB. Two-frame overlap narrows the gap to -0.040 dB, but does not improve mean PSNR and reduces the stride from 3 to 2; this supports the one-frame setting as a trade-off between reconstruction quality and redundant computation. With this setting, mean PSNR remains near 26 dB from 4 to 25 frames, while overlap MSE stays within 0.001426–0.001458 from 7 to 25 frames. Because each chunk is predicted independently from its own input frames and no recurrent state is propagated across chunks, cross-window disagreement should not accumulate with sequence length. The near-constant overlap MSE from 7 to 25 frames supports this.

A.5 Runtime Reporting

Time/frame in Tab. 1 and Tab. 2 denotes reconstruction latency per frame: for each sequence, we divide the total wall-clock time required for reconstruction by its number of frames. For PointSplatV2, this includes the full feed-forward pipeline, i.e., DA3 feature and depth extraction, query construction, and decoding. For optimization-based baselines, it denotes the per-scene optimization time amortized over the sequence frames, not the rendering frame rate after reconstruction; once fitted, explicit Gaussian rasterizers render at interactive speeds.

A.6 Robustness to Geometric Priors

We use the same full-image DNA-Rendering protocol as Tab. 2: 8 input views at 512×512 resolution and full-frame metrics, without TTO. Holding all other inputs fixed, we perturb aligned DA3 depth before unprojection, context poses with local SE(3) noise, or context-mask boundaries with random dilation or erosion. Each factor shares the same random seed.

Tab. 4 reports representative perturbations and Fig. 8 shows the full sweep. Moderate depth and mask noise retain 22.24 and 23.06 dB PSNR, respectively, versus 24.08 dB with clean geometry, and change Flickering by at most 0.02 points. Pose noise has a larger effect: $0.5°/0.5\%$ noise reduces PSNR to 22.00 dB. All spatial metrics decline with stronger perturbations, particularly for pose noise, showing that calibrated context geometry remains important. The Flickering score changes little across perturbations. Optical-flow-based flickering metrics can remain stable or increase slightly when reconstructions lose high-frequency detail and become smoother, so this stability does not imply that temporal modeling is unaffected.

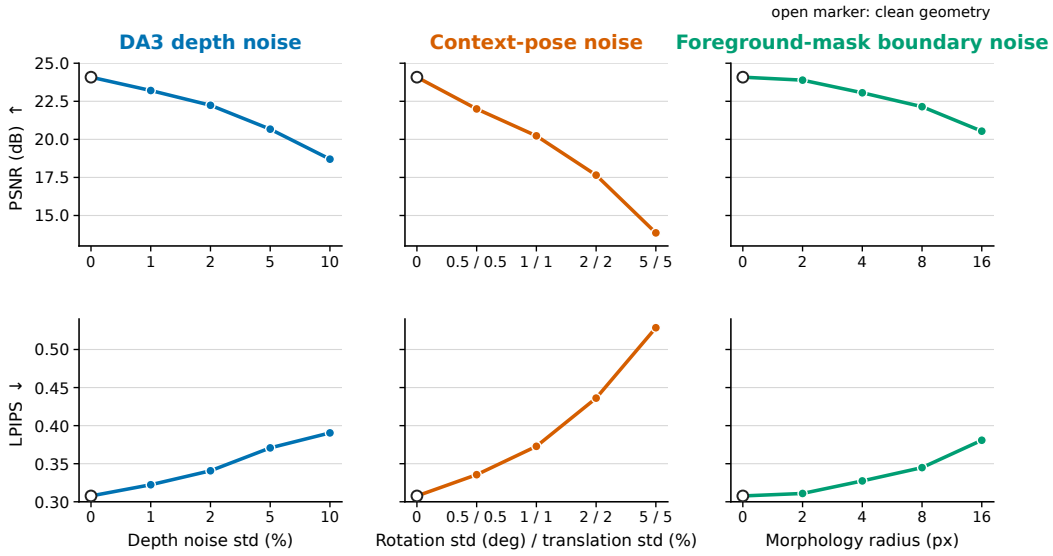

Figure 8: **Robustness to geometry-prior perturbations.** We perturb one input prior at a time while keeping the sparse-view reconstruction and evaluation protocol fixed. The open marker denotes clean geometry. Quality degrades monotonically as the perturbation magnitude increases; camera-pose errors cause the largest degradation, whereas moderate mask-boundary noise has a smaller effect.