SmoothMotionVectors: Optimizing Your Content for Video Codecs in Free View Video Compression

MINGYANG SONG, DisneyResearch|Studios, Switzerland and ETH Zürich, Switzerland
YANG ZHANG, DisneyResearch|Studios, Switzerland
SIYU TANG, ETH Zürich, Switzerland
TUNÇ OZAN AYDIN, DisneyResearch|Studios, Switzerland

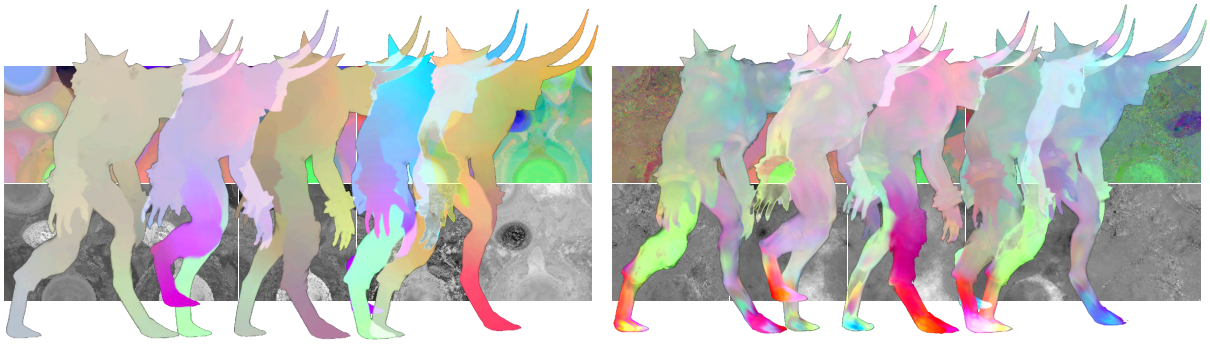
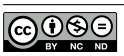


Fig. 1. Conventional video codecs exploit spatial and temporal self-similarity to efficiently compress structured signals. We bring this principle to dynamic scene reconstruction with 3D Gaussian Splatting by organizing deformation outputs through a tailored training and compression pipeline. The resulting motion vectors are smoother across space and time (left), allowing greater storage reduction without sacrificing visual quality. When this structure is not preserved (right), file sizes grow and quantization artifacts become noticeable. We show per-timestep offsets relative to the first frame and their rearranged representations using SOG [Morgenstern et al. 2024]. This sample scene is part of the D-NeRF [Pumarola et al. 2021] dataset.

We present an empirical study of free-view video compression for dynamic scenes reconstructed with 3D Gaussian Splatting, examining how practical pipeline design choices affect reconstruction fidelity and storage efficiency. Rather than introducing new representations, we analyze how commonly used components, including temporal chunking, deformation-based reconstruction, and quantization-aware training, interact in practice. We observe that partitioning long sequences into shorter temporal segments, such as GOPs, simplifies optimization and improves reconstruction fidelity, but can introduce additional storage overhead. We further show that encouraging smooth motion vectors across both space and time produces deformation signals that are easier for standard video codecs to compress, leading to improved rate–distortion performance. When integrated into a unified pipeline, these design choices consistently benefit different deformation-based reconstruction methods. Across multiple datasets, our approach achieves 20% storage reduction compared with state-of-the-art methods while preserving or improving visual quality, and we discuss sources of variability and ambiguity in current training and evaluation protocols.

Authors' Contact Information: Mingyang Song, DisneyResearch|Studios, Switzerland and ETH Zürich, Switzerland, mingyang.song@inf.eth.ch; Yang Zhang, DisneyResearch|Studios, Switzerland, yang.zhang@disneyresearch.com; Siyu Tang, ETH Zürich, Switzerland, siyu.tang@inf.ethz.ch; Tunç Ozan Aydın, DisneyResearch|Studios, Switzerland, tunc@disneyresearch.com.



SIGGRAPH Conference Papers '26, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2554-8/2026/07
https://doi.org/10.1145/3799902.3811112

CCS Concepts: • **Computing methodologies** → **Reconstruction**; **Image compression**.

ACM Reference Format:
Mingyang Song, Yang Zhang, Siyu Tang, and Tunç Ozan Aydın. 2026. SmoothMotionVectors: Optimizing Your Content for Video Codecs in Free View Video Compression. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3799902.3811112

1 Introduction

Recent advances have significantly improved static scene reconstruction from images, with the goal of recovering geometry and appearance aligned with input views for free-view exploration and downstream applications. Neural Radiance Fields (NeRF) [Mildenhall et al. 2021] and subsequent variants represent scene properties as functions of 3D position using neural networks. Despite notable improvements in reconstruction quality and convergence speed, these neural primitives [Müller et al. 2022] remain limited in representational capacity and computational efficiency. In contrast, 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] adopts explicit anisotropic 3D ellipsoids from classical graphics and achieves differentiable volumetric rendering with high reconstruction fidelity and rendering speed, making it a dominant approach in recent work.

Dynamic scene reconstruction methods typically extend these static formulations. NeRF-based approaches incorporate time either as an additional input or through time-varying network parameters [Cao and Johnson 2023; Fridovich-Keil et al. 2023; Knodt 2022;

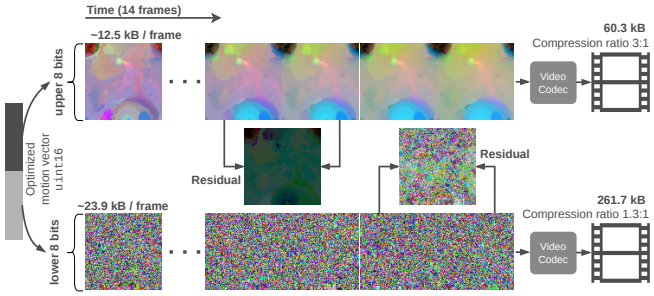


Fig. 2. **A representative example illustrating how spatial and temporal coherence influence compression ratio.** Conventional video codecs exploit spatial and temporal redundancy, which is reflected in per-frame file size and overall video size. Differences between adjacent frames highlight the importance of temporal coherence, as lower-magnitude residuals are easier to compress.

Li et al. 2022b; Mihajlovic et al. 2024; Park et al. 2021a,b; Pumarola et al. 2021; Shao et al. 2023; Yan et al. 2023]. GS-based methods instead model temporal variation using native 4D primitives [Duan et al. 2024; Yang et al. 2024a,b], explicit parametric trajectories [Lee et al. 2024; Li et al. 2024; Lin et al. 2024; Park et al. 2025], or deformation fields adapted from NeRF-based formulations [Dai et al. 2025; Wu et al. 2023; Xu et al. 2024; Yang et al. 2023]. Hybrid approaches further combine these strategies to mitigate their respective limitations [Song et al. 2025; Yoon et al. 2025].

In this work, we adopt 3D Gaussian Splatting as the backbone and model temporal variation using deformation fields $\Phi_\theta(\cdot)$ that warp a shared canonical template across time. While such representations support many downstream applications, we focus specifically on reconstruction fidelity and storage efficiency. Prior studies [Dai et al. 2025; Li et al. 2025; Wang et al. 2024] establish strong baselines by drawing inspiration from conventional 2D video compression. However, a key inductive bias of natural videos is often underemphasized: content that is smoother in space and time exhibits lower entropy and is therefore easier to compress, as illustrated in Fig. 2.

Rather than proposing new deformation formulations, we study how existing deformation-based reconstruction methods can better exploit spatial and temporal regularity through appropriate pipeline design. As demonstrated across Grid4D [Xu et al. 2024] and SpDeF [Song et al. 2025] in Sec. 4.4, we examine a set of practical design choices, including temporal chunking, training schedules, quantization-aware optimization, and content-aware organization of motion vectors. We show that these choices lead to smoother motion representations that are more amenable to compression across different deformation field designs. In addition, we present representative challenging cases observed in deformation-based methods, with the goal of informing future system and pipeline design.

2 Related Work

2.1 3DGS and its Compression

Vanilla 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] represents a static scene using anisotropic 3D Gaussian primitives. Each primitive is parameterized by a 3D position $\mathbf{x} \in \mathbb{R}^{N\times 3}$, a rotation represented as a quaternion $\mathbf{q} \in \mathbb{R}^{N\times 4}$, a scale $\mathbf{s} \in \mathbb{R}^{N\times 3}$, an opacity $\sigma \in \mathbb{R}^{N\times 1}$, a base color $\mathbf{c}_{\text{dc}} \in \mathbb{R}^{N\times 3}$, and a specular color $\mathbf{c}_{\text{rest}} \in \mathbb{R}^{N\times 45}$, where N denotes the number of Gaussian primitives. We refer to each primitive as a point in the remainder of the manuscript. These points are rasterized onto camera planes via volumetric rendering. The initial point set is obtained from COLMAP [Schönberger and Frahm 2016] and is refined through densification to improve coverage in under-reconstructed regions. In this work, we adopt the improved densification strategy from AbsGS [Ye et al. 2024] and refer readers to the original papers for further details.

Compression methods for 3DGS can be broadly categorized based on whether neural networks are involved. Methods that operate directly on explicit attributes rely primarily on bit quantization and vector quantization [Fan et al. 2024; Papantonakis et al. 2024; Xie et al. 2024]. When neural components such as grid- or plane-based representations are introduced [Cao and Johnson 2023; Müller et al. 2022], entropy models can be incorporated for context-aware compression [Chen et al. 2024, 2025; Zhan et al. 2025]. SOG [Morgenstern et al. 2024] exemplifies a direction of compression on explicit attributes by reorganizing them into spatially smooth images and compressing them with standard compression codecs.

2.2 3DGS-based Dynamic Scene Reconstruction

Dynamic 3DGS methods can model temporal trajectories explicitly using line segments [Duan et al. 2024; Yang et al. 2024b] or polynomials [Li et al. 2024; Lin et al. 2024]. While effective for large or fast motions, these representations require points to be retired once their trajectories are no longer valid, leading to redundancy and increased storage in long sequences. In contrast, deformation field–based methods [Song et al. 2025; Wang et al. 2025; Wu et al. 2023, 2025; Xu et al. 2024; Yang et al. 2023; Yoon et al. 2025] deform a shared canonical space over time, allowing points to maintain consistent identities across the sequence.

Designing $\Phi_\theta(\cdot)$ to balance flexibility and smoothness is nontrivial, particularly for long sequences. Training requires rendering full images across multiple timesteps, during which geometry and appearance may vary substantially, leading to large gradient fluctuations and unstable optimization. In deformation-based pipelines, point attributes at time t are obtained as:

$$\begin{gathered}\delta\mathbf{x}_t, \delta\mathbf{q}_t = \Phi_\theta(\mathbf{x}_c, t),\\ \mathbf{x}_t = \mathbf{x}_c + \delta\mathbf{x}_t, \quad \mathbf{q}_t = \mathbf{q}_c + \delta\mathbf{q}_t,\\ \mathbf{s}_t = \mathbf{s}_c, \quad \sigma_t = \sigma_c, \quad [\mathbf{c}_{\text{dc}}, \mathbf{c}_{\text{rest}}]_t = [\mathbf{c}_{\text{dc}}, \mathbf{c}_{\text{rest}}]_c,\end{gathered} \tag{1}$$

where canonical attributes and deformation parameters are jointly optimized. In practice, the canonical space $\mathbf{x}_c$ is not predefined and tends to converge to an average over time, which can deviate significantly from individual frames in long sequences.

Photometric supervision and regularization are applied after rasterization:

$$\begin{gathered}\hat{I} = \text{Rasterize}(\mathbf{x}_t, \mathbf{q}_t, \mathbf{s}_t, \sigma_t, [\mathbf{c}_{\text{dc}}, \mathbf{c}_{\text{rest}}]_t),\\ \mathcal{L}_{recon} = ||I - \hat{I}|| + \lambda_{reg}\mathcal{R}(\mathbf{x}_c, \theta),\end{gathered} \tag{2}$$

where $\mathcal{R}(\cdot)$ enforces local rigidity or smoothness [Duan et al. 2024; Gao et al. 2024; Huang et al. 2023; Kilian et al. 2007; Oblak et al. 2024; Prokudin et al. 2023; Song et al. 2025; Sorkine and Alexa 2007].

Although neural inductive biases and regularization help stabilize training, the optimization challenges are not fully resolved.

To mitigate these issues, long sequences are commonly divided into temporal windows. Chunking reduces motion complexity and improves optimization stability but increases storage roughly with the number of segments. A key challenge is therefore to retain the scalability and reconstruction benefits of chunking while offsetting the resulting storage overhead through compression.

3 Method

3.1 Training Schedule

Inspired by video compression standards [Marpe et al. 2006; Sze et al. 2014], we describe the main stages of 4D video compression as follows: 1) The video is segmented into temporal chunks, which may be overlapping or disjoint, analogous to the Group of Pictures (GOP) structure in traditional video coding; 2) Dynamic reconstruction techniques are employed to fit the frames within each GOP; 3) All time-varying attributes are estimated via $\Phi_\theta(\cdot)$; 4) These attributes are then reformulated as a combination of a reference (typically the first frame) plus offsets; 5) Compression methods are subsequently applied to both the reference and the offsets. We use C to denote the total number of GOPs and place it as a superscript to indicate the GOP index. Each GOP contains T frames. Adjacent GOPs share O overlapping frames. For convenience, the subscript t now denotes the relative time within each GOP.

As illustrated in Fig. 3, our training proceeds sequentially over GOPs, reflecting a deliberate trade-off between storage efficiency and parallelism. While a more parallel design could improve computational efficiency by removing dependencies on earlier outputs and mitigating error accumulation, it also makes preserving the temporal consistency of the static background considerably more difficult. Because Gaussian splats approximate image content with ellipsoids, visually similar regions across GOPs may still correspond to spatially misaligned structures, making reliable correspondence estimation challenging. As a result, a parallel scheme would likely require storing static background content independently for different GOPs, thereby increasing the overall file size, as shown in Fig. 7. We therefore adopt sequential encoding to better preserve background consistency and improve compression efficiency, while recognizing that introducing greater parallelism will be important for scaling to very long sequences. We refer the reader to the supplementary material for a detailed description of each stage.

3.2 Dynamic Scene Reconstruction

We consider representative grid-based designs of the deformation field $\Phi_\theta(\cdot)$ and refer readers to the original papers for implementation details. In our experiments, we focus on two widely used formulations, SpDeF [Song et al. 2025] and Grid4D [Xu et al. 2024], which serve as the base reconstruction methods in our pipeline. Throughout the benchmarking tables, we denote our full pipeline as Ours+[base method]. Both methods decompose the 4D space into lower-dimensional components to enable efficient and localized updates: SpDeF adopts a tri-plane representation with temporal planes, while Grid4D relies on a quad-grid decomposition with hash-based indexing following InstantNGP [Müller et al. 2022]. As defined in Eq. 1, we restrict the time-varying attributes to the point center and rotation.

With Grid4D [Xu et al. 2024] as the underlying representation, our approach can be contrasted with 4DMotionLayer [Dai et al. 2025] in formulation and pipeline analysis. We further extend this comparison by studying deformation field training over entire long sequences versus shorter GOP segments. The higher fidelity observed in the latter setting is likely due to reduced point-wise discrepancies between the canonical space and dynamic observations. In the case of Grid4D, which relies on hash grids, restricting the temporal span can additionally alleviate the effects of hash collisions (Fig. 6). More generally, different deformation fields introduce different inductive biases, and our results demonstrate that these biases can influence the RD trade-off in certain scene configurations, as shown in Tab. 5.

3.3 Quantization-aware Optimization

We propose quantization-aware optimization during training to account for the effects of attribute quantization at compression time. Positions $\mathbf{x}$ are not quantized during training, since directly converting them from `float32` to `float16` produces nearly identical results during optimization and after compression. For all remaining attributes, we adopt Straight Through Estimation (STE) to enable gradient-based optimization despite the presence of non-differentiable quantization. The procedure is defined as:

$$\begin{aligned} \hat{v} &= \text{round}\left(\tfrac{v - v_{min}}{v_{range}} \times 2^{\text{bit}}\right), \\ \bar{v} &= \hat{v} \div 2^{\text{bit}} \times v_{range} + v_{min}, \\ \tilde{v} &= \text{SG}\left(\bar{v} - v\right) + v, \end{aligned} \tag{3}$$

where v denotes all attributes except for $\mathbf{x}$ and $\mathbf{c}_{\text{rest}}$. Here, $\hat{v}$ represents the quantized values that are ultimately stored, $\bar{v}$ denotes the de-quantized values, and $\tilde{v}$ is used for gradient computation. The stop-gradient operator $\text{SG}(\cdot)$ allows the model to account for quantization error during optimization. This is sufficient in practice, as photometric rendering of overlapping Gaussians does not require high numerical precision.

The same quantization procedure is applied to the entries of the $\mathbf{c}_{\text{rest}}$ codebooks. For all attributes except the scale $\mathbf{s}$, the minimum and maximum values are recomputed dynamically during training, as these attributes are bounded and independent of the global scene scale. We find that the choice of the maximum scale $\mathbf{s}_{\max}$ has a noticeable impact on reconstruction quality, particularly for scenes with distant regions, and therefore determine it empirically.

Let $z \in \mathbb{R}^{M\times 45}$ denote the learnable codebook entries for $\mathbf{c}_{\text{rest}} \in \mathbb{R}^{N\times 45}$ ($M \ll N$), and $\tilde{z}$ their quantized counterparts obtained via Eq. 3. The commitment loss is defined as:

$$\begin{aligned} \text{indices} &= \text{NNS}(\mathbf{c}_{\text{rest}}, \tilde{z}), \\ \bar{\mathbf{c}}_{\text{rest}} &= \tilde{z}[\text{indices}], \\ \tilde{\mathbf{c}}_{\text{rest}} &= \text{SG}(\bar{\mathbf{c}}_{\text{rest}} - \mathbf{c}_{\text{rest}}) + \mathbf{c}_{\text{rest}}, \\ \mathcal{L}_{commitment} &= \text{MSE}(\bar{\mathbf{c}}_{\text{rest}}, \mathbf{c}_{\text{rest}}), \end{aligned} \tag{4}$$

where NNS denotes nearest-neighbor search and MSE denotes mean squared error. This loss encourages the continuous values to stay close to their assigned codebook entries. The overall quantization and compression workflow is summarized in Fig. 4. By jointly optimizing reconstruction and quantization effects, the final objective

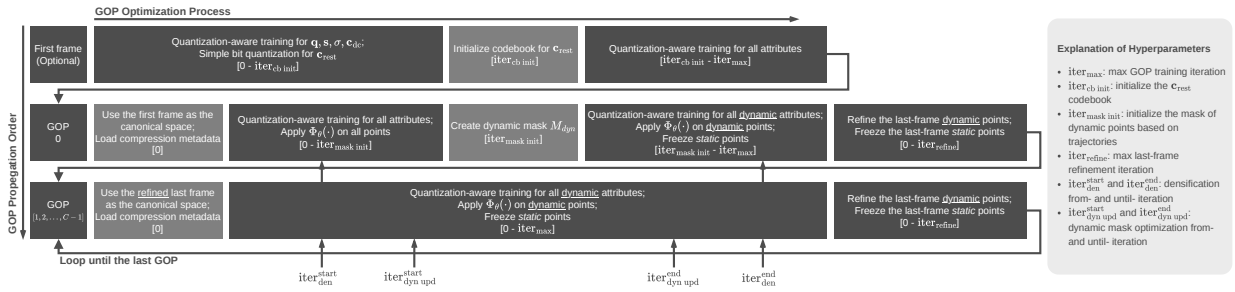


Fig. 3. **Outline of our training schedule.** Events occurring during optimization are highlighted in lighter gray boxes. The dynamic mask is extracted after an initial training phase on the first GOP at iteration $\text{iter}_{\text{mask init}}$, with a detailed breakdown provided in the supplementary material. The corresponding hyperparameter values are reported in Tab. 1.

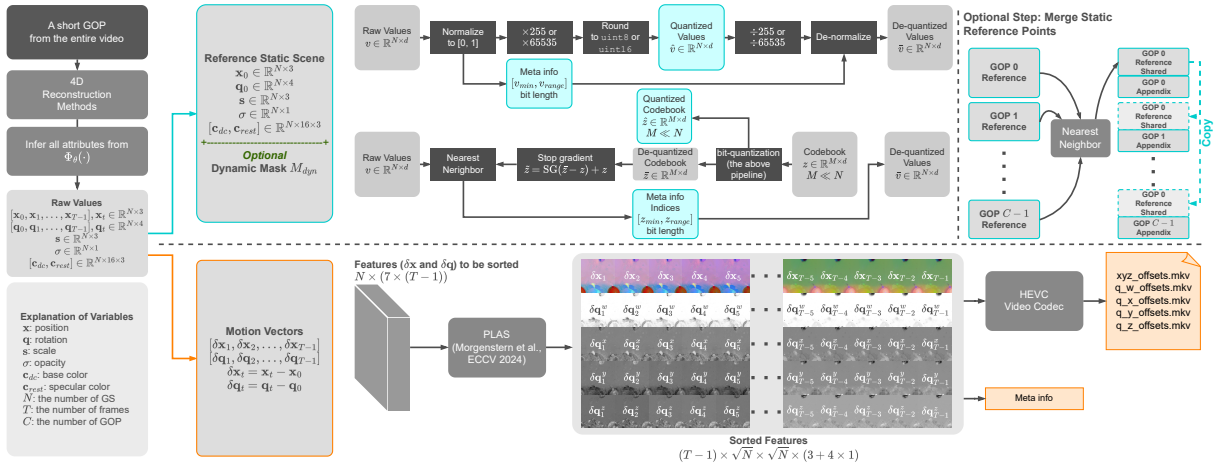


Fig. 4. **The compression pipeline following reconstruction.** Compression is then applied to the explicit attributes rather than to the deformation field itself. Files stored for the reference scene are shown in cyan, while the resulting motion vector files are shown in orange.

is:

$$\mathcal{L} = \mathcal{L}_{recon} + \lambda_c \mathcal{L}_{commitment}, \tag{5}$$

where λ_c controls the weight of the commitment loss. We do not apply context-aware compression or explicit entropy models to the static attributes, primarily to keep the design simple, although more sophisticated entropy modeling could further reduce storage. Nevertheless, with STE and the commitment loss, we achieve performance that is nearly identical to the uncompressed setting, as shown in Tab. 6.

3.4 Compression Pipeline

Once reconstruction is complete, the intermediate deformation field $\Phi_\theta(\cdot)$ is no longer needed. As illustrated in Fig. 4, we retain only the explicit attributes at each time step for compression. The reference positions $\mathbf{x}_0$ are stored in `float16`, while all remaining attributes are quantized into fixed data types together with their associated metadata.

We compress motion by encoding the per-point offsets $\delta\mathbf{x}_t$ and $\delta\mathbf{q}_t$ using standard video codecs. To this end, we first reshape the motion vectors into image-like tensors. Specifically, we concatenate the position offsets in $\mathbb{R}^3$ and rotation offsets in $\mathbb{R}^4$ from all time steps along the channel dimension, forming a tensor of size $N_{dyn} \times D$, where $D = 7 \times (T-1)$ and N_{dyn} denotes the number of dynamic points. For simplicity, we use N to denote this quantity in the remainder of this section.

To exploit spatial redundancy, we reorganize this tensor into a 2D grid, then apply PLAS [Morgenstern et al. 2024] to obtain a structured tensor of size $\sqrt{N} \times \sqrt{N} \times D$. This tensor is further reshaped into a conventional video representation of size $(T-1) \times (\sqrt{N} \times \sqrt{N}) \times 7$. Because standard video codecs operate on either three-channel or single-channel images, we encode the position offsets as RGB frames and distribute the four quaternion components across four sequences of grayscale frames. Prior to video encoding, we apply per-channel min–max normalization to the motion vectors, uniformly quantize each component to 16-bit integers, and store the resulting upper and lower 8-bit values as separate PNG images. When the underlying reconstruction yields motion vectors that are spatially and temporally coherent, as discussed in Fig. 2, keeping only the upper 8 bits of the quantized representation introduces negligible degradation while further reducing storage. To accomplish this, we typically require some form of regularization or a specific architectural design to promote smoothness, which generally comes

Table 1. **Hyperparameters used in our training schedule.** These parameters should be interpreted together with Fig. 3. For first-frame reconstruction, we follow the default densification schedule of 3DGS.

Hyper.	Value	Hyper.	Value	Hyper.	Value
iter_{max}	10000	$\text{iter}_{\text{refine}}$	150, 3000	batch size	2
$\text{iter}_{\text{cb init}}$	7500	$\text{iter}_{\text{mask init}}$	5000	s_{max}	5, 20, 50
$\text{iter}^{\text{start}}_{\text{dyn upd}}$	3000	$\text{iter}^{\text{end}}_{\text{dyn upd}}$	7500	$\text{iter}^{\text{start}}_{\text{den}}$	3000
$\text{iter}^{\text{end}}_{\text{den}}$	7500				

at a cost in terms of runtime overhead and extra hyperparameters, as shown in Tab. 5.

Finally, we apply the HEVC [Sze et al. 2014] (H.265) codec to encode the PNG sequences into videos, fully leveraging inter-frame redundancy. Position offsets are compressed in an almost lossless manner using a Constant Rate Factor (CRF) of 0, whereas quaternion offsets are compressed with a CRF value of 15. Complete codec configurations are provided in the supplementary materials.

3.5 Bit Rate Control

As our training procedure does not use an explicit entropy model, the rate–distortion (RD) trade-off is governed primarily by the densification threshold. Reducing this threshold increases point density and generally improves reconstruction fidelity, at the expense of higher storage cost. The encoding configuration can also affect the RD trade-off through the size of the motion vectors, as discussed in Sec. 4.5.2. Applying more aggressive quantization in the video codec can further reduce their storage footprint, but may come at the cost of reduced fidelity and temporal stability. The combined impact is illustrated in Fig. 8.

4 Experiments

4.1 Hyperparameters

We present the configuration of our training schedule parameters in Tab. 1. For details on the hyperparameters of $\Phi_\theta(\cdot)$ and λ_{reg}, please refer to the supplementary materials. Notably, we observe that multi-batch training can alleviate projection ambiguity, thereby lowering the risk of overshooting and ultimately decreasing the total number of points, which in turn saves storage. This observation has been documented in greater detail in 3DGS^2 [Lan et al. 2025], 4DRotor [Duan et al. 2024] and FastGS [Ren et al. 2025]. We disable the opacity reset step in the original 3DGS and instead employ the opacity reduction from AbsGS [Ye et al. 2024]. Additionally, we apply exposure compensation using an affine transformation.

4.2 Datasets

We evaluate our method on three real-world datasets. Neur3D [Li et al. 2022b] contains six multi-view scenes with 300 frames each, and MeetRoom [Li et al. 2022a] includes three scenes of the same length. In addition, we evaluate our pipeline on the NeRF-DS [Yan et al. 2023] dataset, which features dynamic scenes captured with moving cameras. This setting provides an additional evaluation scenario with varying camera setups, where using 2D optical flow as prior information for separating dynamic points is not easily applicable. For all datasets, we follow the standard train/test camera splits used in prior work and perform evaluation via view interpolation with front-facing camera setups.

Table 2. **Run-to-run variability of photometric metrics for 3DGS.** We report the best and worst runs here, and provide the full table in the supplementary material. All runs use the same random seed.

Run Id	Flame Steak (Neur3D [Li et al. 2022b]) 1st Frame			
	PSNR↑	SSIM↑	LPIPS(VGG)↓	LPIPS(Alex)↓
Best(B)	34.78	95.68	17.22	6.77
Worst(W)	34.23	95.62	17.22	6.69
W→B	34.92	95.78	17.15	6.68

Fig. 5. PSNR discrepancies between different runs in Neur3D [Li et al. 2022b] mainly come from color shifts, as evidenced by the difference before and after applying color correction on Run W. Zoom in for better visualization.

4.3 Training & Evaluation Protocols

As shown in Tab. 2, photometric metrics, particularly PSNR, exhibit noticeable variability across runs even when the random seed is fixed. This behavior is consistent with prior observations [Rota Bulò et al. 2024; Yun et al. 2025] and reflects the sensitivity of these metrics to factors such as color and exposure differences in the test views. We find that a large portion of the observed variance can be attributed to global color shifts. After performing a simple global color alignment on Run W to match Run B (W→B), the run with lower PSNR exhibit comparable visual quality and similarly detailed fine-grained structures, as shown in Fig. 5. Based on this observation, we adopt a transparent evaluation protocol in which we run static 3DGS reconstruction ten times on the first frame and select the instance with the highest PSNR. We then execute the full pipeline five times to assess robustness. In addition, we report worst-case performance on Neur3D (Tab. 3) to reflect the variability inherent in the training process, and note that either best-case or worst-case results may reasonably be reported in future benchmarking.

We further observe that LPIPS scores (indicated as VGG or Alex, depending on the chosen weights) can vary depending on implementation details, such as whether image values are rescaled from $[0, 1]$ to $[-1, 1]$, an issue also discussed in [Rota Bulò et al. 2024]. To improve clarity, we report results using two variants that differ only in this normalization step. The variant that does not apply this rescaling is marked with a superscript asterisk (*).

4.4 Results

In addition to per-frame image metrics, we report the video metrics VMAF-NEG [Bampis et al. 2019] and ColorVideoVDP [Mantiuk et al. 2024]. We also use tLPIPS (×1000) as a proxy metric for temporal

Table 3. **Results on the Neur3D [Li et al. 2022b] dataset.** SSIM and LPIPS values are scaled by 100 for readability. § marks results corresponding to the worst reconstructed first frame. Our variants with full training steps are **averaged** over five runs. High(H) and low(L) refer to relatively high or low bitrates, respectively, which are determined by the densification threshold as explained in Sec. 3.5. We use the half-resolution images without undistortion for training and evaluating both GIFStream's and our variants. FPS is measured on a system with Intel(R) Core(TM) i9-14900K CPUs and an NVIDIA RTX A6000 GPU.

Method	Configs		Neur3D Average (6 scenes; 300 frames; Resolution: 1352 × 1014)								
	T	O	PSNR↑	SSIM↑	VGG/Alex*↓	VMAF-NEG↑	ColorVideoVDP↑	tLPIPS↓	FPS↑	Train↓	**Size↓**
Grid4D	w/o comp. or GOP		31.49	93.60	14.31/5.67	62.16	8.36	3.3	106	155	290.57
SpDeF	w/o comp. or GOP		31.60	94.08	12.80/4.50	69.58	8.49	2.1	140	160	398.87
4DMotionLayer	30	5	32.55	94.88	12.83/N/A	N/A	N/A	N/A	185	73	21.23
GIFStream-H	30	0	31.61	94.01	13.92/5.26	66.94	8.45	4.4	95	141	17.00
GIFStream-L			31.22	93.87	14.33/5.59	65.79	8.40	4.4	95	141	14.47
Ours+Grid4D-H	30	5	32.61	94.25	12.70/4.53	69.56	8.60	3.9	63	360	17.48
Ours+Grid4D-H-1/2			32.65	94.29	12.82/4.63	68.15	8.57	3.3	63	135	16.05
Ours+Grid4D-H-1/4			32.57	94.24	13.02/4.82	67.07	8.55	2.9	63	75	15.70
Ours+SpDeF-H	30	5	32.58	94.26	12.71/4.53	68.65	8.55	2.8	65	420	16.89
Ours+SpDeF-H-§			31.94	94.19	12.76/4.64	68.23	8.53	2.8	65	420	16.94
Ours+SpDeF-H-1/2			32.61	94.25	12.90/4.69	68.01	8.55	2.6	65	150	15.00
Ours+SpDeF-H-1/4			32.37	94.13	13.05/4.82	67.37	8.50	2.4	65	89	14.35
Ours+SpDeF-L	30	5	32.33	94.00	13.57/5.24	65.78	8.53	2.7	76	420	7.98
Ours+SpDeF-L-1/2			32.31	93.96	13.87/5.52	64.95	8.51	2.5	76	120	7.32
Ours+SpDeF-L-1/4			32.16	93.83	14.05/5.65	64.73	8.46	2.3	76	70	7.02
GIFStream-H	60	0	31.55	93.81	14.25/5.65	66.10	8.37	4.1	95	117	9.75
GIFStream-L			31.23	93.52	15.12/6.26	62.77	8.31	4.3	95	117	7.32
Ours+Grid4D-H	60	10	32.55	94.21	12.76/4.59	68.43	8.56	3.7	80	210	12.43
Ours+SpDeF-H			32.53	94.25	12.74/4.55	68.59	8.54	3.1	76	210	12.81
Ours+SpDeF-H-§			31.93	94.19	12.81/4.67	68.22	8.51	3.0	76	210	12.72
Ours+SpDeF-H-1/2			32.48	94.16	13.00/4.78	67.86	8.53	2.8	76	95	12.00
Ours+SpDeF-H-1/4			32.35	94.09	13.19/4.97	67.02	8.49	2.5	76	54	11.72
Ours+SpDeF-L			32.35	94.01	13.59/5.24	66.16	8.53	3.0	76	210	6.26

Table 4. **Results on the MeetRoom [Li et al. 2022a] dataset.** We adopt the camera poses released by Swift4D [Wu et al. 2025] and apply its preprocessing pipeline to obtain undistorted images for training and evaluation of both GIFStream and our variants. The results of our variants trained with the full schedule are **averaged** over five runs. We further note that GIFStream can demonstrate particularly strong visual performance in some cases with fast motions.

Method	Configs		MeetRoom Average (300 frames; Resolution: Discussion: 1222 × 684, Trimming: 1222 × 685, VRHeadset: 1224 × 685)								
	T	O	PSNR↑	SSIM↑	VGG/Alex*↓	VMAF-NEG↑	ColorVideoVDP↑	tLPIPS↓	FPS↑	Train↓	**Size↓**
4DMotionLayer	30	5	32.77	95.93	17.00/N/A	N/A	N/A	N/A	N/A	N/A	21.20
GIFStream-H	30	0	31.85	96.12	16.55/3.42	69.72	8.55	6.3	95	117	12.63
GIFStream-L			31.70	96.13	16.78/3.51	69.71	8.55	6.1	95	117	8.57
Ours+SpDeF-H	30	5	32.29	96.16	15.60/3.21	73.09	8.60	4.3	75	270	13.11
Ours+SpDeF-H-1/2			32.28	96.12	15.78/3.26	72.74	8.61	4.1	75	112	12.30
Ours+SpDeF-H-1/4			32.25	96.10	15.78/3.30	72.65	8.60	4.2	75	64	13.67
Ours+SpDeF-L			32.27	96.21	16.28/3.32	73.86	8.62	4.2	75	244	9.06

stability. Detailed evaluation protocols are provided in the supplementary material. As shown in Tab. 3–5, our pipeline consistently improves performance across a variety of base methods, achieving comparable or better quality than prior baselines while requiring less storage. Moreover, we report the performance of our variants trained for one-half (-1/2) and one-quarter (-1/4) of the full training schedule. We present the corresponding RD curves along with ablation studies in Fig. 8-9. When the reconstruction task is simplified through chunking and training-time regularization, optimization becomes more stable, and the performance gap between different deformation field designs becomes smaller. This suggests that, under such settings, multiple existing formulations of $\Phi_\theta(\cdot)$ can perform

Table 5. **Results on the NeRF-DS [Yan et al. 2023] dataset.** We use the pure-MLP variant of SpDeF, which is well suited to this dataset's limited spatial extent and slow motion. Temporal chunking is omitted as unnecessary. Ours+SpDeF-H is **averaged** over five runs. Comparing Ours+SpDeF and Ours+Grid4D highlights the trade-off between stronger regularization, efficiency, and performance. In Ours+Grid4D, reduced variability in the upper 8 bits of the motion vectors makes quantization errors more visible after decoding; we discuss this further in the supplementary materials.

Method	NeRF-DS Average (7 scenes; 500-800 frames)				
	PSNR↑	SSIM↑	VGG/Alex↓	Time↓	**Size↓**
SpDeF	24.40	85.81	24.47/12.85	120	67.93
Ours+Grid4D w/o compress.	23.55	83.10	26.96/14.85	34	66.27
Ours+SpDeF-H	23.95	84.43	25.64/13.84	78	5.90
Ours+SpDeF-L	23.68	84.27	25.80/14.03	35	2.72
Ours+Grid4D-H	18.70	66.92	42.00/30.81	34	6.40

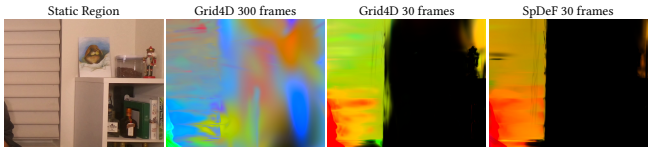


Fig. 6. **Effect of pipeline design on motion vectors.** Compared with the original Grid4D [Xu et al. 2024], training Grid4D over shorter GOPs produces smoother and more consistent motion vectors, especially in static regions (black). By contrast, the original setup, trained over the full 300-frame sequence, results in nearly all points being treated as moving.

well, and that the observed gains are driven primarily by pipeline-level design choices. We illustrate the enhanced motion vectors in Fig. 6. We report rendering frame-per-second (FPS) over the full sequence, measured by sequentially loading the .npz and .mkv files in a Python for-loop; this therefore includes data-loading overhead.

We present visual comparisons in Fig. 10, and provide additional qualitative results and motion vector visualizations in Fig. 11. Both figures highlight visual limitations that are not captured by the averaged quantitative results. Although SpDeF [Song et al. 2025] and Grid4D [Xu et al. 2024] achieve similar quantitative performance under our framework, notable differences in visual behavior can be observed in Fig. 12. In particular, Ours+Grid4D preserves higher per-frame fidelity, while exhibiting minor temporal instability when discarding the lower 8 bits.

4.5 Ablation Study

4.5.1 Bit length and quantization awareness. The impacts of bit-length selection and awareness are presented in Tab. 6.

4.5.2 Encoding Configurations. We compress the motion vectors using H.265 at multiple CRFs, as presented in Fig. 8.

4.5.3 Canonical frame propagation. By propagating the dynamic mask from the beginning to the end, we keep a common static background shared by all GOPs, which reduces file size by storing the static points only once, as shown in the optional step in Fig. 4.

Table 6. **Bit-length ablation for RD trade-offs.** The primary setting uses $\mathbf{x} \in \mathbb{R}^{N\times3}$: `float16`, $\mathbf{q} \in \mathbb{R}^{N\times4}$: `uint8`, $\mathbf{s} \in \mathbb{R}^{N\times3}$: `uint16`, $\sigma \in \mathbb{R}^{N\times1}$: `uint8`, $\mathbf{c}_{\text{dc}} \in \mathbb{R}^{N\times3}$: `uint8`, and $\mathbf{c}_{\text{rest}} \in \mathbb{R}^{N\times45}$: `uint8` with codebook size $M = 1024$. The ablation columns list only the differences from the primary setting. Experiments are conducted on the first frame of the *flame_steak* scene from Neur3D [Li et al. 2022b], reporting the **best** PSNR over five runs. The lower PSNR of w/o Quant. compared with our primary result does not indicate severe degradation, as clarified in Tab. 2. Results achieving a PSNR above 34.7 exhibit nearly identical visual quality.

	Primary	Abl. $\mathbf{s}$	Abl. $\mathbf{c}_{\text{rest}}$	w/o Quant.	w/o Aware
Bit setting	default	$\mathbf{s}$ = `uint8`	$M = 8192$	all `float32`	default
PSNR↑	34.99	33.60	34.72	34.78	33.85
LPIPS(VGG)↓	17.28	20.18	17.12	17.22	18.36
Size (MB)↓	1.8	1.7	1.8	24.24	1.8

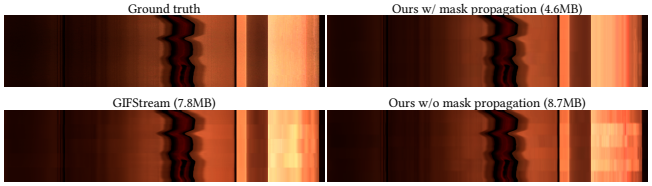


Fig. 7. **Temporal consistency across neighboring GOPs (T=60).** The comparisons represent the trade-off between parallel and sequential execution discussed in Sec. 3.1.

Table 7. **Ablation study on the effect of batch size.** Experiments are conducted on the first frame of the *cook_spinach* scene. Results report the **best** *converged* PSNR among five runs.

Batch Size	Iter	PSNR↑	VGG/Alex↓	**#GS↓**	Time↓
1	10,000	32.56	18.07/ 7.09	300K	11min
1	20,000	31.62	18.17/ 6.97	340K	25min
2	10,000	33.35	18.87/ 7.55	105K	17min

More importantly, This approach also yields smoother transitions between neighboring GOPs, as shown in Fig. 7.

4.5.4 Batch size. As shown in Tab. 7, increasing the batch size leads to noticeable differences in the number of reconstructed points. This observation suggests that seemingly low-level training choices can have a meaningful impact on downstream performance.

5 Limitation & Conclusion

While temporal chunking alleviates some optimization challenges, deformation-based methods still have inherent limitations. Training is less efficient than in short-lifespan methods, per-Gaussian temporal attribute methods, and approaches guided by 2D priors such as optical flow or segmentation, particularly when multi-batch training and additional regularization are required. Performance also remains sensitive to data quality (e.g., image distortion) and object motion speed, while hyperparameter tuning is still needed due to scene-scale variations across datasets. Limitations and failure cases are shown in Fig. 11-13. A promising direction is multi-pass

encoding and adaptive GOP length, which allocates more training and storage resources to complex motions through shorter GOPs.

At its core, this work is motivated by the idea that promoting spatially and temporally smooth content can make dynamic scene representations more compressible. Our method constitutes one concrete instantiation of this principle, realized through a carefully designed pipeline and strong off-the-shelf reconstruction techniques. More broadly, we hope this perspective will inspire future work to explore alternative mechanisms for enforcing such smoothness, or even new paradigms that transcend these heuristics altogether, thereby further advancing dynamic scene reconstruction and compression.

Acknowledgments

We gratefully acknowledge the creators of the open-source datasets used in this work. The *hellwarrior* scene in Fig. 1 was created by Albert Pumarola et al. [Pumarola et al. 2021]. The *flame_steak*, *sear_steak*, *cut_roasted_beef* and *cook_spinach* scenes in Fig. 5, 6, 10, 11, 12, and 13 were created by Zhaoyang Lv et al. [Li et al. 2022b]. The *VRHeadset* scene in Fig. 11 and 13 was captured by Li et al. [Li et al. 2022a]. We thank Wu et al. [Wu et al. 2025] for releasing the MeetRoom [Li et al. 2022a] camera parameters, which we use in Tab. 4.

References

Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. 2020. ꟻLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3, 2 (2020), 15:1–15:23. doi:10.1145/3406183

Christos G. Bampis, Zhi Li, and Alan C. Bovik. 2019. Spatiotemporal Feature Integration and Model Fusion for Full Reference Video Quality Assessment. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 8 (2019), 2256–2270. doi:10.1109/TCSVT.2018.2868262

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

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. 2024. HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression. In *European Conference on Computer Vision*.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. 2025. HAC++: Towards 100X Compression of 3D Gaussian Splatting. (2025).

Pinxuan Dai, Peiquan Zhang, Zheng Dong, Ke Xu, Yifan Peng, Dandan Ding, Yujun Shen, Yin Yang, Xinguo Liu, Rynson W. H. Lau, and Weiwei Xu. 2025. 4D Gaussian Videos with Motion Layering. *ACM Trans. Graph.* 44, 4, Article 124 (July 2025), 14 pages. doi:10.1145/3731189

Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 2024. 4D-Rotor Gaussian Splatting: Towards Efficient Novel-View Synthesis for Dynamic Scenes. In *Proc. SIGGRAPH*.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. 2024. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems* 37 (2024), 140138–140158.

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

Quankai Gao, Qiangeng Xu, Zhe Cao, Ben Mildenhall, Wenchao Ma, Le Chen, Danhang Tang, and Ulrich Neumann. 2024. Gaussianflow: Splatting gaussian dynamics for 4d content creation. *arXiv preprint arXiv:2403.12365* (2024).

Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. 2023. SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes. *arXiv preprint arXiv:2312.14937* (2023).

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics* 42, 4 (2023), 1–14.

Martin Kilian, Niloy J Mitra, and Helmut Pottmann. 2007. Geometric modeling in shape space. In *ACM SIGGRAPH 2007 papers*. 64–es.

Julian Knodt. 2022. Continuous Dynamic-NeRF: Spline-NeRF. *arXiv preprint arXiv:2203.13800* (2022).

Lei Lan, Tianjia Shao, Zixuan Lu, Yu Zhang, Chenfanfu Jiang, and Yin Yang. 2025. 3dgs2: Near second-order converging 3d gaussian splatting. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*. 1–10.

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

Hao Li, Sicheng Li, Xiang Gao, Abudouaihati Batuer, Lu Yu, and Yiyi Liao. 2025. GIFStream: 4D Gaussian-based Immersive Video with Feature Stream. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 21761–21770.

Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. 2022a. Streaming radiance fields for 3d video synthesis. *Advances in Neural Information Processing Systems* 35 (2022), 13485–13498.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. 2022b. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5521–5531.

Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. 2024. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8508–8520.

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

Rafal K. Mantiuk, Param Hanji, Maliha Ashraf, Yuta Asano, and Alexandre Chapiro. 2024. ColorVideoVDP: A visual difference predictor for image, video and display distortions. *ACM Trans. Graph.* 43, 4, Article 129 (July 2024), 20 pages. doi:10.1145/3658144

D. Marpe, T. Wiegand, and G.J. Sullivan. 2006. The H.264/MPEG4 advanced video coding standard and its applications. *IEEE Communications Magazine* 44, 8 (2006), 134–143. doi:10.1109/MCOM.2006.1678121

Marko Mihajlovic, Sergey Prokudin, Marc Pollefeys, and Siyu Tang. 2024. ResFields: Residual Neural Fields for Spatiotemporal Signals. In *International Conference on Learning Representations (ICLR)*.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. 2024. Compact 3d scene representation via self-organizing gaussian grids. In *European Conference on Computer Vision*. Springer, 18–34.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.

Sara Oblak, Despoina Paschalidou, Sanja Fidler, and Matan Atzmon. 2024. ReMatching Dynamic Reconstruction Flow. *arXiv preprint arXiv:2411.00705* (2024).

Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. 2024. Reducing the Memory Footprint of 3D Gaussian Splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 7, 1 (May 2024). https://repo-sam.inria.fr/fungraph/reduced_3dgs/

Jongmin Park, Minh-Quan Viet Bui, Juan Luis Gonzalez Bello, Jaeho Moon, Jihyong Oh, and Munchurl Kim. 2025. SplineGS: Robust Motion-Adaptive Spline for Real-Time Dynamic 3D Gaussians from Monocular Video. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*. 26866–26875.

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. 2021a. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5865–5874.

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

Sergey Prokudin, Qianli Ma, Maxime Raafat, Julien Valentin, and Siyu Tang. 2023. Dynamic Point Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 7964–7976.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10318–10327.

Shiwei Ren, Tianci Wen, Yongchun Fang, and Biao Lu. 2025. FastGS: Training 3D Gaussian Splatting in 100 Seconds. *arXiv preprint arXiv:2511.04283* (2025).

Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. 2024. Revising densification in gaussian splatting. In *European Conference on Computer Vision*. Springer, 347–362.

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. 2023. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic

reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16632–16642.

Mingyang Song, Yang Zhang, Marko Mihajlovic, Siyu Tang, Markus Gross, and Tunç Ozan Aydın. 2025. Spline Deformation Field. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 74, 10 pages. doi:10.1145/3721238.3730676

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, Vol. 4. Citeseer, 109–116.

Vivienne Sze, Madhukar Budagavi, and Gary J. Sullivan. 2014. *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Springer Publishing Company, Incorporated.

Penghao Wang, Zhirui Zhang, Liao Wang, Kaixin Yao, Siyuan Xie, Jingyi Yu, Minye Wu, and Lan Xu. 2024. V^3: Viewing Volumetric Videos on Mobiles via Streamable 2D Dynamic Gaussians. *ACM Trans. Graph.* 43, 6, Article 187 (Nov. 2024), 13 pages. doi:10.1145/3687935

Rui Wang, Quentin Lohmeyer, Mirko Meboldt, and Siyu Tang. 2025. Degauss: Dynamic-static decomposition with gaussian splatting for distractor-free 3d reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6294–6303.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2023. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528* (2023).

Jiahao Wu, Rui Peng, Zhiyan Wang, Lu Xiao, Luyang Tang, Jinbo Yan, Kaiqiang Xiong, and Ronggang Wang. 2025. Swift4D: Adaptive divide-and-conquer Gaussian Splatting for compact and efficient reconstruction of dynamic scene. *arXiv preprint arXiv:2503.12307* (2025).

Shuzhao Xie, Weixiang Zhang, Chen Tang, Yunpeng Bai, Rongwei Lu, Shijia Ge, and Zhi Wang. 2024. MesonGS: Post-training Compression of 3D Gaussians via Efficient Attribute Transformation. In *European Conference on Computer Vision*. Springer.

Jiawei Xu, Zexin Fan, Jian Yang, and Jin Xie. 2024. Grid4D: 4D Decomposed Hash Encoding for High-fidelity Dynamic Gaussian Splatting. *arXiv preprint arXiv:2410.20815* (2024).

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

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. 2023. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101* (2023).

Zeyu Yang, Zijie Pan, Xiatian Zhu, Li Zhang, Jianfeng Feng, Yu-Gang Jiang, and Philip HS Torr. 2024a. 4D Gaussian Splatting: Modeling Dynamic Scenes with Native 4D Primitives. *arXiv preprint* (2024).

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. 2024b. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. In *International Conference on Learning Representations (ICLR)*.

Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. 2024. AbsGS: Recovering Fine Details in 3D Gaussian Splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia* (Melbourne VIC, Australia) *(MM '24)*. Association for Computing Machinery, New York, NY, USA, 1053–1061. doi:10.1145/3664647.3681361

Jihwan Yoon, Sangbeom Han, Jaeseok Oh, and Minsik Lee. 2025. SplineGS: Learning Smooth Trajectories in Gaussian Splatting for Dynamic Scene Reconstruction. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=tMG6btjBfd

Youngsik Yun, Jeongmin Bae, Hyunseung Son, Seoha Kim, Hahyun Lee, Gun Bang, and Youngjung Uh. 2025. Compensating Spatiotemporally Inconsistent Observations for Online Dynamic 3D Gaussian Splatting. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 139, 9 pages. doi:10.1145/3721238.3730678

Yu-Ting Zhan, Cheng-Yuan Ho, Hebi Yang, Yi-Hsin Chen, Jui Chiu Chiang, Yu-Lun Liu, and Wen-Hsiao Peng. 2025. CAT-3DGS: A context-adaptive triplane approach to rate-distortion-optimized 3DGS compression. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*.

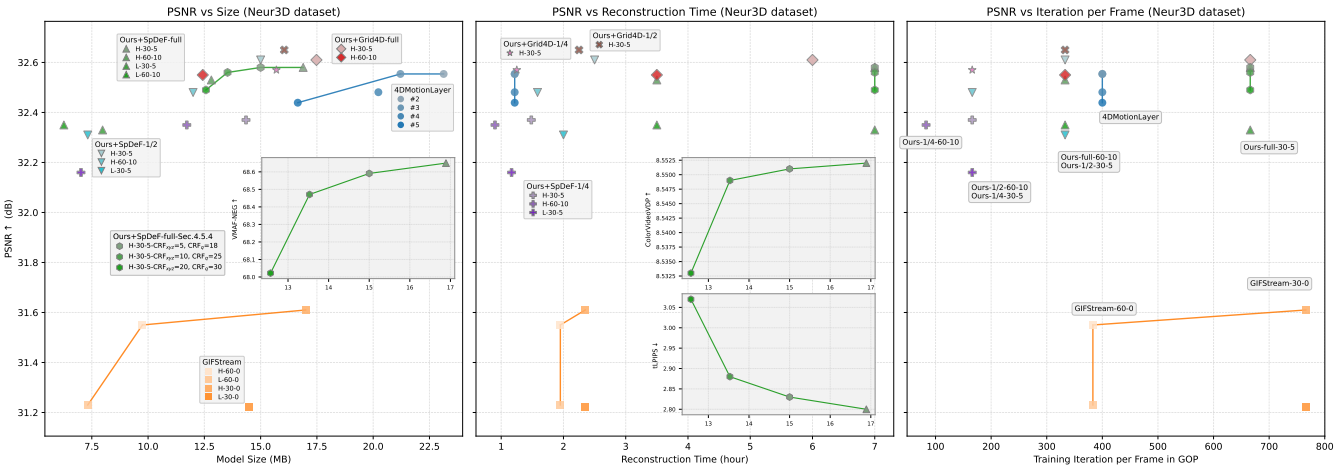


Fig. 8. **Rate–distortion–reconstruction–time trade-off.** The legend entries follow the format: **[Method]**-[(Optional) Training Budget (-full,-1/2,-1/4)]-**[Rate (H,L)]**-**[GOP Length T]**-**[Overlap Frames O]**. This figure should be read in conjunction with Tab. 3. The results reported for 4DMotioLayer [Dai et al. 2025] are taken from Tab. 10 of their official manuscript, where #3 corresponds to their default configuration. Unless otherwise noted for the ablation study in Sec. 4.5.2, our motion-vector encoding settings are identical and follow Sec. 3.4, namely $\mathrm{CRF}_{\mathrm{xyz}} = 0$ and $\mathrm{CRF}_{\mathrm{q}} = 15$. As expected, applying mild quantization ($\mathrm{CRF}_{\mathrm{xyz}} = 5$ and $\mathrm{CRF}_{\mathrm{q}} = 18$) to motion vectors that are already sufficiently smooth can further reduce the data size with only minor quality loss. The performance of GIFStream [Li et al. 2025] can be enhanced by adopting a shorter GOP length, which strikes a better balance among per-frame quality, file size, and computational efficiency, while its temporal stability requires further improvement. We account for the impact of our multi-batch training in the iterations-per-frame computation, defined as $2 \times$ total iterations/GOP length. For instance, for Ours+SpDeF-1/2-H-60-10, the value displayed in the third subplot is computed as $2 \times 10{,}000 \times (1/2)/60 \approx 166$.

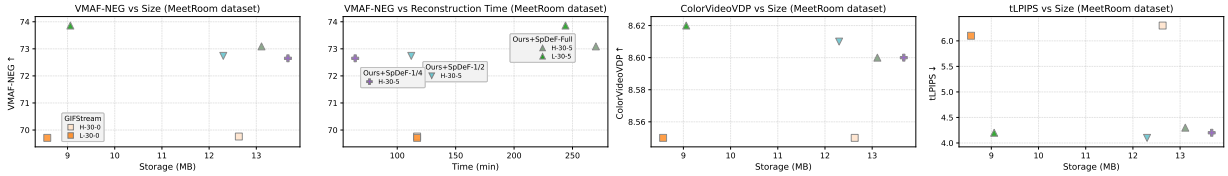


Fig. 9. **Additional trade-offs on video metrics.** This figure should be read in conjunction with Tab. 4. We observe that although GIFStream trails in average performance metrics, its looser constraints allow for more accurate reconstruction of rapidly moving objects.

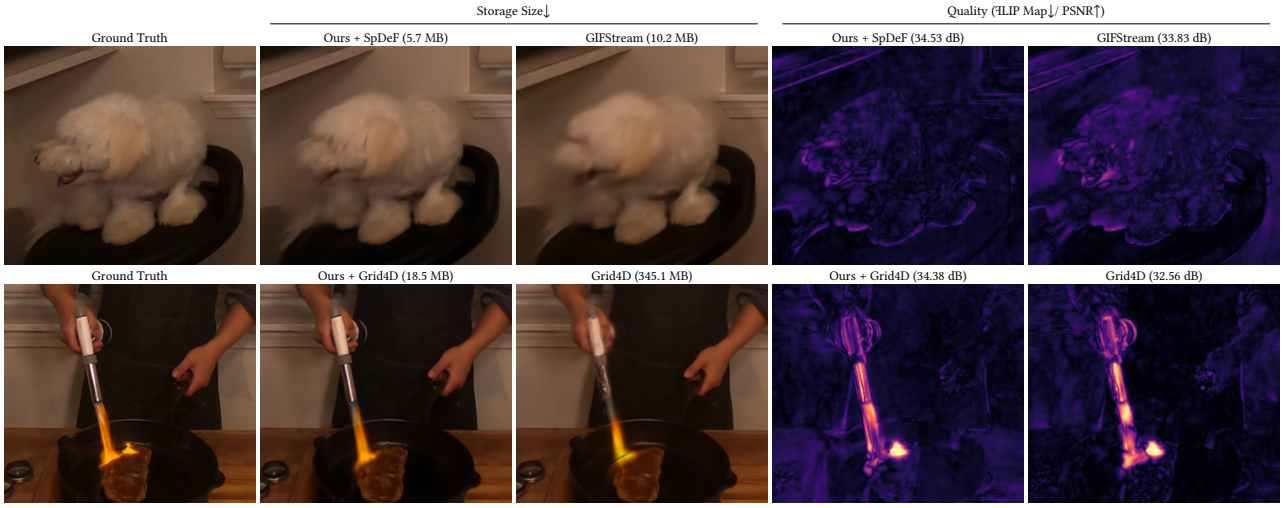


Fig. 10. **Visual comparison.** We present rendered RGB frames and their corresponding error maps computed using ꟻLIP [Andersson et al. 2020]. In some cases, the base deformation method over-smooths high-frequency details, as visible in the flame.

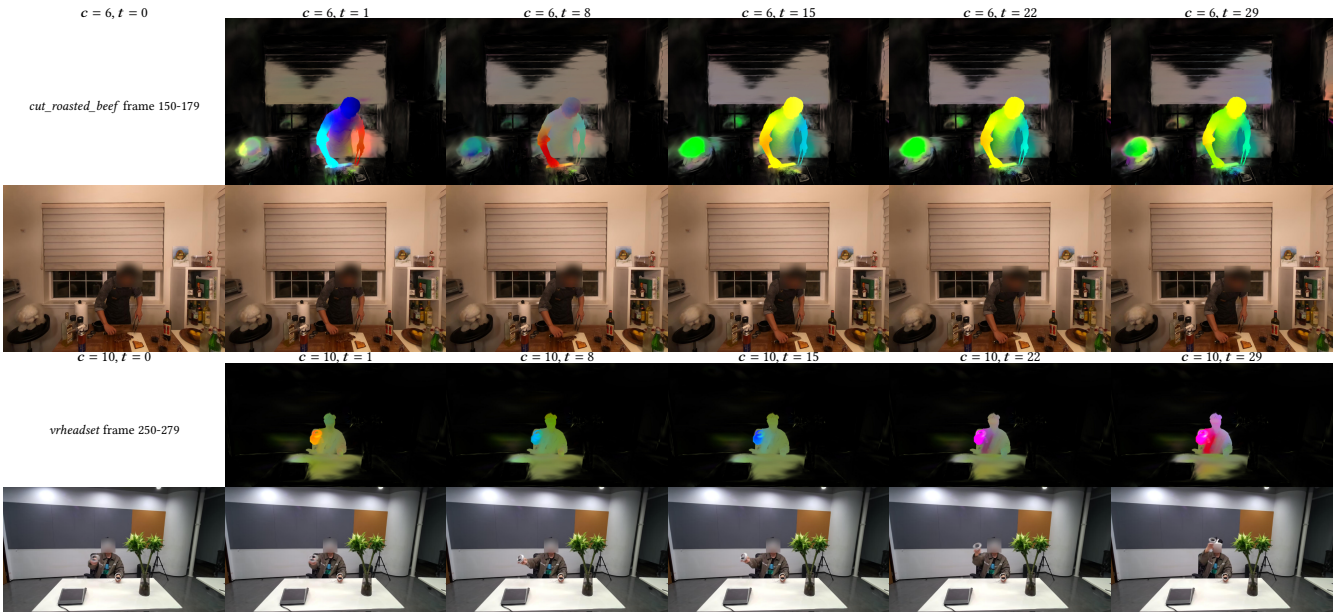


Fig. 11. We present the rendered reference frame, color-encoded motion vectors, and the corresponding rendered dynamic frames. In *cut_roasted_beef*, our method captures non-static points in the window reflections. In *vrheadset*, some points, such as the VR controller handles, are not fully identified as dynamic, revealing a limitation of the current mask update mechanism.

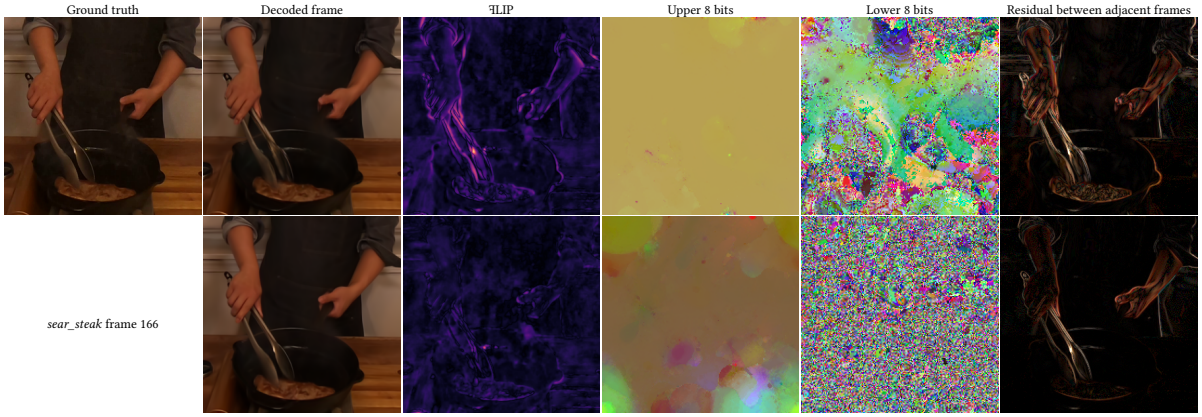


Fig. 12. **Visualization of sorted motion vectors and their value distributions.** The first row shows results from Ours+Grid4D, and the second row shows Ours+SpDeF. In some cases, the vector distribution may deviate from the smoothness assumption, revealing a minor limitation of the current method. We visualize the ground truth, reconstruction, error map (ꟻLIP), upper 8 bits, lower 8 bits, and residuals between adjacent frames.

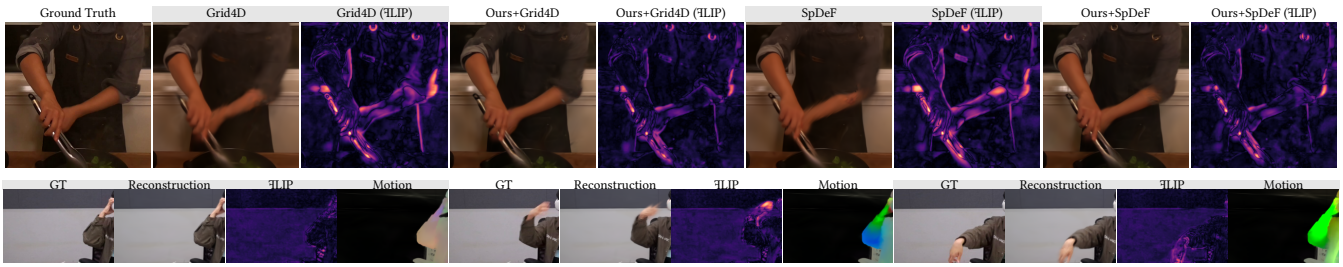


Fig. 13. **Representative failure cases.** Row 1: Rapid motion leads to degraded reconstruction quality on *cook_spinach* frame 109. Row 2: Motion blur cannot be adequately explained by the deformation model and leads to visible artifacts. We provide additional failure examples in the supplementary material.