PackUV: Packed Gaussian UV Maps for 4D Volumetric Video

Aashish Rai[1] Angela Xing[1] Anushka Agarwal[2] Xiaoyan Cong[1] Zekun Li[1]
Tao Lu[1] Aayush Prakash[3] Srinath Sridhar[2]

[1]Brown University [2]UMass Amherst [3]Meta Reality Labs

https://ivl.cs.brown.edu/packuv

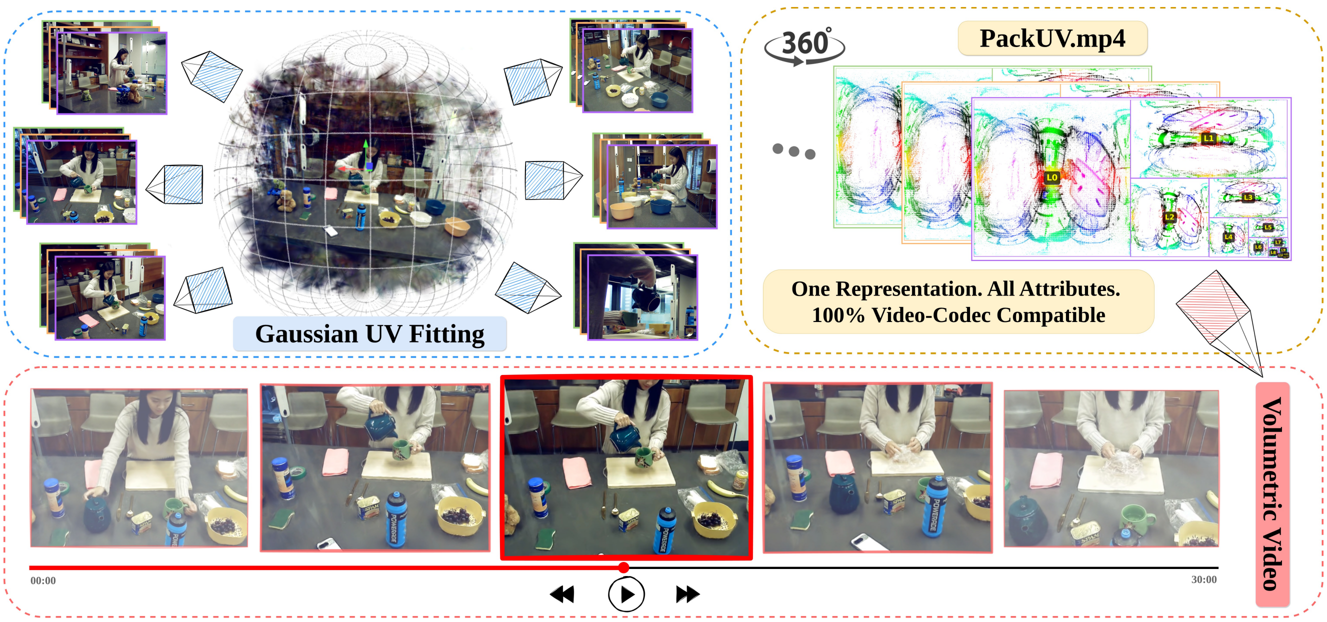

Figure 1. *We propose a novel and compact 4D representation,* ***PackUV****, for volumetric videos that packs 3D Gaussian attributes into a sequence of 2D UV atlases (yellow, top right). PackUV is readily compatible with existing video coding infrastructure* (e.g., *can be coded with HEVC, FFV1). We also propose* ***PackUV-GS****, a method to directly fit Gaussian attributes from multi-view RGB videos into structured PackUV (blue, top left) via optical flow-guided keyframing and Gaussian labeling to fit arbitrary length sequences with temporal consistency even in the presence of large motions and disocclusions. The fitted scene can be rendered back to streamable volumetric video from any viewpoint (red, bottom). We also propose PackUV-2B, the largest 4D multi-view dataset containing* ***2B*** *frames captured with over 50 synchronized cameras to provide 360° coverage.*

Abstract

Volumetric videos offer immersive 4D experiences, but remain difficult to reconstruct, store, and stream at scale. Existing Gaussian Splatting based methods achieve high-quality reconstruction but break down on long sequences, temporal inconsistency, and fail under large motions and disocclusions. Moreover, their outputs are typically incompatible with conventional video coding pipelines, preventing practical applications. We introduce PackUV, a novel 4D Gaussian representation that maps all Gaussian attributes into a sequence of structured, multi-scale UV atlases, enabling compact, image-native storage. To fit this representation from multi-view videos, we propose PackUV-GS, a temporally consistent fitting method that directly optimizes Gaussian parameters in the UV domain. A flow-guided Gaussian labeling and video keyframing module identifies dynamic Gaussians, stabilizes static regions, and preserves temporal coherence even under large motions and disocclusions. The resulting UV atlas format is the first unified volumetric video representation compatible with standard video codecs (e.g., FFV1) without quality loss, enabling efficient streaming within existing multimedia infrastructure. To evaluate long-duration volumetric capture, we present PackUV-2B, the largest multi-view 4D dataset to date, featuring more than 50 synchronized 360 cameras, substantial motion, and frequent disocclusions across 100 sequences and 2B (billion) frames. Extensive experiments

demonstrate that our method surpasses existing baselines in rendering fidelity while scaling to sequences up to 30 minutes with consistent quality.

1. Introduction

Volumetric videos are a form of immersive media that capture scenes in three dimensions and across time (4D), making them viewable from any perspective. They promise numerous applications in AR/VR, entertainment, sports, as well as in applications requiring 4D understanding, for instance, in robotics [19, 22, 64, 93]. Unsurprisingly, creating volumetric videos from multiple camera views has been a long-standing challenge in computer vision and graphics [4, 13, 48, 65, 72].

Common approaches for volumetric video reconstruction rely on explicit representations, such as point clouds [26, 62, 69] meshes [55], multi-plane images [12, 67, 73, 77], or multi-sphere images [2, 6]. However, these methods are limited in their ability to render complex scenes and are highly memory intensive. Meanwhile, radiance fields [49, 83] and 3D Gaussian Splatting [31] have emerged as leading representations for 3D reconstruction and volumetric video [11, 35, 68, 80, 91, 92]. Despite their success, these methods [11, 40, 80, 85, 91, 92] struggle to operate on videos longer than a few seconds. *Streaming* approaches [35, 45, 68, 86] address this via online fitting but struggle to retain **long-duration temporal consistency**, capture **large motions**, and handle **disocclusions** (*e.g.*, when a new object enters the scene). Furthermore, volumetric videos produced by these methods cannot be seamlessly shared due to their large size and required bespoke compression methods [14] that are incompatible with existing multimedia infrastructure.

To address these challenges, we introduce **PackUV**, a novel 4D Gaussian representation that packs 3D Gaussian attributes into a sequence of structured, multi-layered 2D UV maps. These UV map layers are further compacted into a single progressive atlas (Figure 1), enabling efficient storage. We also present **PackUV-GS**, a fitting method that generates temporally-consistent volumetric videos from multi-view input. Unlike previous 3DGS-to-2D approaches that rely on lossy post-hoc UV unwrapping [61, 81], PackUV-GS directly fits Gaussian parameters into the UV domain. To maintain temporal coherence under large motions and disocclusions, an optical-flow–guided module identifies dynamic Gaussians, enforces flow-based keyframing, and selectively freezes gradients in static regions to ensure stable optimization. This design supports high-quality reconstruction for sequences of arbitrary duration while maintaining quality. The resulting image-native representation replaces point-centric storage with a sequence of ordered, multi-scale 2D UV atlases, enabling efficient streaming and storage via standard video coding methods (*e.g.*, HEVC, FFV1). To the best of our knowledge, this is the *first unified representation that applies conventional video coding directly to all 3DGS attributes*, with no quality loss, to bridge 4D Gaussian representations and the existing video infrastructure.

We evaluate the performance of our method and justify design choices on a variety of existing datasets. However, existing datasets [29, 39, 59, 87, 89] are largely restricted to frontal cameras and exhibit limited motions and disocclusions. To better showcase the abilities of our representation and compare it with existing work, we captured the largest long-duration multi-view video dataset, **PackUV-2B**. PackUV-2B features real-world dynamic scenes with more than 50 synchronized cameras, providing 360° coverage in both controlled studio and uncontrolled in-the-wild settings. In total, PackUV-2B contains 100 sequences with more than **2B (billion) frames** featuring a diverse range of scenarios including human-human, human-object, and human-robot interactions. Extensive experiments show that our method outperforms all baselines across standard metrics and can model much longer sequences (up to 30 minutes) while maintaining consistent quality. To summarize, our contributions include:

- **PackUV**, a new volumetric video representation that packs 3D Gaussian attributes into a sequence of UV atlases for efficient streaming and storage, making it readily compatible with existing video coding infrastructure.
- **PackUV-GS**, an efficient method to fit PackUV directly from multiview videos using optical-flow-based keyframing and Gaussian labeling to handle large motions, disocclusions, and temporal consistency.
- **PackUV-2B**, the largest multi-view 4D dataset with 2B frames, large motions, and disocclusions. It provides 360° coverage from 50+ synchronized cameras.

2. Related Work

In this brief related work, we focus on methods and representations for reconstructing volumetric videos.

4D Volumetric Video. Volumetric video fitting is a long-standing problem, with works originally focusing on using multi-view images [7, 9, 58, 96], multi-plane images [12, 67, 73, 77], light fields [15, 25, 30, 36] as well as explicit representations like point clouds [26, 62, 69] meshes [55], voxels [16, 17, 52, 54], or multi-sphere images [2, 6]. More recently, radiance fields [5, 10, 49], in particular 3D Gaussian splatting [31], have emerged as the de facto method for static novel view synthesis and reconstruction. Building on 3DGS, a significant amount of work on dynamic scenes has subsequently emerged [3, 8, 11, 18, 23, 24, 27, 28, 32, 33, 38, 39, 41, 42, 44, 57, 60, 63, 66, 74–76, 80, 82, 85, 88, 90, 91]. Deformable3DGS [91], 4DGS [80], and Grid4D [84] define a deformation field mapping canonical Gaussian primitives to specific time steps. RealTime4DGS [90] and 4D-Rotor-GS [18] introduce 4D Gaussian primitives, improving flexibility for a variety of dynamic scenes. Despite the progress, these methods are limited to short sequences (few seconds), memory intensive, temporally inconsistent, or cannot handle disocclusions (see Section 6).

Recent methods like LongVolCap [87] make tremendous progress by leveraging a hierarchical temporal 4D Gaussian representation to compactly model long-horizon scenes, but still struggle to fit arbitrary durations due to the growth of Gaussians. To address the memory cost of offline 3DGS fitting, 3DGStream [68] and ATGS [11] propose online training and model dynamics of 3D Gaussians per-frame with a neural transformation cache. However, the per-frame storage cost of 3D Gaussians remains large, making it infeasible to transmit and play in a streaming manner like conventional videos. Ex4DGS [34] addresses memory overhead by separating Gaussians into linearly moving 'static' and fully dynamic Gaussians. In addition to static-dynamic Gaussian separation, GIFstream [37] and Motion Layering [14] improve dynamic modeling through time-dependent feature streams. However, these methods still struggle to model large motions and disocclusions.

Volumetric Video Representations. Representing and storing volumetric videos is significantly more expensive than 2D images or videos due to the additional spatial and temporal dimensions. Several approaches have been proposed, either by pruning unused Gaussians, cleverly compressing Gaussian attributes, or using learning [1, 20, 21, 35, 43, 51, 53, 56]. Recently, representations for 'flattening' 3D Gaussians into 2D form have been receiving attention, for instance, SOG [50] and UV projection [61, 81]. However, all these methods are limited to static 3D scenes. For 4D volumetric videos a naive sequence of static representations would still be too large.

Recognizing this, some recent works [14, 78] transfer ideas from 2D video coding to volumetric videos. However, due to the unstructured nature of regular 3DGS attributes, these methods have relatively large compression losses, or heavy computational requirements. Structured UV projection methods [61, 81] are promising since they can then be combined with existing image or video coding methods. However, projecting an already-optimized 3DGS into a UV map is lossy and computationally redundant.

Our PackUV representation, together with our native PackUV-GS fitting method, overcomes these limitations by directly producing a sequence of UV atlases that are fully compatible with existing video coding methods (*e.g.*, HEVC, AVC, FFV1) while being lossless. In addition, our method can handle arbitrary length sequences while preserving temporal consistency under large motions and disocclusions.

3. Preliminaries

We first provide a brief background on 3D Gaussian Splatting [31] and UVGS [61] as our method builds upon them.

3D Gaussian Splatting ((3DGS)) [31]: 3DGS is a representation of 3D shape and appearance consisting of a set of Gaussian primitives with a position $\mu \in \mathbb{R}^3$, and covariance Σ. Additionally, each 3D Gaussian primitive explicitly encodes view-dependent appearance via spherical harmonics (SH) coefficients c and an opacity value $o \in \mathbb{R}$. These attributes are optimized by minimizing the loss between the rendered and reference images. To render a viewpoint, the Gaussians are projected as 2D splats and combined with α-blending using a tile-based rasterizer.

UV-based Point-to-Image Transformation. The original 3DGS representation consists of an unstructured set of permutation-invariant primitives. This unstructured nature poses challenges for downstream tasks, particularly when dealing with thousands or millions of Gaussians.

UVGS [61] addresses this by introducing a structured UV-based point-to-image transformation that reformulates 3D Gaussians from an unordered set into a spatially organized UV image via spherical projection. Each Gaussian g_i centered at $\mu_i = (x_i, y_i, z_i)$ is transformed into spherical coordinates $(\rho_i, \theta_i, \phi_i)$, where $\rho_i = \sqrt{x_i^2 + y_i^2 + z_i^2}, \theta_i = \tan^{-1}(y_i, x_i)$, and $\phi_i = \cos^{-1}\left(\frac{z_i}{\rho_i}\right)$. The azimuthal angle θ_i and polar angle ϕ_i are normalized to discrete UV coordinates in a map of size $M \times N$:

$$u_i = \left\lfloor \frac{\pi + \theta_i}{2\pi} \times M \right\rfloor, \quad v_i = \left\lfloor \frac{\phi_i}{\pi} \times N \right\rfloor. \tag{1}$$

Since multiple Gaussians may project to the same UV coordinate, UVGS uses multiple layers (K layers) to store top primitives, ordering them by opacity o within each pixel. The final UV mapping is $f : \text{UVCoords} \times \text{LayerIdx} \to \mathbb{R}^D$, where D encompasses all Gaussian attributes: $f(u, v, k) = \{\rho,\ r,\ s,\ o,\ c\} \in \mathbb{R}^D$.

The transformed UVGS representation introduces spatial coherence and resolves the permutation invariance problem. Interestingly, due to opacity-based sorting during the Gaussian mapping process, UVGS can effectively recover the surface-level Gaussians. *The capability to represent a set of 3D Gaussian primitives in 2D while preserving surface-level details is of particular importance to our work.*

However, it should be noted that this post-optimization UVGS mapping is highly lossy, since it projects only the Gaussian centers (mean positions) onto the UV space. As a result, it performs well for simple 3D objects but applying this post-hoc transformation to pretrained 4D Gaussian sequences often degrades visual quality, causing missing details and temporal inconsistencies (see supplementary).

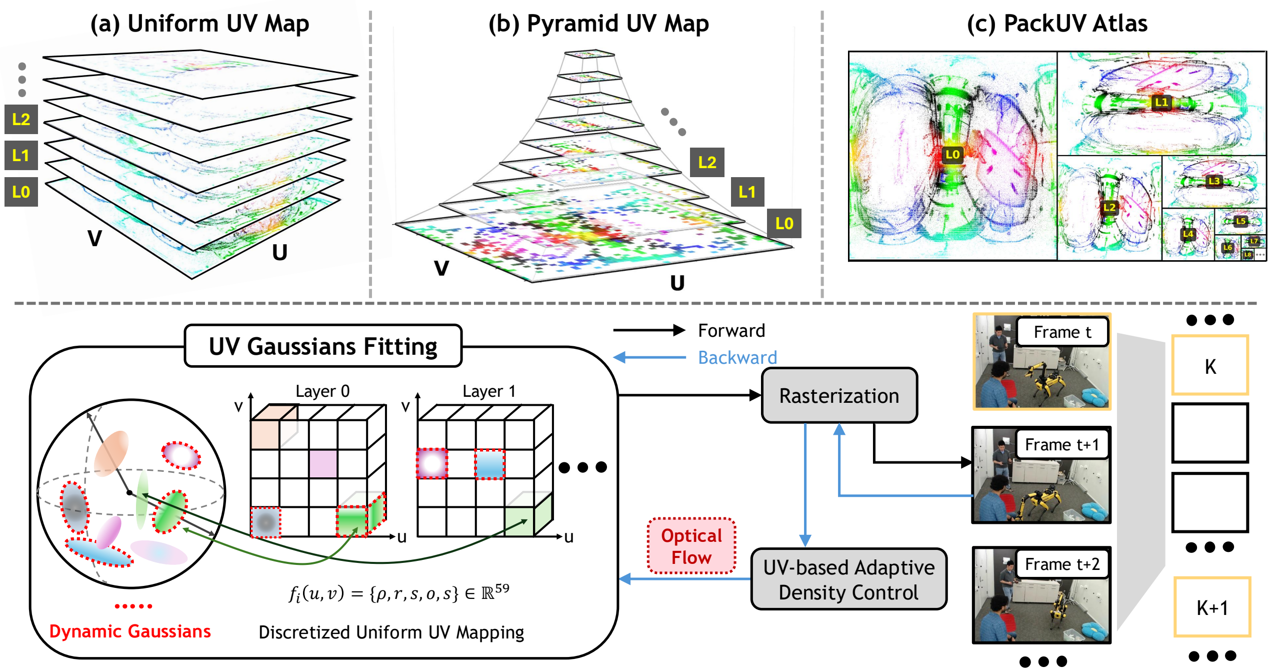

Figure 2. (Top) Three UV-map organization strategies: (a) naïvely stacking UV layers (deep layers become more and more sparse); (b) a geometric-progression UV pyramid (more uniform sparsity with less storage); (c) PackUV, which packs all pyramid layers into a single UV atlas for efficient, codec-friendly processing. (Bottom) We propose PackUV-GS, a new representation based on 3DGS with a discrete spatial distribution constraint via UV fitting. It uses multiple-layer UV images to store the Gaussian attributes during 3DGS fitting. To constrain the 3D Gaussians located on the discrete rays, we propose a UV-based Adaptive Density Control. We also use a stream-based training schema based on keyframes (image with yellow border).

4. Method

Our goal is to retain the structural benefits of UV mapping, preserve 3DGS's strong reconstruction quality, and capture 4D dynamic scenes. To achieve this, we first propose PackUV– a novel representation that combines a UV mapping strategy with progressive downscaling to represent 4D volumetric video. Second, we propose a novel fitting method, PackUV-GS (Section 4.2) that uses optical flow based keyframing and Gaussian labeling to ensure smooth and lossless temporally-consistent reconstruction over long-horizon videos. Our approach ensures efficient representation, accurate reconstruction, and scalability to arbitrary length in complex dynamic environments.

4.1. PackUV Atlas

Pyramid UV Mapping: While direct UV optimization produces structured Gaussian maps, storing all K layers at uniform resolution $M \times N$ incurs significant memory overhead—particularly problematic for high-resolution dynamic sequences. However, we made an important observation: after sorting by opacity, **deeper layers (higher K) contain progressively fewer visible Gaussians** due to occlusion and opacity-based sorting across datasets (more details in the supplementary). This means that not all layers are equally important, and we can therefore adopt a *progressive, pyramid-like representation* (see Figure 2, top). Instead of storing all K layers at base resolution $M_0 \times N_0$, we apply geometric downsampling that alternates dimension reduction:

$$(M_k, N_k) = \begin{cases} (M_0, N_0), & k = 0 \\ (M_{k-1}, N_{k-1}/2), & k \text{ odd} \\ (M_{k-1}/2, N_{k-1}), & k \text{ even} \end{cases}.$$

As shown in Figure 2 (b), this pattern, $\{M_0 \times N_0,\ M_0 \times N_0/2,\ M_0/2 \times N_0/2,\ M_0/2 \times N_0/4, \dots\}$, reflects increasing sparsity in deeper layers.

UV Atlas Layout To maximize compactness, we pack the K progressive layers into a single texture atlas $\mathcal{A}$ via recursive subdivision in layout that resembles a quadtree [94] (see Figure 2 (c)):

- Layer 0: Occupies right region at full resolution $M_0 \times N_0$.
- Layers 1–2: L_1 (rotated 90° CCW) and L_2 (horizontal) subdivide the left region.
- Layers 3+: Continue recursive packing rightward of L_2, alternating orientation (odd layers rotated, even horizontal) at progressively finer resolutions.

This packing technique achieves **88.5% efficiency** (utilized pixels / total atlas pixels) while significantly outperforming grid (∼60%) or pyramid (∼75%) layouts. The atlas dimensions we use are:

$$W_{\mathcal{A}} = N_0 + \sum_{k=1}^{K-1} N_k, \quad H_{\mathcal{A}} = \max_k M_k.$$

Finally, to represent volumetric video, we assemble a continuous sequence of such UV atlases, each one representing 1 video frame. PackUV is seamlessly integrated into our training pipeline: during optimization, UV maps are maintained at their respective progressive resolutions, and upon convergence, layers are packed into the atlas for efficient storage and streaming (Section 4.2).

4.2. PackUV-GS Fitting

The goal here is to efficiently fit PackUV directly from multiview videos using optical-flow-based keyframing and Gaussian labeling to handle large motions, disocclusions, and maintain temporal consistency.

Fitting UV Maps Directly. Instead of first fitting 3DGS and then projecting to the UV space [61], we initialize and optimize the scene directly within UV space with fixed spatial resolution and predetermined layer count (K). Let $U \in \mathbb{R}^{M\times N\times K\times D}$ denote the UV maps, where $M \times N$ defines the UV grid resolution, K is the number of layers per pixel, and D encodes all Gaussian attributes:

$$U[u_i, v_i, k] = g_i = \{\rho_i,\ r_i,\ s_i,\ o_i,\ c_i\} \in \mathbb{R}^D. \tag{2}$$

This direct optimization not only preserves the structural benefits for downstream tasks but also enforces Gaussian sparsity through the discrete UV grid structure.

4.2.1. Video Keyframing

To efficiently fit given synchronized multi-view videos with T frames, represented as a set $\{V_n\}_{n=1}^{N}$, where N is the total number of views. We divide each video into s set of m temporal segments. To do so, for each frame t, we compute the optical-flow magnitude $M(t)$ on one video, select the top $(m-1)$ magnitude peaks with a minimum separation θ, and use the first frame of every segment as a keyframe. These keyframes define the segment boundaries. For each keyframe F_i^K, the PackUV Gaussians are initialized from the previous keyframe which preserves temporal and spatial consistency. The frames between keyframes are treated as transition frames. Each transition frame F^t is initialized from the preceding frame and refined with a few training iterations compared to F^K:

$$\mathcal{G}(K) \leftarrow \mathrm{Update}(\mathcal{G}(K-1))\,, \mathcal{G}(t) \leftarrow \mathrm{Update}(\mathcal{G}(t-1))\,.$$

This staged, stream-based strategy enables efficient reconstruction of high-fidelity dynamic scenes while allowing us to parallelize the fitting process. Frames exhibiting high drift, occlusions/disocclusions, or appearance breaks are promoted to keyframes. This keyframing technique helps us handle arbitrary length sequences, large motions, and disocclusions without quality degradation over time. More details are given in the supplementary.

4.2.2. Gaussian Labeling

On top of sequential keyframing pipeline for dynamic Gaussian splatting, we also use optical flow to isolate dynamic regions and freezes static Gaussians during optimization. Flow is computed per camera via RAFT [70], and a CUDA-accelerated covariance-aware projection robustly determines which 3D Gaussians overlap with flow-detected dynamics. The approach improves temporal stability and training efficiency on multi-sequence videos while preserving static backgrounds.

Optical Flow and Binary Motion Masks. For each camera view c, we estimate forward optical flow $\mathbf{F}^c_{(t-1)\to t}$ between consecutive frames (I^c_{t-1}, I^c_t) using RAFT [70]. We form a binary motion mask by thresholding the flow magnitude and dilating to include local context:

$$M_t^c(\mathbf{p}) = \begin{cases} 1, & \|\mathbf{F}^c_{t-1\to t}(\mathbf{p})\|_2 > \tau, \\ 0, & \text{otherwise.} \end{cases}$$
$$M_t^c \leftarrow \mathrm{dilate}\big(M_t^c;\, r\big)\,.$$

τ is the flow magnitude threshold and r the dilation radius.

Covariance-Aware Gaussian Masking: Each Gaussian g_i has mean $\boldsymbol{\mu}_i \in \mathbb{R}^3$, diagonal scales $\mathbf{s}_i \in \mathbb{R}^3$, and rotation $\mathbf{q}_i$ (unit quaternion). Its 3D covariance is represented using $\mathbf{R}(\cdot)$ is the rotation matrix and $\mathbf{S}(\mathbf{s}) = \mathrm{diag}(\mathbf{s})$ [31].

Let $\mathbf{T}_c$ denote the camera c's view transformation matrix and $\mathbf{J}_c(\cdot)$ its 2×3 Jacobian evaluated at the camera-space mean. Following EWA splatting [97], the 2D covariance of g_i in image space is

$$\Sigma^{2D}_{i,c} \;=\; \mathbf{J}_c\, \Sigma^{3D}_{i,\mathrm{cam}}\, \mathbf{J}_c^\top\,, \qquad \Sigma^{3D}_{i,\mathrm{cam}} = \mathbf{T}_c\, \Sigma^{3D}_i\, \mathbf{T}_c^\top\,. \tag{3}$$

We then project the mean to normalized device coordinates (NDC), obtain pixel coordinates $\mathbf{m}_{i,c} \in \mathbb{R}^2$, and test overlap with the motion mask using the Mahalanobis metric [46]. A pixel $\mathbf{p}$ is inside the ellipse if

$$d^2(\mathbf{p};\, \mathbf{m}_{i,c},\, \Sigma^{2D}_{i,c}) \;=\; (\mathbf{p}-\mathbf{m}_{i,c})^\top \big(\Sigma^{2D}_{i,c}\big)^{-1} (\mathbf{p}-\mathbf{m}_{i,c}) \;\le\; 9\,.$$

A Gaussian is marked dynamic for camera c if any pixel within a radius derived from the ellipse's largest eigenvalue satisfies $M_t^c(\mathbf{p}) = 1$:

$$D_{i,c} \;=\; \bigvee_{\mathbf{p}\in\mathcal{E}_{i,c}} M_t^c(\mathbf{p})\,, \quad \mathcal{E}_{i,c} = \{\mathbf{p}\,|\, d^2(\mathbf{p}) \le 9\}\,. \tag{4}$$

The final dynamic mask across cameras is an OR aggregation:

$$D_i \;=\; \bigvee_{c\in\mathcal{C}} D_{i,c}\,.$$

To make the computations realtime, we implement the above in a custom CUDA kernel that, for each Gaussian, (i) computes $\Sigma^{2D}_{i,c}$ via the projection Jacobian, (ii) estimates a sampling radius from the largest eigenvalue, and (iii) scans pixels within this radius, testing $d^2 \le 9$ and $M_t^c(\mathbf{p}) = 1$.

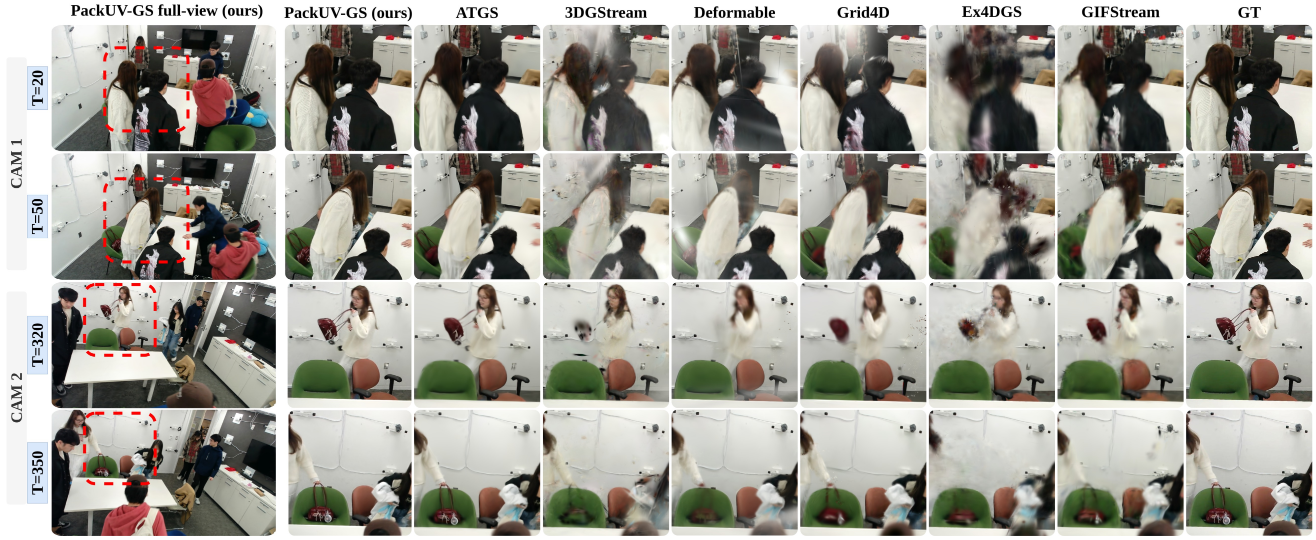

Figure 3. PackUV-GS vs. baselines for large motion and disocclusion handling. The proposed keyframing and Gaussian labeling strategy effectively manages complex scenarios, such as new objects or people entering a room and dispersing. Zoom to view better.

Gradient Freezing. During backpropagation we zero gradients of all static Gaussians, i.e.,

$$\nabla_{\boldsymbol{\theta}_i}\mathcal{L} \leftarrow D_i \nabla_{\boldsymbol{\theta}_i}\mathcal{L}, \qquad D_i \in \{0,1\},$$

where $\boldsymbol{\theta}_i$ includes position, scale, rotation, color, and opacity. We additionally reset optimizer momentum for static Gaussians periodically to prevent drift. When densifying, child Gaussians inherit the parent's dynamic/static label to preserve the dynamic ratio across training.

4.2.3. UV-Based Pruning

To reduce redundancy and keep discretized UV mapping during optimization, we introduce two pruning strategies:

- **Valid UV Projection Pruning.** After densification, we recalculate each child Gaussian's UV coordinates $(u_{\text{scaled}}, v_{\text{scaled}})$. Gaussians failing to satisfy equation 1 are pruned. This structural pruning enforces sparsity while aligning Gaussians more closely with scene geometry, improving both efficiency and visual quality.
- **Max-K UV Pruning.** To prevent overpopulation at individual UV pixel, we retain only the top K Gaussians per pixel based on opacity. Formally, for each coordinate (u, v) with associated Gaussians $\mathcal{G}_{uv}$:
$|\mathcal{G}_{uv}| > K \quad \Rightarrow \quad \mathcal{G}_{uv} \leftarrow \text{TopK}(\mathcal{G}_{uv}, K)$

These pruning techniques reduce memory overhead, focus learning on surface-relevant regions, and improve reconstruction quality while accelerating optimization.

4.2.4. Objective

Let $\hat{I}_t^c$ be the rendered image for camera c and ground-truth I_t^c. The photometric objective blends L1 and SSIM:

$$\mathcal{L}_{\text{photo}} = (1-\lambda_{\text{ssim}})\, \|\hat{I}_t^c - I_t^c\|_1 + \lambda_{\text{ssim}} \left(1-\text{SSIM}(\hat{I}_t^c, I_t^c)\right).$$

To discourage floaters and oversized primitives, we regularize scales and opacities, optionally restricted to dynamic Gaussians (D_i=1):

$$\mathcal{L}_{\text{scale}} = \mathbb{E}_i \left[\max\{0,\ \max(\mathbf{s}_i) - s_{\max}\}\right]^2, \tag{5}$$

$$\mathcal{L}_{\text{opacity}} = \mathbb{E}_i\, \alpha_i \,(1-\alpha_i)\,. \tag{6}$$

The total loss is

$$\mathcal{L} = \mathcal{L}_{\text{photo}} + \mathcal{L}_{\text{depth}} + \lambda_{\text{scale}}\, \mathcal{L}_{\text{scale}} + \lambda_{\text{opacity}}\, \mathcal{L}_{\text{opacity}}\,.$$

Low Precision Optimization. Unlike prior methods that quantize Gaussian parameters only after training, PackUV-GS performs low-precision optimization (LPO) directly over the attributes $\theta = \mu, \mathbf{s}, \mathbf{r}, \alpha, \mathbf{c}$. At each iteration, the renderer consumes a uniformly quantized K-bit proxy $\tilde{\theta}$, while gradients flow through a straight-through estimator and FP32 master weights. The training loss (L1+SSIM with a scheduled regularizer) and optimizer (Sparse Adam [47]) remain unchanged. LPO compensates quantization error, preserving both PSNR and training speed. We use 8-bit precision for $\mathbf{s}, \mathbf{r}, \alpha, \mathbf{c}$ and 16-bit for $\mathbf{x}$, later split into two 8-bit channels for storage. This 8-bit–per-channel layout is compatible with standard video codecs (e.g., HEVC, AVC, FFV1), enabling both lossless and lossy compression.

5. PackUV-2B Dataset

Existing multi-view datasets [29, 39, 59, 87, 89] mostly use front-facing cameras and offer limited diversity, making them insufficient for evaluating volumetric video methods under fast motion, large deformation, and disocclusion. We introduce PackUV-2B, a real-world, long-horizon multi-view dataset comprising 100 dynamic sequences and over 2Bframes, captured with 55-88 synchronized cameras. The data spans studio and in-the-wild settings, including human–human, human–object, and robot–object interactions.

Table 1. **Dataset Comparisons.** We compare our newly captured dataset PackUV-2B with existing multi-view datasets across sequence count, total frames, camera setup, resolution, maximum FPS, scenario type, and view range. PackUV-2B contains 100 diverse sequences totaling over 2B (billion) high-quality frames, recorded with more than 50 cameras at 1920×1200 resolution. The capture system supports up to 90 FPS, providing high temporal fidelity.

Dataset	Sequence	Frames	Camera	Resolution	Max FPS	Scenario	View Range
D-NeRF [59]	8	–	–	800×800	–	Simulator	360
CMU Panoptic [29]	65	$\sim$ 15M	31 (+ 480*)	1920×1080	30	Real-World	360
N3DV [39]	6	$\sim$ 38K	21	2028×2704	30	Real-World	face forward
NeRF-DS [89]	8	$\sim$ 8K	2	480×270	–	Real-World	face forward
SelfCap [87]	3	$\sim$ 1.5M	22	3840×2160	60	Real-World	face forward
PackUV-2B	**100**	$\sim$ **2B+**	**55–88**	1920×1200	**90**	Real-World	**360**

Sequences average 10 minutes and reach up to 30 minutes. To form a comprehensive benchmark, PackUV-2B includes activities with broad variation in motion speed (from slow pouring water to fast basketball and dance), motion scale (from table-top manipulation to pickleball), and object properties (rigid, articulated, reflective, transparent). To our knowledge, PackUV-2B surpasses existing datasets in sequence length, camera count, motion complexity, dynamic diversity, and data volume. We present detailed comparisons between PackUV-2B and existing datasets in Table 1. We plan to release PackUV-2B publicly, with the aim of establishing it as a new benchmark for evaluating general-purpose, long-horizon dynamic reconstruction. Additional details about PackUV-2B can be found in the supplementary material.

6. Experiments

In this section, we conduct comprehensive experiments on three different datasets to verify the effectiveness of our method in generating long volumetric videos with large motion and disocclusion while being compatible with the existing video coding infrastructure. Section 6.1 discusses the qualitative and quantitative evaluations of our proposed method and compare it to the baselines. In Section 6.2, we show how PackUV can be losslessly streamed via existing video infrastructure. The ablation study in Section 6.3 shows the effectiveness of each component in our method.

Setup. We set the PackUV atlas resolution M_0 and N_0 to 1024, and the number of UV layers K to 8. θ for keyframing is set to 30. λ_{scale} and λ_{opacity} were set to 0.0001.

Dataset. We conducted experiments on widely used real-world datasets, N3DV [39] and SelfCap [87], as well as our PackUV-2B. The N3DV dataset is collected from 21 cameras to capture the central scene from the face-forward views at 2704×2028 resolution and 30 FPS. We select the *flame_steak* sequence for evaluation and set aside a testing camera following [39]. The SelfCap dataset contains longer sequences, captured with 22 cameras at 4K resolution and 60 FPS. We evaluate on two sequences, *hair_release* and *yoga*, and reserve 2 cameras for testing. We additionally select 5 sequences from PackUV-2B containing multiple subjects and complex motion: *baby_dance*, *spot*, *entering_room*, *object_place*, *kitchen*, and *entering_room*. For all datasets, we downsample to 1.6K resolution following [31].

Baselines. We compare with several state-of-the-art baseline methods, 3DGStream [68], 4DGS [79], RealTime4DGS [90], Deformable3DGS [91], ATGS [11], Grid4D [84], Ex4DGS [34], and GIFStream [37]. We train 3DGStream on the full sequence length. To mitigate high VRAM consumption in Deformable3DGS, RealTime4DGS, 4DGS, and Grid4D, we split the sequences into segments and train each segment independently. Although ATGS employs a streaming-framework, we observe that training on full sequences results in gradient explosion and adopt the segmented training strategy. We performed all the experiments on RTX 3090 GPU, 256 GB of RAM and 8 CPU cores. We evaluate rendering quality using average PSNR, SSIM, and LPIPS [95] across all timestamps.

6.1. Qualitative and Quantitative Results

Qualitative results. Figure 3 shows qualitative comparisons where our method excels at handling large motions and severe disocclusions. Our keyframing and Gaussian-labeling strategy robustly manages complex events, including new objects or people entering and interacting (e.g., people entering a room and dispersing). Additional results appear in the supplementary material. Deformable3DGS, RealTime4DGS, Grid4D, and 4DGS depend on deformation networks, leading to high VRAM usage, poor scalability to long sequences, and difficulty modeling newly emerging objects. Although streaming-based approaches can support longer sequences, they still produce from artifacts. Specifically, 3DGStream struggles with large motions, ATGS suffers from gradient explosion, and GIFStream contains flickering across the sequentially trained segments. In contrast, PackUV-GS maintains higher and more consistent performance across all motion regimes, enabling high-fidelity free-viewpoint rendering with efficient memory usage. As shown in Figure 3,1 and the supplementary material, PackUV-GS produces sharper, more temporally coherent views.

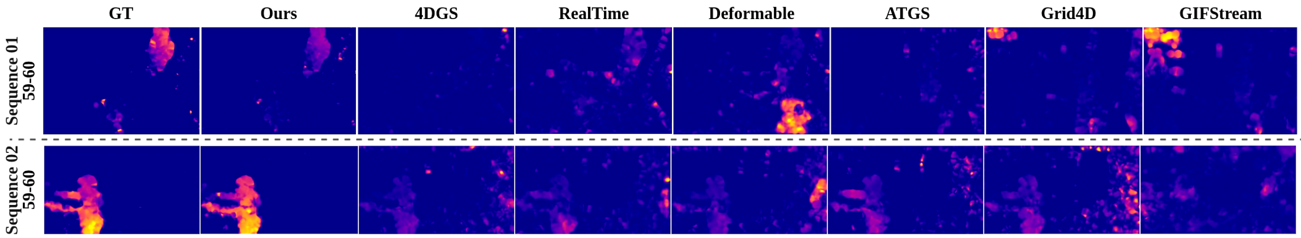

Figure 4. Optical flow. To assess long-term temporal stability, we compute optical flow between consecutive timestamps.

Table 2. Quantitative Comparison. We report PSNR, SSIM, LPIPS, and train time (in hours) for a window length of 60 timestamps. We also report the method's streaming ability and compatibility with the existing video coding infrastructure.

Method	PackUV-2B				SelfCap [86]				N3DV [39]					
	PSNR↑	SSIM↑	LPIPS↓	Train↓	PSNR↑	SSIM↑	LPIPS↓	Train↓	PSNR↑	SSIM↑	LPIPS↓	Train↓	Stream	Codec
3DGStream	23.17	0.826	0.33	1.00	19.77	0.769	0.36	1.43	31.17	0.952	0.23	0.31	Full	No
4DGS	23.11	0.808	0.35	2.30	19.56	0.745	0.37	3.18	29.81	0.951	0.21	3.27	No	No
RealTime	21.37	0.790	0.38	4.48	19.46	0.747	0.41	8.07	32.29	0.955	0.21	2.48	No	No
Deformable	20.07	0.778	0.33	2.04	17.89	0.708	0.38	2.09	26.51	0.935	0.24	0.62	No	No
ATGS	21.42	0.796	0.36	1.13	15.48	0.664	0.51	1.82	30.99	0.934	0.24	1.97	Pseudo	No
Grid4D	21.58	0.790	0.37	1.13	17.53	0.701	0.44	1.82	30.87	0.954	0.197	1.97	No	No
Ex4DGS	20.73	0.789	0.39	0.83	17.62	0.680	0.39	1.23	31.57	0.944	0.23	0.59	No	No
GIFStream	21.92	0.795	0.39	1.61	19.78	0.745	0.35	2.05	31.10	0.954	0.25	0.42	Pseudo	Partial
Ours	27.41	0.842	0.28	1.05	22.52	0.783	0.31	1.12	32.81	0.953	0.21	1.37	Full	Full

Table 3. We present quantitative ablation study for various components of our method on PSNR, SSIM, and LPIPS.

Method	PSNR↑	SSIM↑	LPIPS↓	Method	PSNR↑	SSIM↑	LPIPS↓
Ours	27.41	0.84	0.28	w/o Keyframe	20.95	0.77	0.38
w/o UV Optim	23.81	0.79	0.33	w/o Labeling	25.42	0.82	0.31
No Atlas	27.43	0.84	0.28	w/o Codec	27.41	0.84	0.28
No LPO	27.52	0.85	0.27	–	–	–	–

Quantitative results. Table 2 quantitatively demonstrates that PackUV-GS outperforms all baselines in visual quality metrics across datasets. While 3DGStream, ATGS, and GIFStream support streaming longer sequences, they still yield suboptimal performance. This demonstrates the effectiveness of our proposed representation and method for handling long-duration sequences, while maintaining compatibility with standard multimedia infrastructure.

Long-term temporal consistency. To assess long-term temporal stability, we compute optical flow between consecutive timestamps from novel viewpoints. For methods that require a segmented training strategy, we compute optical flow for consecutive timestamps between segments. As shown in Figure 4, this demonstrates the inability of deformation based methods to maintain temporal consistency over time. This further highlights the importance of online fitting methods like ours. It is interesting to note that the rendering quality of 3DGStream and ATGS degrades over time whereas PackUV-GS is consistent.

6.2. Video Coding of PackUV

We can encode PackUV using standard 2D video codecs while losslessly preserving quality of the 4D scene. Because PackUV-GS maintains temporally consistent UV layouts and only updates per-pixel attributes over time for dynamic regions, the atlas sequence exhibits *strong spatial locality* and *temporal coherence*, allowing direct reuse of mature video coding pipelines.

For each frame in PackUV, we group PackUV layers into 8-bit triplets before encoding them with lossless codecs (*e.g.*, FFV1, HuffYUV) via FFmpeg [71]. Each channel is globally normalized using per-channel min/max computed over the entire sequence. To guarantee bit-exact recovery of the original attributes, we transmit a compact sidecar containing the exact normalization parameters. Decoding is thus invertible: it recovers the atlas sequence, which is de-normalized to reproduce the original values.

This design reduces 4D Gaussian scenes to conventional video assets without custom codecs, enabling storage, streaming, and decoding with off-the-shelf tooling. In practice, we obtain a perfect reconstruction with zero error in the lossless setting via FFV1. These results confirm that PackUV enables treating 4DGS as standard video content while retaining faithful scene recovery.

6.3. Ablation Study and Discussion

We perform comprehensive ablations on PackUV-2B to validate each component of our method, examining atlas-based packing, Gaussian labeling, low-precision optimization (LPO), video coding, direct UV-space optimization, and video keyframing (Table 3), using PSNR, SSIM, and LPIPS for evaluation. Removing UV initialization and UV pruning (optimizing only via post-hoc UV projection) leads to a clear drop in PSNR and other metrics, confirming the importance of direct UV-space optimization for fine detail preservation. We also assess mapping from uniform UV-space Gaussians to the PackUV atlas. Because deeper UV layers contain few primitives, compressing them into lower-resolution atlases incurs negligible quality loss.

Our ablations also demonstrate that low-precision optimization is effectively lossless and provides a superior alternative to post-training quantization for efficient Gaussian storage. Moreover, LPO enables easy integration with standard video-coding pipelines without requiring specialized post-processing steps such as in Motion Layering, VCubed, or GIFStream. Because our PackUV representation consists entirely of 8-bit images, it can be directly encoded as video and decoded back without reconstruction loss. Finally, we assess the effect of video keyframing. Resetting gradients at keyframes helps maintain spatial and temporal consistency, preventing the gradual quality degradation observed when training without keyframes. Removing keyframing results in a clear decline in average PSNR and other metrics. Additional results are in the supplementary material.

7. Conclusion

We presented PackUV, a unified 4D Gaussian representation that reorganizes volumetric video into structured UV atlases compatible with standard video codecs, and PackUV-GS, an efficient fitting pipeline that maintains long-term temporal consistency under large motions and disocclusions. By directly optimizing all Gaussian attributes in the UV domain, our approach enables high-quality reconstruction for arbitrarily long sequences while improving streaming efficiency. We further introduced PackUV-2B, the largest long-duration multi-view 4D dataset to date, providing challenging benchmarks for future research. Extensive experiments demonstrate that our method outperforms prior work in quality, scalability, and practicality, offering a promising step toward deployable volumetric video in real-world systems.

References

[1] Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3d gaussian splats through pruning. *arXiv preprint arXiv:2406.18214*, 2024. 3

[2] Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *European Conference on Computer Vision*, pages 441–459. Springer, 2020. 2

[3] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16610–16620, 2023. 2

[4] Aayush Bansal, Minh Vo, Yaser Sheikh, Deva Ramanan, and Srinivasa Narasimhan. 4d visualization of dynamic events from unconstrained multi-view videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5366–5375, 2020. 2

[5] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5855–5864, 2021. 2

[6] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)*, 39(4):86–1, 2020. 2

[7] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, page 425–432, New York, NY, USA, 2001. Association for Computing Machinery. 2

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

[9] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.*, 32(3), 2013. 2

[10] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European conference on computer vision*, pages 333–350. Springer, 2022. 2

[11] Decai Chen, Brianne Oberson, Ingo Feldmann, Oliver Schreer, Anna Hilsmann, and Peter Eisert. Adaptive and temporally consistent gaussian surfels for multi-view dynamic reconstruction. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 742–752. IEEE, 2025. 2, 3, 7

[12] Rongsen Chen, Fang-Lue Zhang, Simon Finnie, Andrew Chalmers, and Taehyun Rhee. Casual 6-dof: free-viewpoint panorama using a handheld 360 camera. *IEEE Transactions on Visualization and Computer Graphics*, 29(9):3976–3988, 2022. 2

[13] Michael Cohen, Steven J. Gortler, Richard Szeliski, Radek Grzeszczuk, and Rick Szeliski. The lumigraph. Association for Computing Machinery, Inc., 1996. 2

[14] Pinxuan Dai, Peiquan Zhang, Zheng Dong, Ke Xu, Yifan Peng, Dandan Ding, Yujun Shen, Yin Yang, Xinguo Liu, Rynson WH Lau, et al. 4d gaussian videos with motion layering. *ACM Transactions on Graphics (TOG)*, 44(4):1–14, 2025. 2, 3

[15] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. In *Computer Graphics Forum*, pages 305–314. Wiley Online Library, 2012. 2

[16] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts

Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. Fusion4d: real-time performance capture of challenging scenes. *ACM Trans. Graph.*, 35(4), 2016. 2

[17] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, page 65–74, New York, NY, USA, 1988. Association for Computing Machinery. 2

[18] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d-rotor gaussian splatting: towards efficient novel view synthesis for dynamic scenes. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 2, 3

[19] DVB Project. Study mission on volumetric video – a new entertainment experience (dvb bluebook s101). Technical Report BlueBook S101, Digital Video Broadcasting (DVB), 2024. 2

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

[21] Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision*, pages 165–181. Springer, 2024. 3

[22] Peter Fasogbon, Surarshan Bisht, Jaakko Kernen, Ugurcan Budak, Lauri Ilola, and Lukasz Kondrad. Volumetric video use cases for xr immersive streaming. In *Proceedings of the 2024 8th International Conference on Education and Multimedia Technology*, page 1–8, New York, NY, USA, 2024. Association for Computing Machinery. 2

[23] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 2

[24] Wanshui Gan, Hongbin Xu, Yi Huang, Shifeng Chen, and Naoto Yokoya. V4d: Voxel for 4d novel view synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 2

[25] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. *The Lumigraph*. Association for Computing Machinery, New York, NY, USA, 1 edition, 2023. 2

[26] Kaiyuan Hu, Yili Jin, Haowen Yang, Junhua Liu, and Fangxin Wang. Fsvvd: A dataset of full scene volumetric video. In *Proceedings of the 14th Conference on ACM Multimedia Systems*, pages 410–415, 2023. 2

[27] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4220–4230, 2024. 2

[28] Mustafa Işık, Martin Rünz, Markos Georgopoulos, Taras Khakhulin, Jonathan Starck, Lourdes Agapito, and Matthias Nießner. Humanrf: High-fidelity neural radiance fields for humans in motion. *ACM Transactions on Graphics (TOG)*, 42(4):1–12, 2023. 2

[29] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 2, 6, 7

[30] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Trans. Graph.*, 35(6), 2016. 2

[31] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2, 3, 5, 7

[32] Seoha Kim, Jeongmin Bae, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Sync-nerf: Generalizing dynamic nerfs to unsynchronized videos. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2777–2785, 2024. 2

[33] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. In *European Conference on Computer Vision*, pages 252–269. Springer, 2025. 2

[34] Junoh Lee, ChangYeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. Fully explicit dynamic gaussian splatting. *Advances in Neural Information Processing Systems*, 37:5384–5409, 2024. 3, 7

[35] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024. 2, 3

[36] Marc Levoy and Pat Hanrahan. *Light Field Rendering*. Association for Computing Machinery, New York, NY, USA, 1 edition, 2023. 2

[37] Hao Li, Sicheng Li, Xiang Gao, Abudouaihati Batuer, Lu Yu, and Yiyi Liao. Gifstream: 4d gaussian-based immersive video with feature stream. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 21761–21770, 2025. 3, 7

[38] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Ping Tan. Streaming radiance fields for 3d video synthesis. *Advances in Neural Information Processing Systems*, 35: 13485–13498, 2022. 2

[39] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022. 2, 6, 7, 8

[40] Yiqing Liang, Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. Gaufre: Gaussian deformation fields for real-time dynamic novel view synthesis. In *Proc. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025. 2

[41] 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 2023 Conference Papers*, pages 1–9, 2023. 2

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

[43] Wenkai Liu, Tao Guan, Bin Zhu, Luoyuan Xu, Zikai Song, Dan Li, Yuesong Wang, and Wei Yang. Efficientgs: Streamlining gaussian splatting for large-scale high-resolution scene representation. *IEEE MultiMedia*, 2025. 3

[44] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 2

[45] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024. 2

[46] Prasanta Chandra Mahalanobis. On tests and measures of group divergence. *J. Asiat. Soc. Bengal*, 26:541–588, 1930. 5

[47] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, New York, NY, USA, 2024. Association for Computing Machinery. 6, 3

[48] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6878–6887, 2019. 2

[49] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

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

[51] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compgs: Smaller and faster gaussian splatting with vector quantization. In *European Conference on Computer Vision*, pages 330–349. Springer, 2024. 3

[52] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015. 2

[53] Michael Niemeyer, Fabian Manhardt, Marie-Julie Rakotosaona, Michael Oechsle, Daniel Duckworth, Rama Gosula, Keisuke Tateno, John Bates, Dominik Kaeser, and Federico Tombari. Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. *arXiv preprint arXiv:2403.13806*, 2024. 3

[54] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L. Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Qin Cai, Philip A. Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, page 741–754, New York, NY, USA, 2016. Association for Computing Machinery. 2

[55] Rafael Pagés, Konstantinos Amplianitis, Jan Ondrej, Emin Zerman, and Aljosa Smolic. Volograms & v-sense volumetric video dataset. *ISO/IEC JTC1/SC29/WG07 MPEG2021/m56767*, 2021. 2

[56] Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the memory footprint of 3d gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1):1–17, 2024. 3

[57] Sida Peng, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Representing volumetric videos as dynamic mlp maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4252–4262, 2023. 2

[58] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Trans. Graph.*, 36(6), 2017. 2

[59] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 2, 6, 7

[60] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2

[61] Aashish Rai, Dilin Wang, Mihir Jain, Nikolaos Sarafianos, Kefan Chen, Srinath Sridhar, and Aayush Prakash. Uvgs: Reimagining unstructured 3d gaussian splatting using uv mapping. *arXiv preprint arXiv:2502.01846*, 2025. 2, 3, 5, 1

[62] Ignacio Reimat, Evangelos Alexiou, Jack Jansen, Irene Viola, Shishir Subramanyam, and Pablo Cesar. Cwipc-sxr: Point cloud dynamic human dataset for social xr. In *Proceedings of the 12th ACM Multimedia Systems Conference*, pages 300–306, 2021. 2

[63] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. 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*, pages 16632–16642, 2023. 2

[64] Aljosa Smolic, Konstantinos Amplianitis, Matthew Moynihan, Neill O'Dwyer, Jan Ondrej, Rafael Pagés, Gareth W Young, and Emin Zerman. Volumetric video content creation

for immersive xr experiences. In *London Imaging Meeting*, pages 54–54. Society for Imaging Science and Technology IS&T 7003 Kilworth Lane . . . , 2022. 2

[65] Bong-Soo Sohn, Chandrajit Bajaj, and Vinay Siddavanahalli. Volumetric video compression for interactive playback. *Comput. Vis. Image Underst.*, 96(3):435–452, 2004. 2

[66] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742, 2023. 2

[67] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 175–184, 2019. 2

[68] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20675–20685, 2024. 2, 3, 7

[69] Yuan-Chun Sun, I-Chun Huang, Yuang Shi, Wei Tsang Ooi, Chun-Ying Huang, and Cheng-Hsin Hsu. A dynamic 3d point cloud dataset for immersive applications. In *Proceedings of the 14th Conference on ACM Multimedia Systems*, pages 376–383, 2023. 2

[70] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 5

[71] The FFmpeg team. FFmpeg. `https://ffmpeg.org`, 2025. Accessed: May 24, 2025. 8

[72] Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. Ignor: Image-guided neural object rendering. *arXiv preprint arXiv:1811.10720*, 2018. 2

[73] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 551–560, 2020. 2

[74] Feng Wang, Sinan Tan, Xinghang Li, Zeyue Tian, Yafei Song, and Huaping Liu. Mixed neural voxels for fast multi-view video synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19706–19716, 2023. 2

[75] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Jingyi Yu, and Lan Xu. Fourier plenoctrees for dynamic radiance field rendering in real-time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13524–13534, 2022.

[76] Liao Wang, Qiang Hu, Qihan He, Ziyu Wang, Jingyi Yu, Tinne Tuytelaars, Lan Xu, and Minye Wu. Neural residual radiance fields for streamably free-viewpoint videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 76–87, 2023. 2

[77] Yanru Wang, Zhihao Huang, Hao Zhu, Wei Li, Xun Cao, and Ruigang Yang. Interactive free-viewpoint video generation. *Virtual Reality & Intelligent Hardware*, 2(3):247–260, 2020. 3D Visual Processing and Reconstruction Special Issue. 2

[78] Yuxuan Wang, Xuanyu Yi, Zike Wu, Na Zhao, Long Chen, and Hanwang Zhang. View-consistent 3d editing with gaussian splatting. In *European conference on computer vision*, pages 404–420. Springer, 2024. 3

[79] 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)*, pages 20310–20320, 2024. 7

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

[81] Tiange Xiang, Kai Li, Chengjiang Long, Christian Häne, Peihong Guo, Scott Delp, Ehsan Adeli, and Li Fei-Fei. Repurposing 2d diffusion models with gaussian atlas for 3d generation. *arXiv preprint arXiv:2503.15877*, 2025. 2, 3, 1

[82] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics, 2024. 2

[83] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, pages 641–676. Wiley Online Library, 2022. 2

[84] Jiawei Xu, Zexin Fan, Jian Yang, and Jin Xie. Grid4d: 4d decomposed hash encoding for high-fidelity dynamic gaussian splatting. *Advances in Neural Information Processing Systems*, 37:123787–123811, 2024. 2, 7

[85] Zhen Xu, Sida Peng, Haotong Lin, Guangzhao He, Jiaming Sun, Yujun Shen, Hujun Bao, and Xiaowei Zhou. 4k4d: Real-time 4d view synthesis at 4k resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20029–20040, 2024. 2

[86] 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*, 43(6), 2024. 2, 8, 9

[87] 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):1–18, 2024. 2, 3, 6, 7

[88] Jinbo Yan, Rui Peng, Luyang Tang, and Ronggang Wang. 4d gaussian splatting with scale-aware residual field and adaptive optimization for real-time rendering of temporally complex dynamic scenes. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 7871–7880, 2024. 2

[89] Zhiwen Yan, Chen Li, and Gim Hee Lee. Nerf-ds: Neural radiance fields for dynamic specular objects. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8285–8295, 2023. 2, 6, 7

[90] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 2, 3, 7

[91] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20331–20341, 2024. 2, 7

[92] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. 2024. 2

[93] Gareth W. Young, Néill O'Dwyer, and Aljosa Smolic. Chapter 21 - volumetric video as a novel medium for creative storytelling. In *Immersive Video Technologies*, pages 591–607. Academic Press, 2023. 2

[94] Lvmin Zhang, Shengqu Cai, Muyang Li, Gordon Wetzstein, and Maneesh Agrawala. Frame context packing and drift prevention in next-frame-prediction video diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 4

[95] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 7

[96] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 23(3):600–608, 2004. 2

[97] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. 5

PackUV: Packed Gaussian UV Maps for 4D Volumetric Video

Supplementary Material

8. PackUV-2B Dataset

We capture PackUV-2B, a novel real-world dataset featuring long-horizon, multi-view dynamic sequences. PackUV-2B comprises 100 such sequences, with over 2B(billion) frames in total, and encompasses a diverse array scenarios, including human-human interaction, human-object interaction, robot-object interaction, among others. The sequences in PackUV-2B average approximately 10 minutes in length, with some extending up to 30 minutes. Notably, to establish PackUV-2B as a comprehensive benchmark for evaluating dynamic reconstruction approaches, we have curated sequences of varying difficulty levels, aiming to cover a broad spectrum of real-world settings. For instance, in terms of motion speed, PackUV-2B includes movements ranging from slow actions like "inching along" to fast-paced activities such as playing basketball, thereby posing challenges for handling motion blur. Regarding motion scale, PackUV-2B captures both small-scale interactions, like table-top object manipulation, and large-scale activities, such as dance, pickle ball, volleyball, etc. Furthermore, PackUV-2B features diverse object categories, including rigid and articulated objects, as well as some reflective and transparent items. To the best of our knowledge, PackUV-2B significantly surpasses existing datasets in its domain concerning sequence length, number of camera views, motion complexity, dynamic diversity, and overall data volume. More details about the captured sequences are presented in Table 4. The Capture system is discussed in Supp. Sec. 14.

9. Additional Qualitative Results

More qualitative results are shown in figures 8, 10, 9, 11 at the end of supplementary.

10. Lossy Post-Optimization UV Mapping for Scenes

Fig. 5 show lossy UVGS mapping for scenes even when using 48 layers and 1K resolution UV maps. This clearly illustrates the importance of UV mapping while optimizing the Gaussians.

11. Video Coding and PackUV Storage

PackUV allow easy and **lossless** encoding of Volumetric videos using standard 2D video codecs while preserving quality of the 4D scene. Because PackUV-GS maintains temporally consistent UV layouts and only updates per-pixel attributes over time for dynamic regions, the atlas sequence exhibits *strong spatial locality* and *temporal coherence*, allowing direct reuse of mature video coding pipelines. This solved one of the major drawbacks of streaming based methods like GIFStream, 3DGStream and AT-GS - high storage requirements of the trained models for every timestamp for lossless conversion. Using the FFV1 codec for lossless video conversion is giving us an average storage rate of under **10 MBPS**.

Another advantage of PackUV is it's structural arrangement, which can be used for mapping the scene to a latent space to achieve neural compression [61, 81]. Unlike UVGS [61], GaussianAtlas [81], PackUV allows us to represent the entire scene into a single multiscale atlas. This further reduces the complexity of neural networks required to compress it to a latent space. The objective is to compress a single-layer UV atlas representation—which encodes full PackUV atlas to a compact representation that is both easy to store and invertible back to a 4D scene representation. Since the UV format structurally organizes scene attributes in an image-like format, we directly adopt image-based compression pipelines.

11.1. Latent Space Storage

At first, we consider a neural network architecture for autoencoding. We exploit this by designing three lightweight encoder-decoder networks, each responsible for compressing a specific modality:

- **Position Encoder-Decoder:** maps spatial information $\sigma_i \in \mathbb{R}^{3\times H_0\times W_0}$
- **Appearance Encoder-Decoder:** maps RGB/SH color and opacity $\{c_i, o_i\} \in \mathbb{R}^{4-45\times H_0\times W_0}$
- **Covariance Encoder-Decoder:** maps scale and rotation $\{s_i, r_i\} \in \mathbb{R}^{7\times H_0\times W_0}$

Each modality is encoded into a 3-channel latent image per frame. Let $L^t \in \mathbb{R}^{M\times N\times 3}$ be the per-frame latent representation at time t. These latent images are stored along with the shared decoder weights for the entire scene. This approach offers a consistent compression ratio ranging from **[8:1]** to **[128:1]** at the cost of minimal degradation (**PSNR drop: [1.5] dB** to **PSNR drop: [4.5] dB**). And the decoding process is very efficient due to the light-weight MLP based decoder.

12. PackUV-GS Optimization Details

Gaussian Labeling: Explained in Algorithm 1.
Gradient thresholding: A common challenge in the existing dynamic 3D reconstruction methods is the uncontrolled growth of the number of Gaussian primitives over time. This progressive accumulation leads to excessive memory

Figure 5. Post-optimization lossy UV mapping fails to capture details of a real-world scene even with 48 layers and 1K resolution.

consumption and frequent out-of-memory (OOM) errors, especially when fitting long video sequences. To mitigate this issue, we introduce a *gradient thresholding* mechanism that constrains the total number of Gaussians allowed during optimization. The key idea is to retain only those Gaussians that meaningfully contribute to the reconstruction objective while pruning away redundant or inactive ones.

Let the loss function be denoted by $\mathcal{L}$, and consider a Gaussian primitive g_i. We compute the gradient norm of its contribution to the loss:

$$\|\nabla_{g_i}\mathcal{L}\|_2$$

If this norm falls below a predefined threshold τ, the Gaussian is considered non-contributory and is removed from the scene representation. Formally:

$$\text{If } \|\nabla_{g_i}\mathcal{L}\|_2 > \tau, \quad \text{then } g_i \text{ is pruned}$$

This strategy ensures that only actively contributing Gaussians are retained during optimization, effectively capping the total number of primitives and maintaining computational efficiency. By doing so, we avoid unnecessary memory overhead and preserve high-quality reconstruction over long temporal horizons.

12.1. Video Keyframing

To efficiently fit given synchronized multi-view videos with T frames, represented as a set $\{V_n\}_{n=1}^{N}$, where N is the total number of views. We divide each video into s set of m temporal segments. To do so, for each frame t, we compute the optical-flow magnitude $M(t)$ on one video, select the top $(m-1)$ magnitude peaks with a minimum separation θ, and use the first frame of every segment as a keyframe. These keyframes define the segment boundaries. For each keyframe F_i^K, the PackUV Gaussians are initialized from the previous keyframe which preserves temporal and spatial consistency. The frames between keyframes are treated as transition frames. Each transition frame F^t is initialized from the preceding frame and refined with a few training iterations compared to F^K:

$$\mathcal{G}(K) \leftarrow \text{Update}(\mathcal{G}(K-1))\,, \mathcal{G}(t) \leftarrow \text{Update}(\mathcal{G}(t-1))\,.$$

This staged, stream-based strategy enables efficient reconstruction of high-fidelity dynamic scenes while allowing us to parallelize the fitting process. Frames exhibiting high drift, occlusions/disocclusions, or appearance breaks are promoted to keyframes. This keyframing technique helps us handle arbitrary length sequences, large motions, and disocclusions without quality degradation over time.

Through our experiments we observe that keeping keyframes farther apart can cause the quality to degrade over time, thus we keep the keyframe threshold to 30 timestamps.

Table 4. **PackUV-2B Dataset Layout.** We present our newly captured dataset PackUV-2B sequences . We report number of sequences captured, total timestamps (T), total cameras used to capture the sequence, FPS, setting type, and view range. We also report special tags for each dataset describing the type of activity in the captured sequence. PackUV-2B contains 100 diverse sequences totaling over 2B (billion) high-quality frames, recorded with more than 50 cameras at 1920×1200 resolution. The capture system supports up to 90 FPS, providing high temporal fidelity. Tags: RI - Robot Interaction, HI - Human-human Interaction, OI - Object Interaction, SP - Sports, LM - Large Motion, DO - Disocclusion, TR - Transparent/Reflective Objects, EN - Entertainment .

Sequence	Num Sequences	Num Cameras	FPS	Setting	Tags	View Range
Baby Dance	3	88	30	Studio	EN, HI, DO	360
Spot	3	88	30, 90	Studio	RI, HI, OI, DO	360
Volleyball	3	88	30, 90	Studio	SP, HI, LM, DO	360
Kitchen	3	55	30	Non-studio	OI	320
Woodwork	4	55	30	Non-studio	OI	320
Pickleball	30	55	30	Non-studio	SP, HI, LM	300
Meat Shop	11	84	30	Non-studio	OI, DO	320
Kuka Robot	2	84	30	Non-studio	RI, OI	300
Panda Robot	3	84	30	Non-studio	RI, OI	300
Articulation	2	84	30	Studio	HI, OI, DO	360
Chair Play	1	86	30	Studio	HI, OI, DO, LM	360
Object Placement	1	88	30	Studio	HI, OI, DO, LM	360
Dance 01	3	80	30	Studio	HI, DO, LM, EN	360
Dance 02	7	84	30, 60, 90	Studio	HI, DO, LM, EN	360
Dance 03	3	82	30, 60, 90	Studio	DO, LM, EN	360
Dance 04	3	85	30	Studio	DO, LM, EN	360
Yoga	3	78	30	Studio	LM, SP	360
Tools Play	2	82	30	Studio	LM, DO, OI	360
Board Games	4	86	30	Studio	HI, OI, LM, DO	360
Photography	4	88	30	Studio	HI, OI, LM, DO	360
Conversation	5	84	30	Studio	HI, LM, DO	360
PackUV-2B (TOTAL)	**100**	**55–88**	30-90	-	-	**∼360**

12.2. Low-Precision Training

Unlike most prior works that employ lossy quantization after optimization for storage efficiency, PackUV-GS utilizes low precision optimization (LPO) for learning the Gaussian attributes $\theta = \{\mu, \mathbf{s}, \mathbf{r}, \alpha, \mathbf{c}\}$. Our experiments show that LPO effectively compensates for the quantization loss, maintaining both PSNR and training efficiency. At each iteration, the renderer operates on a quantized proxy $\tilde{\theta}$ obtained by a uniform K-bit fake quantizer with scale Δ. We use a straight-through estimator (STE) to preserve gradient flow while keeping master weights in FP32. The training objective remains the same, combining an L1+SSIM photometric term with a scheduled regularizers. Backpropagation updates FP32 master parameters with Sparse Adam optimizer [47].

We use 8-bit reduced precision for $\{\mathbf{s}, \mathbf{r}, \alpha, \mathbf{c}\}$ and 16-bit for $\{\mathbf{x}\}$. During storage, the 16-bit $\{\mathbf{x}\}$ values are split into two 8-bit parts. This 8-bit-per-channel design makes PackUV readily compatible existing video coding infrastructures, enabling the direct application of both lossless or lossy compression methods (*e.g*., HEVC, AVC, FFV1).

We also conduct several experiments to evaluate the loss incurred from reduced-precision training using different levels of bit quantization. From these experiments, we observe that that quantizing the position parameters ($\{\mathbf{x}\}$) cause the largest PSNR drop during training. Consequently, we retain them in FP16, which introduces negligible error during optimization. In contrast, the other attributes ($\{\mathbf{s}, \mathbf{r}, \alpha, \mathbf{c}\}$) show no noticeable PSNR drop in our experiments, even when quantized to 8-bit integers after appropri-

Algorithm 1 Covariance-Aware Flow Masking (with/without CUDA kernel)

Require: Gaussian parameters $\{\boldsymbol{\mu}_i, \mathbf{s}_i, \mathbf{q}_i\}_{i=1}^{N}$, camera views $\mathcal{C}$, flow masks $\{M_t^c\}_{c\in\mathcal{C}}$, use_cuda flag
Ensure: Dynamic mask $D_i \in \{0, 1\}$ for each Gaussian i

Initialize $D_i \leftarrow 0$ for all $i = 1, \dots, N$ ▷ All static by default
if use_cuda **and** CUDA available **then**
    // CUDA Path: Covariance-Aware Projection
    for each camera $c \in \mathcal{C}$ **do**
        for each Gaussian i **in parallel (CUDA) do**
            // Step 1: Transform to camera space
            Compute 3D covariance: $\Sigma_i^{3D} = \mathbf{R}(\mathbf{q}_i)\mathbf{S}(\mathbf{s}_i)^2\mathbf{R}(\mathbf{q}_i)^\top$
            $\Sigma_{i,\mathrm{cam}}^{3D} \leftarrow \mathbf{T}_c \Sigma_i^{3D} \mathbf{T}_c^\top$
            $\boldsymbol{\mu}_{i,\mathrm{cam}} \leftarrow \mathbf{T}_c \boldsymbol{\mu}_i$
            // Step 2: Project to 2D image space
            Compute projection Jacobian $\mathbf{J}_c$ at $\boldsymbol{\mu}_{i,\mathrm{cam}}$
            $\Sigma_{i,c}^{2D} \leftarrow \mathbf{J}_c \Sigma_{i,\mathrm{cam}}^{3D} \mathbf{J}_c^\top$
            $\mathbf{m}_{i,c} \leftarrow \Pi_c(\boldsymbol{\mu}_i)$ ▷ Project to pixel coords
            // Step 3: Compute sampling radius from covariance
            $\lambda_{\max} \leftarrow$ largest eigenvalue of $\Sigma_{i,c}^{2D}$
            $r_{\mathrm{sample}} \leftarrow \lceil 3\sqrt{\lambda_{\max}} \rceil$ ▷ 3-sigma coverage
            // Step 4: Test ellipse overlap with flow mask
            $D_{i,c} \leftarrow 0$
            for $\mathbf{p} \in \{\mathbf{m}_{i,c} + \Delta \mid \|\Delta\|_\infty \le r_{\mathrm{sample}}\}$ **do**
                if $\mathbf{p}$ within image bounds **then**
                    $d^2 \leftarrow (\mathbf{p} - \mathbf{m}_{i,c})^\top (\Sigma_{i,c}^{2D})^{-1} (\mathbf{p} - \mathbf{m}_{i,c})$
                    if $d^2 \le 9$ **and** $M_t^c(\mathbf{p}) = 1$ **then**
                        $D_{i,c} \leftarrow 1$ ▷ Mark dynamic
                        break
                    end if
                end if
            end for
            // Step 5: Aggregate across cameras
            if $D_{i,c} = 1$ **then**
                $D_i \leftarrow 1$ ▷ Atomic OR
            end if
        end for
    end for
else
    // Fallback: Simple Point Projection
    for each camera $c \in \mathcal{C}$ **do**
        for each Gaussian i **do**
            // Direct projection of Gaussian center
            $\mathbf{p}_i \leftarrow \Pi_c(\boldsymbol{\mu}_i)$ ▷ Project mean to pixel coords
            if $\mathbf{p}_i$ within image bounds **then**
                if $M_t^c(\mathbf{p}_i) = 1$ **then**
                    $D_i \leftarrow 1$ ▷ Mark as dynamic
                end if
            end if
        end for
    end for
end if
return $\{D_i\}_{i=1}^{N}$

ate scaling.

13. Viewer

We also built a custom viewer for rendering our PackUV-2B atlas maps on top of the Tiny Gaussian Splatting Viewer by Li Ma, which originally supports OpenGL and CUDA backends for static scenes. Our viewer extends this functionality to enable interactive camera control, real-time playback at arbitrary frame rates, and frame-by-frame navigation through dynamic sequences.

Rendering is performed using OpenGL shaders and shader storage buffer objects (SSBOs). Each video frame is represented as a PackUV atals stored as a npz file, decreasing our loading latency compared to .ply files. And to accelerate playback, we cache a flattened, OpenGL-ready version of each frame on the CPU. At runtime, when advancing to a new frame, the viewer copies this preprocessed buffer to the GPU for rendering. Between frame updates, the current buffer remains active, allowing continuous rendering and smooth scene exploration.

14. CAPTURE System and Camera Synchronization

For capturing the sequences in PackUV-2B, we constructed a dedicated capture studio equipped with 88 synchronized static cameras, as shown in Figure 6. For the non-studio captures, we also built a wireless version of the similar capture setup. The ultra-large scale of the PackUV-2B dataset raises great challenges to the corresponding data processing steps. To this end, we develop an automatic pipeline to handle camera calibration, color and lighting correction, and automatic synchronization. Notably, the automatic synchronization is efficiently achieved by a carefully designed data structure in Section 14. These cameras are uniformly distributed across the four walls of a rectangular room, enabling the capture of fine-grained details with minimal occlusion. PackUV-2B provides high-resolution frames (1920 × 1200) at frame rates of 30, 60, or 90 FPS, selected based on the level of dynamics in each sequence. We plan to make PackUV-2B publicly available, with the goal of establishing it as a new benchmark for evaluating general-purpose, long-horizon dynamic reconstruction.

CAPTURE is a designated multi-view capture system with 88 cameras, suitable for various kinds of capture settings. This section provides an overview of an AVL tree implementation designed specifically for managing data captured by this system. Based on the timecode of each frame data, this implementation achieves automatic synchronization and efficient search. In practice, it is non-trivial and time-consuming to achieve high-accuracy synchronization efficiently when working with such a large number of cameras. We introduces an AVL tree implementation designed specifically for automatically synchronizing, managing and querying frame data based on timecodes. AVL trees are self-balancing binary search trees, ensuring that operations like insertion, deletion, and search can be performed efficiently, typically in $O(log n)$ time, where n is the number of nodes in the tree.

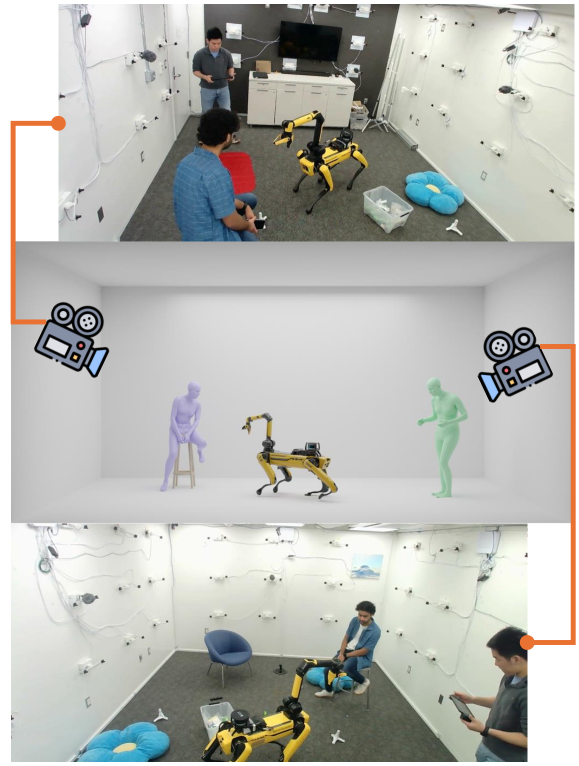

Figure 6. **CAPTURE Studio Layout.**

The primary goal of this AVL tree is to transform unstructured raw frame data into an efficient data structure with synchronized frames. This structure allows for:

1. Quick Lookups: Rapidly searching all the frames at some timecode within a specified tolerance (threshold).
2. Data Persistence: Once an AVL tree is built, it can be saved as a binary file to avoid the need to rebuild it from the raw file in the future.

14.1. Implementation

The overview of the implementation can be divided into two steps:

1. Build an AVL tree for each camera;
2. After randomly selecting a reference camera, iterate through all the frames in the reference camera, and search the closest frames from all the other AVL trees as the synchronized frames.

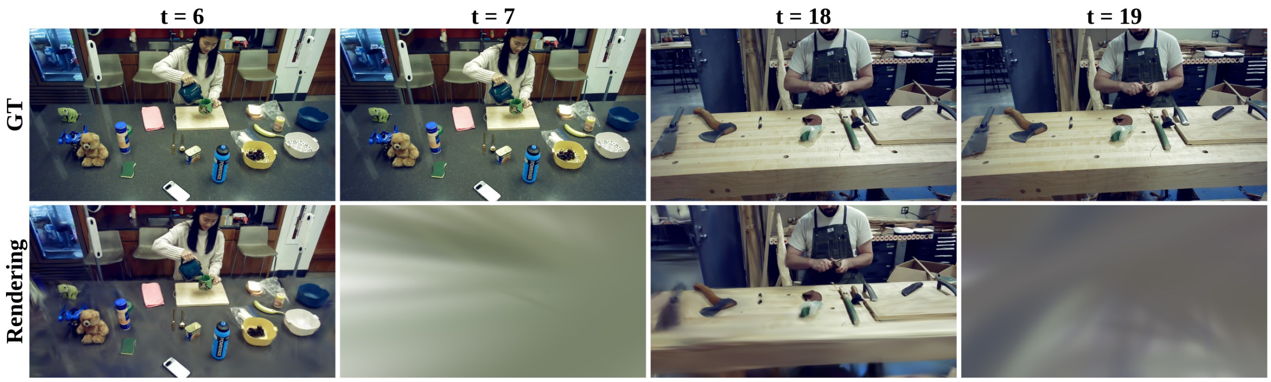

Figure 7. Shows gradient explosion in ATGS training.

14.2. Build an AVL tree

Following 2, We build an AVL tree for each camera based on the camera information file which stores the correspondence of the frame index idx_i and the timecode t_i. It takes as input each pair of $\{idx_i, t_i\}$, and then inserts it as a node into the AVL tree. The AVL tree's ensures an insertion logic that the tree remains balanced after each new node is added.

Algorithm 2 Build an AVL Tree from a Camera Information File

function BUILDAVLTREE(filename)
  root node $r \leftarrow$ NULL
  Open a camera information file as $\mathcal{F}$
  for all f in $\mathcal{F}$ **do**
    Parse f to get a $\{$timecode string t_i, frame index idx_i
    $r \leftarrow$ INSERTINTOAVL(r, t_i, idx_i) ▷ InsertIntoAVL handles node creation and tree balancing
  end for
  Close f
  return r
end function

With a built AVL tree, we can achieve efficient and fast lookups. Since all the data is stored in an AVL tree, which is a type of Binary Search Tree (BST), nodes in a BST are organized such that all nodes in the left subtree of a node have timecodes less than the node's timecode, and all nodes in the right subtree have timecodes greater. This structure allows for efficient searching. Given a timecode to search, the code iterates through the tree, keeping track of the node it has encountered so far whose timecode is closest to the target timecode. It also considers a threshold to ensure that the "closest" frame found is "close enough". If the difference of timecodes between the searched frame and reference frame is larger than a pre-defined threshold, that searched frame will be dropped and viewed as "no synchronized frame".

14.3. Iteratively Synchronize All the Cameras

After building AVL trees for all the cameras, we prepare an automatic workflow (Algorithm 4) to synchronize all of them. Firstly, we sample a reference camera, either randomly or intentionally. Then, we go through all the frames in the reference camera and look up the closest frame of every AVL tree as the synchronized frame. Finally, we save the synchronization information as a json file for future use. Moreover, based on torchcodec, we achieve efficiently extracting any specific frames directly from raw MP4 video file without the need to extracting all the frames first.

15. Limitations and Future Work

While our method achieves high-quality 4D neural video renderings, some limitations remain that requires further exploration. A primary challenge arises from the inherently unstructured nature of 3D Gaussian representations. Although our approach introduces structure through UV projection, it still requires enforcing a large spatial arrangement to capture the fine details of real-world scenes. Furthermore, while PackUV enables the use of video coding infrastructure by mapping to a single frame, discovering more optimal mappings could further improve storage efficiency. Another promising direction is to make this representation compatible with AR/VR devices, thereby enabling direct 4D streaming for immersive applications.

Algorithm 3 Find Closest Frame by Timecode

function FINDCLOSEST(root_node r, target_timecode t, threshold τ)
  closest_node $\hat{r} \leftarrow$ NULL
  min_difference $d_{min} \leftarrow \infty$
  current_node $r' \leftarrow$ root_node r'
  while r' IS NOT NULL **do**
    difference $d \leftarrow$ |current_node.timecode − target_timecode|
    if $d < d_{min}$ **then**
      $d_{min} \leftarrow d$
      $\hat{r} \leftarrow r'$
    end if
    if $d = 0$ **then**
      break ▷ Exact match found, cannot be closer
    end if
    if $t <$ the timecode of r' **then**
      $r' \leftarrow$ the left child node of r'
    else
      $r' \leftarrow$ the right child node of r'
    end if
  end while
  if $\hat{r}$ IS NOT NULL **and** $d_{min} > \tau$ **then**
    return NULL ▷ No node found within threshold
  else
    return $\hat{r}$
  end if
end function

Algorithm 4 Synchronization Workflow

procedure SYNCHRONIZEANDEXTRACTFRAMES
  Define `SyncMetaFile` path.
  if file at `SyncMetaFile` exists **then**
    `SyncInfo` $\leftarrow$ LOADSYNCINFO(`SyncMetaFile`) ▷ Loaded existing synchronization info.
  else
    `AVLTrees` $\leftarrow$ GENERATEAVLTREES ▷ Leverage Algorithm 2
    `SyncInfo` $\leftarrow$ SEARCHFRAMESUSINGAVLTREES ▷ Leverage Algorithm 3
    SAVESYNCINFO(`SyncInfo`, `SyncMetaFile`)
  end if
  EXTRACTANDSAVEIMAGEFRAMESFROMVIDEOS(`SyncInfo`) ▷ Extract specific frames directly from videos based on torchcodec.
end procedure

Figure 8. Baseline comparison on PackUV-2B's *Kitchen* sequence.

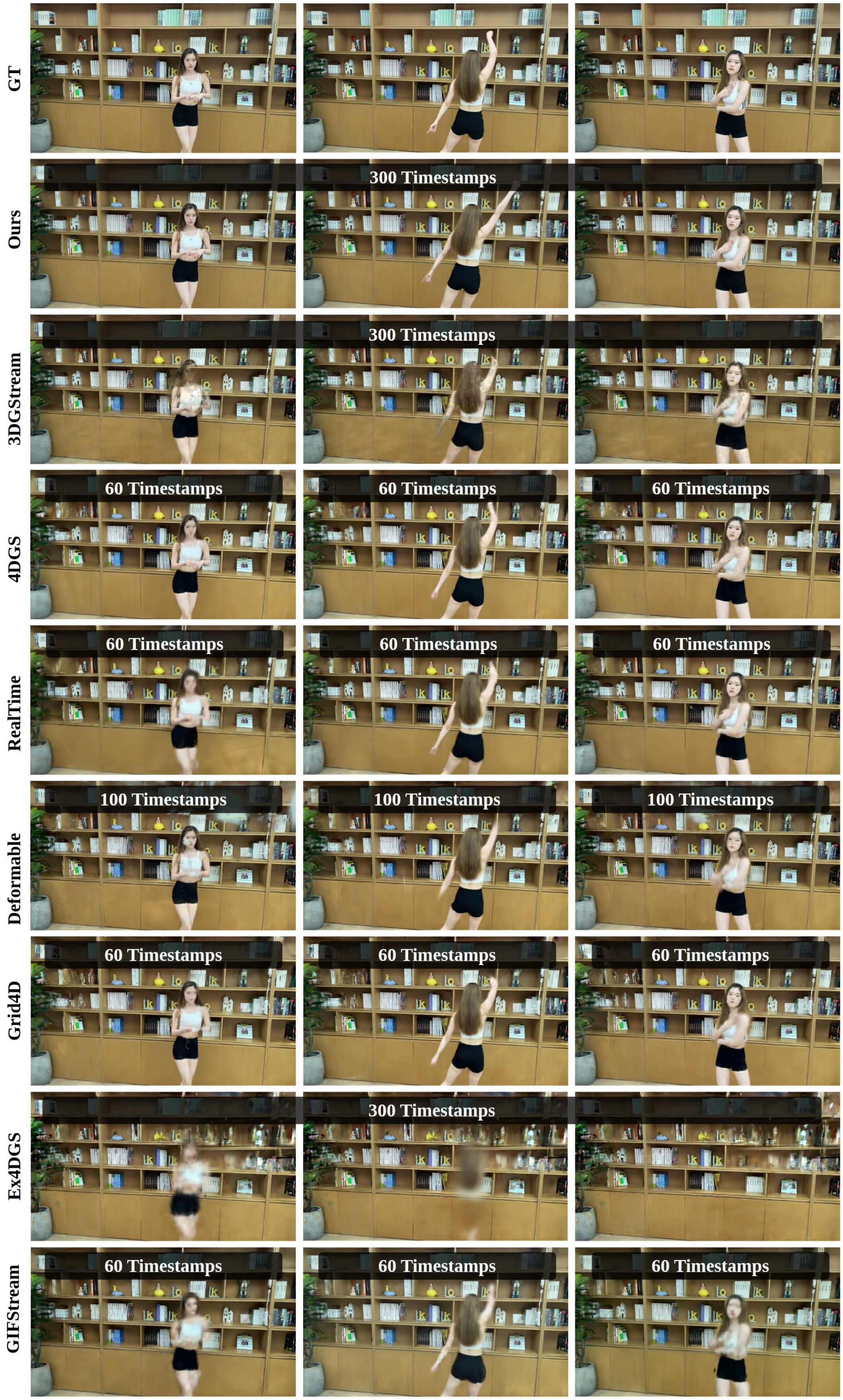

Figure 9. Shows baseline comparison on SelfCap [86] dataset.

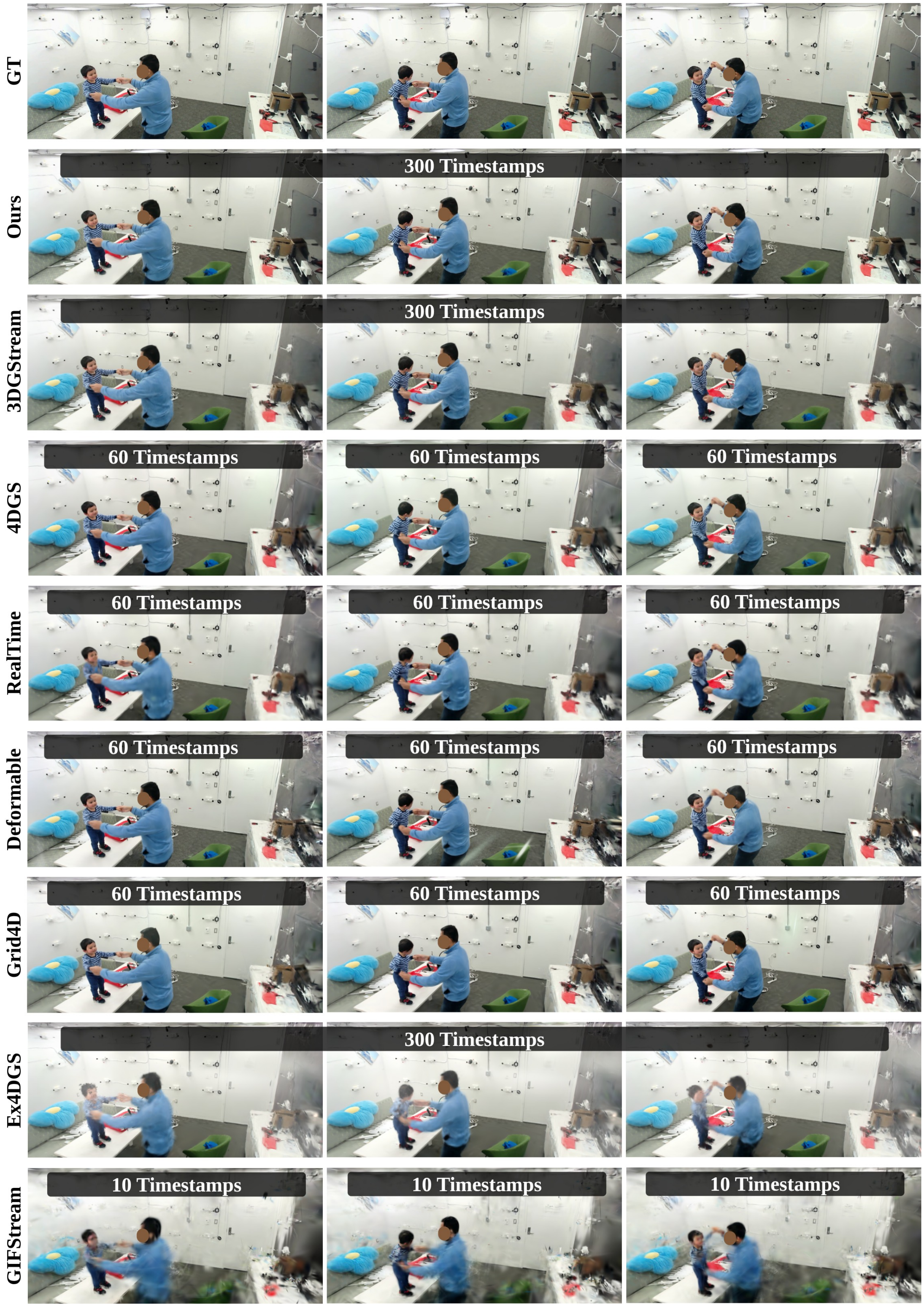

Figure 10. Shows baselines comparison on PackUV-2B's *Baby Dance* sequence.

Figure 11. Baselines comparison on PackUV-2B's *SPOT* sequence.