TRACKSTO4DGS: FEED-FORWARD 4D GAUSSIAN SPLATTING FROM DENSE TRACKS

Anonymous authors
Paper under double-blind review

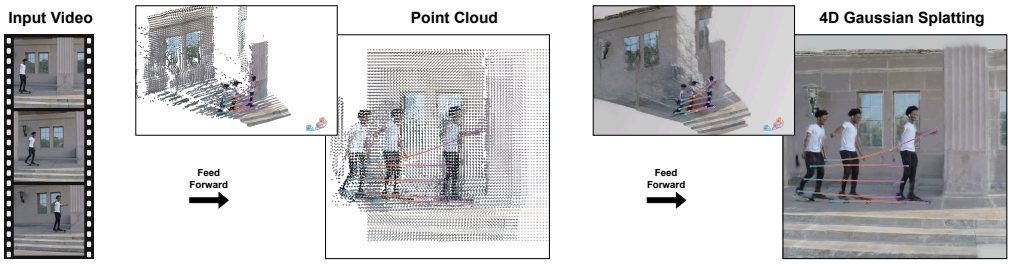

Figure 1: **TRACKSTO4DGS overview.** From a monocular video (left), frozen tracking and reconstruction models produce dense 3D trajectories (middle). A trained network predicts the appearance of one Gaussian per trajectory, giving a dynamic scene renderable from novel views (right). The pipeline is fully feed-forward. Several time instants are composited into each image. See videos at `https://tracksto4dgs.github.io`.

ABSTRACT

Rendering dynamic scenes from monocular video has attracted growing interest, driven by applications in AR/VR, egocentric perception, and embodied AI. The task is challenging, as it requires jointly recovering accurate geometry, motion, and appearance from a single moving viewpoint. Recently, pretrained foundation models for tracking and reconstruction have made remarkable progress on the first two, providing robust 3D geometry and dense correspondences across diverse environments. However, their outputs are point clouds, which lack the representation needed to render high-quality video. We present TRACKSTO4DGS, an architecture that closes this gap by decoupling scene geometry and motion from appearance. Rather than learning temporal dynamics from scratch through an ambiguous rendering loss, TRACKSTO4DGS leverages frozen foundation models to explicitly define the 3D trajectory of each Gaussian, while a dedicated Attribute Network learns only the remaining properties required for high-fidelity rendering. To make dense tracking data computationally tractable, a sparse bottleneck reasons over a compact set of representative trajectories before decoding attributes for the full dense set. By inheriting the robust geometry of these foundation models, TRACKSTO4DGS achieves strong zero-shot generalization. Trained exclusively with a rendering loss on fixed-length egocentric video, a single model improves average PSNR by 2.0 dB over the strongest baseline across six diverse benchmarks, spanning egocentric, handheld, and object-centric captures, and seamlessly generalizes to unseen video lengths without per-dataset tuning.

1 INTRODUCTION

Dynamic novel view synthesis aims to reconstruct a scene from a casual monocular video such that it can be rendered from novel viewpoints at any moment in time. The task has attracted growing interest, driven by applications in AR/VR, egocentric perception, and embodied AI, where everyday footage is captured by a single moving camera. It is also highly challenging: the scene's geometry, motion, and appearance must all be recovered while each moment is observed from only one viewpoint. Under such sparse observations, camera motion, object motion, and depth are entangled, and many different dynamic scenes can explain a given video equally well.

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

Existing approaches follow two paradigms, trading off accuracy against efficiency. Per-video optimization methods (Luiten et al., 2024; Wu et al., 2024; Yang et al., 2024c) fit a dynamic 3D Gaussian representation (Kerbl et al., 2023) to a given video. They repeatedly render the input frames and update the scene's geometry, motion, and appearance until the renderings match. Since no information is shared across videos and since ambiguous motion must be resolved through this slow iterative process, reconstructing even a short clip often takes hours. Feed-forward methods (Xu et al., 2025; Lin et al., 2026; 2025a; Yang et al., 2026) amortize this cost by training a network to predict the representation in a single pass. However, they entangle the entire problem: a single network predicts where Gaussians lie, how they move, and how they look. Motion is thus learned either through a rendering loss, which remains ambiguous even when augmented with geometric priors, or from ground-truth 4D supervision, which is available mostly in synthetic or specially captured data. Either way, these models learn temporal dynamics largely from scratch, and their generalization is bottlenecked by video datasets that are far smaller and less diverse than the internet-scale data powering dedicated geometry foundation models (Wang et al., 2025a; 2026).

Meanwhile, foundation models have made remarkable progress on exactly the components that feed-forward 4D methods struggle to learn. Monocular geometry models (Wang et al., 2025a; 2026) recover depth and camera poses that place each observation in 3D, while dense point trackers (Harley et al., 2025; Karaev et al., 2025) associate observations across time. Combined, they describe the scene's geometry and motion explicitly, and can be lifted to 4D with no renderer in the loop (Kasten et al., 2024). Dense tracking, which once took hours per clip (Xu et al., 2025), is now fast with recent efficient trackers (Harley et al., 2025; Karaev et al., 2025). However, trackers are limited to provide a dynamic point cloud: it specifies the position of each surface point at every moment, but lacks the extent, opacity, and view-dependent appearance required to render a high-quality video. What is missing is therefore not motion, but a way to turn explicit trajectories into a renderable representation.

We present TRACKSTO4DGS, a feed-forward architecture that closes this gap by decoupling geometry and motion from appearance (Figure 1). We represent the scene with dynamic 3D Gaussians whose motion is defined explicitly: dense 2D tracks are deterministically lifted to 3D using the depth and camera poses predicted by frozen foundation models (Wang et al., 2026; Harley et al., 2025), and each lifted trajectory directly serves as the center of a Gaussian throughout the video. An Attribute Network then predicts only what these models do not provide: each Gaussian's covariance, opacity, and view-dependent color. Since motion is given rather than learned, the network is trained on unlabeled videos, supervised solely by how well its renderings match the input frames, without any 4D ground truth. By building on foundation models rather than learning motion from scratch, TRACKSTO4DGS inherits the diversity of their training data and generalizes well to unseen scenes.

While this decoupling makes the learning problem far better posed, implementing it naively, with every dense track as a Gaussian, introduces a major computational challenge. Even a short video yields tens of millions of track points, far exceeding what standard attention mechanisms can process. Our key observation is that nearby points on the same surface tend to move together, so the scene's motion can be described by far fewer tracks than are observed. We therefore design the Attribute Network around a sparse bottleneck: it reasons over a compact set of representative tracks, or *seeds*, together with a low-rank descriptor of their shared motion. The heavy attention and graph operations run only on the seeds, and a lightweight decoder then expands the seed features to assign rendering attributes to every dense track. This keeps computation tractable at full track density and, as we show, allows the model to generalize to video lengths unseen during training.

Extensive evaluations show that this decoupled design generalizes robustly beyond its training data. Trained exclusively on unlabeled egocentric videos of fixed length, a single TRACKSTO4DGS model is evaluated on six benchmarks spanning egocentric, handheld, and object-centric captures, without per-dataset tuning or per-video optimization. It achieves state-of-the-art results, improving average PSNR by 2.0 dB over the strongest baseline and ranking first or second on every evaluated dataset and metric. Moreover, it naturally scales to video lengths unseen during training.

Our contributions are:

- Decoupling geometry and motion from appearance in feed-forward 4D Gaussian splatting, allowing it to inherit the robustness of large, pretrained reconstruction and tracking models.
- Bridging the gap between dense tracking data and efficient feed-forward prediction. Our Attribute Network reasons over a compact sparse bottleneck and a low-rank motion rep-

resentation, then decodes attributes for tens of millions of track points, enabling seamless generalization to unseen video lengths.

- State-of-the-art feed-forward dynamic novel view synthesis with strong zero-shot transfer across diverse benchmarks, improving average PSNR by 2.0 dB over the SOTA feed-forward methods.

2 Related Work

Per-video dynamic reconstruction. Dynamic radiance fields and Gaussian representations extend static reconstruction through deformation fields (Park et al., 2021a;b; Pumarola et al., 2021; Yang et al., 2024c; Wu et al., 2024), scene flow (Li et al., 2021), persistent moving Gaussians (Luiten et al., 2024), or spatiotemporal primitives (Yang et al., 2024b). Fitting them to a monocular video, where viewpoint and scene motion change together, is ill-posed (Gao et al., 2022), so recent methods rely on learned depth and correspondence priors (Liu et al., 2025; Wang et al., 2025b; Lei et al., 2025).

Feed-forward dynamic Gaussian prediction. Building on feed-forward static reconstruction (Charatan et al., 2024; Chen et al., 2024; Zhang et al., 2024; Ye et al., 2025a), recent methods predict dynamic Gaussians in a single pass, learning motion either through rendering losses (Liang et al., 2024; Xu et al., 2025) or from explicit 3D motion supervision (Lin et al., 2025a; 2026). NeoVerse (Yang et al., 2026) further uses such reconstructions to condition novel-trajectory video generation. In all of these, the network that predicts rendering attributes must also recover geometry and motion. TracksTo4DGS instead takes Gaussian trajectories from frozen predictors and learns only the attributes.

Point tracks for dynamic reconstruction. Long-range point tracks (Harley et al., 2022; Doersch et al., 2023; Karaev et al., 2024; 2025) constrain motion across frames and are used by reconstruction methods in several ways. Per-video methods optimize Gaussians with shared motion bases (Wang et al., 2025b) or attached to a scaffold of depth-lifted tracks (Lei et al., 2025), or use triangulated tracks to guide Gaussian updates before optimization (Yin et al., 2025). TracksTo4D (Kasten et al., 2024) instead predicts cameras and low-rank dynamic structure from 2D tracks in a single pass. Together, these works show that tracks are a strong signal for dynamic reconstruction.

Pretrained geometry and tracking models. DUSt3R (Wang et al., 2024) and VGGT (Wang et al., 2025a) demonstrate learned reconstruction of scene geometry from image collections. MonST3R (Zhang et al., 2025), CUT3R (Wang et al., 2025c), and VGGT-Ω (Wang et al., 2026) address geometry estimation in dynamic scenes. Complementary tracking models recover long-range correspondences: AllTracker (Harley et al., 2025) produces dense 2D tracks at high resolution, while SpatialTracker and its successor estimate tracks in 3D (Xiao et al., 2024; 2025). D4RT (Zhang et al., 2026) unifies dynamic reconstruction and tracking through a query-based model. We build on these advances by combining VGGT-Ω and AllTracker as frozen sources of geometry and correspondence, with a separate network for Gaussian attribute prediction.

Low-dimensional motion representations. Low-rank models have a long history in non-rigid structure from motion, where they constrain reconstruction through shape bases (Bregler et al., 2000) or trajectory bases (Akhter et al., 2008). TracksTo4D (Kasten et al., 2024) incorporates a low-rank dynamic structure model into a feed-forward network, and Shape of Motion (Wang et al., 2025b) represents motion through shared $\mathrm{SE}(3)$ bases. These works show that natural motion is well captured by a few bases. We exploit this to make dense tracks tractable.

3 Method

Given a monocular video, our goal is to construct a dynamic 3D representation that enables novel view synthesis without per-video optimization. We build on 3D Gaussian Splatting (Kerbl et al., 2023), which represents a scene as a set of anisotropic 3D Gaussians rendered by differentiable rasterization. A dynamic Gaussian scene is defined by two distinct components: the time-varying positions of the Gaussians and their physical attributes, namely their spatial extent, opacity, and color.

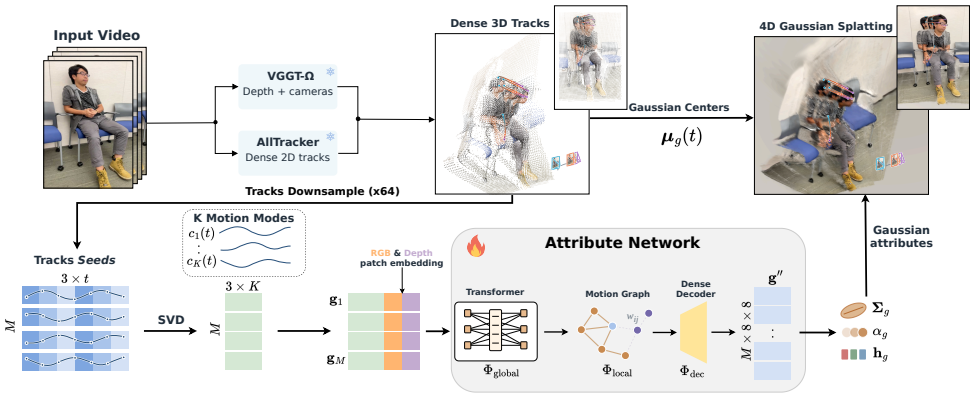

Figure 2: **Overview of TRACKSTO4DGS.** Frozen depth and camera estimates (VGGT-Ω) and dense 2D point tracks (AllTracker) are lifted into dense 3D world-space trajectories, which directly provide the Gaussian centers $\boldsymbol{\mu}_g(t)$. The trajectories are downsampled $64\times$ into M track seeds, and an SVD factorizes them into K shared motion modes $c_1(t), \ldots, c_K(t)$, giving each seed a fixed-size motion descriptor. Each seed token $\mathbf{g}_i$ concatenates this descriptor with RGB and depth patch embeddings. The attribute network fuses the tokens with a global transformer Φ_{global}, refines them over a motion graph Φ_{local}, and a dense decoder Φ_{dec} expands each seed into 8×8 Gaussians with time-invariant covariance $\boldsymbol{\Sigma}_g$, opacity α_g and color $\mathbf{h}_g$. The learned attributes are paired with the fixed trajectories and rendered at each time.

Learning both complex dynamics and appearance from scratch using only rendering losses is severely underconstrained. While prior methods address this ambiguity by using off-the-shelf depth signals as training losses, we take a fundamentally different approach. We observe that recent monocular geometry and dense tracking models are accurate enough that their outputs, lifted to 3D, capture well the scene's motion.

We therefore decouple motion from appearance: First, we run frozen off-the-shelf models for monocular geometry and dense tracking on the input video, obtaining per-frame depth, camera poses, and dense 2D pixel tracks. Relying solely on per-frame geometry would instantiate independent point clouds at each time step, destroying the natural continuity of motion. 2D dense tracks solve this by binding a single Gaussian to the same physical world point across frames. Then, we lift tracked pixels into 3D and use them directly as time-varying Gaussian centers. This leaves our network with a single task: construct continuous, renderable surfaces around points with known motion.

More specifically, given the input video frames $\mathcal{I} = \{\mathbf{I}_t\}_{t=1}^{T}$ and the pre-extracted 3D trajectories, we formulate the dynamic scene $\mathcal{G}(t)$ by fixing the Gaussian centers $\boldsymbol{\mu}_g(t)$ directly to these tracked locations. We then use a feed-forward network to predict the remaining properties. The state of the G Gaussians at time t is defined as:

$$\mathcal{G}(t) = \left\{ \left(\boldsymbol{\mu}_g(t), \boldsymbol{\Sigma}_g, \alpha_g, \mathbf{h}_g \right) \right\}_{g=1}^{G}, \qquad \boldsymbol{\Sigma}_g = \mathbf{Q}(\mathbf{q}_g)\, \mathrm{diag}(\mathbf{s}_g^2) \mathbf{Q}(\mathbf{q}_g)^\top. \tag{1}$$

Here, the network predicts the quaternion $\mathbf{q}_g$, scale $\mathbf{s}_g$, opacity α_g, and spherical-harmonic color $\mathbf{h}_g$.

In the following sections, we detail each stage of this pipeline, summarized in Figure 2. Section 3.1 explains how we construct the dense 3D trajectories that serve as Gaussian centers. Section 3.2 describes the Attribute Network and how it scales efficiently to dense outputs. Finally, Section 3.3 outlines rendering and the training objectives.

3.1 CONSTRUCTING GAUSSIAN TRAJECTORIES

To obtain the 3D trajectories that define our Gaussian centers, we extract and lift dense 2D tracks using frozen foundation models. For each frame, $\mathbf{I}_t$, of the input sequence, a monocular geometry model, VGGT-Ω (Wang et al., 2026), predicts a depth map $\mathbf{D}_t \in \mathbb{R}^{H \times W}$, camera intrinsics $\mathbf{K}_t$, camera-to-world rotation $\mathbf{R}_t$, and camera center $\mathbf{c}_t$. In parallel, a dense tracking model, AllTracker (Harley et al., 2025), extracts 2D point trajectories by tracking pixels from a set of J keyframes bidirectionally across the video window. We denote a 2D track originating from pixel p in keyframe j as $\{\tilde{\mathbf{u}}_{j,p,t}\}_{t=1}^{T}$, where $\tilde{\mathbf{u}}_{j,p,t} \in \mathbb{R}^2$ is its continuous 2D image coordinate at time t.

To construct the 3D Gaussian centers, we lift these 2D tracks into world space using the predicted depth maps and cameras, forming a dense world-space point map $\mathbf{P}_t \in \mathbb{R}^{H\times W\times 3}$ for each frame, where the 3D coordinate $\mathbf{P}_t(x, y) \in \mathbb{R}^3$ at discrete pixel (x, y) is:

$$\mathbf{P}_t(x, y) = \mathbf{R}_t \left(\mathbf{D}_t(x, y)\mathbf{K}_t^{-1}[x, y, 1]^\top\right) + \mathbf{c}_t. \tag{2}$$

We then obtain the corresponding 3D world-space observation of a track at time t, $\tilde{\mathbf{x}}_{j,p,t}$, by bilinearly sampling the point map $\mathbf{P}_t$ at its continuous 2D position $\tilde{\mathbf{u}}_{j,p,t}$.

We resolve missing or invalid observations by applying linear interpolation in world space, and we hold the nearest valid position constant outside the valid tracking interval of each track. Tracks lacking any valid observations are entirely discarded. Complete details regarding scale alignment, coordinate normalization, and validity tests are provided in Appendix B.1. For each retained track from pixel p in keyframe j, we denote its completed 3D trajectory as $\mathcal{X}_{j,p} = \{\tilde{\mathbf{x}}_{j,p,t}\}_{t=1}^{T}$. The full set of valid 3D trajectories across all J keyframes is $\mathcal{X} = \{\mathcal{X}_{j,p}\}$.

Each selected trajectory $\mathcal{X}_{j,p} \in \mathcal{X}$ directly supplies the time-varying center for one Gaussian:

$$\boldsymbol{\mu}_{j,p}(t) = \tilde{\mathbf{x}}_{j,p,t}. \tag{3}$$

The Gaussian centers follow these lifted tracks exactly, which fully determines the scene's motion.

3.2 Predicting Attributes from Seeds

To construct the dynamic scene without per-video optimization, we introduce a feed-forward Attribute Network. Trained across a large dataset of videos, it predicts the remaining attributes of every Gaussian (covariance, opacity, and color) in a single forward pass. Running attention over tens of millions of dense trajectories is prohibitively expensive, so the network instead operates on a sparse set of *seeds*: one representative track per 8×8 keyframe cell, together with the cell's image patch. Processing proceeds in three stages. First, a global transformer aggregates context across the video from the seeds and the scene's shared motion. Second, a local graph network refines these features by exchanging information among nearby seeds that move similarly. Finally, a lightweight decoder expands each seed back to its 64 pixels, predicting attributes for every dense trajectory.

A shared motion descriptor. To form the sparse bottleneck, we partition each keyframe into 8×8 pixel cells and represent each valid cell with a single selected trajectory, which we call a *seed*. Let $\mathbf{x}_{n,t}$ for $n = 1, \ldots, M$ denote the 3D trajectories of the M seeds (the coverage and selection procedure is detailed at the end of this section). To predict attributes, the network needs a description of each seed's motion. Feeding it every seed's full $3 \times T$ trajectory would be wasteful: points on the same surface move coherently, so the motion of all seeds is highly redundant. Following prior work on non-rigid deformation (Kasten et al., 2024), we instead represent each trajectory by its coefficients on a small set of temporal modes shared across the scene:

$$\mathbf{x}_{n,t} \approx \mathbf{b}_{n,0} + \sum_{k=1}^{K-1} \mathbf{b}_{n,k}\, c_{k,t}. \tag{4}$$

Here, $\mathbf{b}_{n,0} \in \mathbb{R}^3$ is the seed's mean position, each shared mode $c_{k,t}$ describes a pattern of motion over time, and the coefficients $\mathbf{b}_{n,k} \in \mathbb{R}^3$ specify how seed n moves along mode k. We obtain the modes by singular value decomposition of the centered seed trajectories, stack them as $\mathbf{C} = [c_{k,t}] \in \mathbb{R}^{(K-1)\times T}$, and use $K = 4$: the mean position and three modes. This factorization serves only as an input descriptor for the network; the Gaussian centers still follow the full dense trajectories of Equation 3.

Local-to-global token aggregation. To predict a seed's attributes, the network must relate its motion to its appearance and local geometry. We therefore take the keyframe where the seed began tracking and extract the RGB colors and log-depth values of its 8×8 cell. We flatten these into patches $\mathbf{p}_n^{\text{rgb}}$ and $\mathbf{p}_n^{\text{depth}}$. Together with the seed's mean position and its motion coefficients $\mathbf{B}_n = [\mathbf{b}_{n,0}, \ldots, \mathbf{b}_{n,K-1}]$, these patches form the seed's initial token:

$$\mathbf{g}_n = \mathrm{Embed}_g\left[\mathbf{p}_n^{\text{rgb}},\, \mathbf{p}_n^{\text{depth}},\, \mathbf{B}_n\right]. \tag{5}$$

Table 1: Novel-view synthesis comparison with state-of-the-art methods on six benchmarks. Protocol details in Section 4. **Best**, second best and third best are highlighted.

	PSNR ↑							SSIM ↑							LPIPS ↓						
Method	ADT	Aria	TUM	DyC	NVIDIA	Pets	Avg	ADT	Aria	TUM	DyC	NVIDIA	Pets	Avg	ADT	Aria	TUM	DyC	NVIDIA	Pets	Avg
4DGT	30.29	26.39	18.26	15.17	16.18	14.46	20.12	0.898	0.817	0.644	0.540	0.404	0.489	0.632	0.075	**0.137**	0.369	0.441	0.547	0.525	0.349
Experts	26.60	24.31	19.21	11.35	11.50	21.39	19.06	0.845	0.771	0.709	0.441	0.279	0.683	0.621	0.106	0.168	**0.191**	0.543	0.605	0.270	0.314
MonST3R	24.34	21.79	18.29	11.03	12.32	16.76	17.42	0.798	0.716	0.676	0.448	0.391	0.627	0.610	0.136	0.223	0.239	0.538	0.484	0.383	0.334
L4GM	23.95	21.93	17.75	12.11	15.47	21.33	18.76	0.744	0.734	0.633	0.517	0.340	0.626	0.599	0.155	0.271	0.303	0.543	0.586	0.353	0.368
MoVieS	27.66	22.80	19.58	15.22	**20.87**	23.34	21.58	0.849	0.779	0.700	0.551	**0.666**	**0.757**	0.717	0.093	0.237	0.249	0.365	**0.172**	**0.242**	0.226
Ours	**31.56**	**27.42**	**22.25**	**17.23**	19.54	**23.70**	**23.62**	**0.935**	**0.878**	**0.817**	**0.626**	0.511	0.692	**0.743**	**0.055**	0.164	0.208	**0.328**	0.259	0.256	**0.212**

While the seed tokens capture local patch information, the network also needs global temporal awareness. We therefore embed the shared temporal modes $\mathbf{C}_{1:K,t}$ from Equation 4, such that each frame t contributes a motion token $\mathbf{m}_t$. A transformer Φ_{global} processes the seed and motion tokens together with four learned register tokens $\mathbf{r}_1, \ldots, \mathbf{r}_4$. We keep only its outputs at the seed positions:

$$\{\mathbf{g}'_n\}_{n=1}^{M} = \Phi_{\text{global}}\big(\{\mathbf{g}_n\}_{n=1}^{M},\ \{\mathbf{m}_t\}_{t=1}^{T},\ \{\mathbf{r}_i\}_{i=1}^{4}\big)_{1:M}. \tag{6}$$

Through attention, each seed combines its own appearance, geometry, and motion coefficients with the behavior of the modes over time. More details are provided in Appendix C.

Local refinement. The global transformer lets every seed attend to every other seed, but it has no explicit preference for seeds on the same surface. Image-grid neighbors are a poor proxy for this: they may lie on different surfaces, and a single surface may be represented by seeds from different keyframes. We therefore build a motion graph in 3D, connecting each seed to its 16 nearest neighbors by mean position $\mathbf{b}_{n,0}$. Each edge is weighted by both spatial proximity and motion similarity:

$$w_{nm} = \exp\left(-\frac{\|\mathbf{b}_{n,0} - \mathbf{b}_{m,0}\|_2^2}{\sigma_p^2} - \frac{\sum_{k=1}^{K-1} \|\mathbf{b}_{n,k} - \mathbf{b}_{m,k}\|_2^2}{\sigma_m^2}\right), \tag{7}$$

where σ_p and σ_m control the spatial and motion bandwidths. Two residual EdgeConv blocks (Wang et al., 2019), denoted Φ_{local}, aggregate the transformer outputs over these weighted edges:

$$\mathbf{g}''_n = \Phi_{\text{local}}\big(\mathbf{g}'_n,\ \{(\mathbf{g}'_m, w_{nm})\}_{m\in\mathcal{N}(n)}\big), \tag{8}$$

where $\mathcal{N}(n)$ is the neighborhood of seed n. As a result, neighbors that are close in 3D but move differently contribute less, which reduces feature mixing across surfaces.

Dense attribute decoding. To achieve high-fidelity rendering, the network must map the enriched sparse features back to the dense pixel level. A point-set decoder Φ_{dec} expands each refined seed feature $\mathbf{g}''_n$ into 64 child features, corresponding to the individual pixels of its original 8×8 cell. Each child feature is then paired with the dense 3D trajectory $\mathcal{X}_{j,p}$ tracked from its pixel (Section 3.1). Parallel heads process these features to predict the Gaussian attributes: a unit quaternion, positive scales, opacity, and spherical-harmonic coefficients. Rather than predicting appearance from scratch, the network outputs a color residual added to the observed keyframe RGB, which anchors the zeroth-order color to the observation and preserves sharp visual details. This design efficiently yields renderable attributes for tens of millions of dense trajectories while restricting all heavy attention and graph operations to the sparse bottleneck.

Coverage selection and seeds. Tracking from multiple keyframes captures surfaces that occlude or become visible over time, but it also produces heavily overlapping trajectories. We eliminate this redundancy using a greedy screen-space coverage test. A dense track is retained only if at least one of its valid observations covers a pixel not already occupied by a previously retained track at that time step. By considering the entire video, a track can still contribute if it covers a newly visible region, even if its other observations overlap existing tracks. This efficiently reduces duplicate coverage without requiring complex 3D surface-identity tests. Further details are provided in Appendix B.2.

3.3 Rendering and Training

We render $\mathcal{G}(t)$ from the predicted camera of frame t using differentiable Gaussian splatting (Kerbl et al., 2023; Ye et al., 2025b). For a randomly sampled set of frames $\mathcal{T}$, the photometric loss is

$$\mathcal{L}_{\text{photo}} = \frac{1}{|\mathcal{T}|} \sum_{t\in\mathcal{T}} \Big[0.8\,\|\hat{\mathbf{I}}_t - \mathbf{I}_t\|_1 + 0.2\,\Big(1 - \text{SSIM}(\hat{\mathbf{I}}_t, \mathbf{I}_t)\Big)\Big], \tag{9}$$

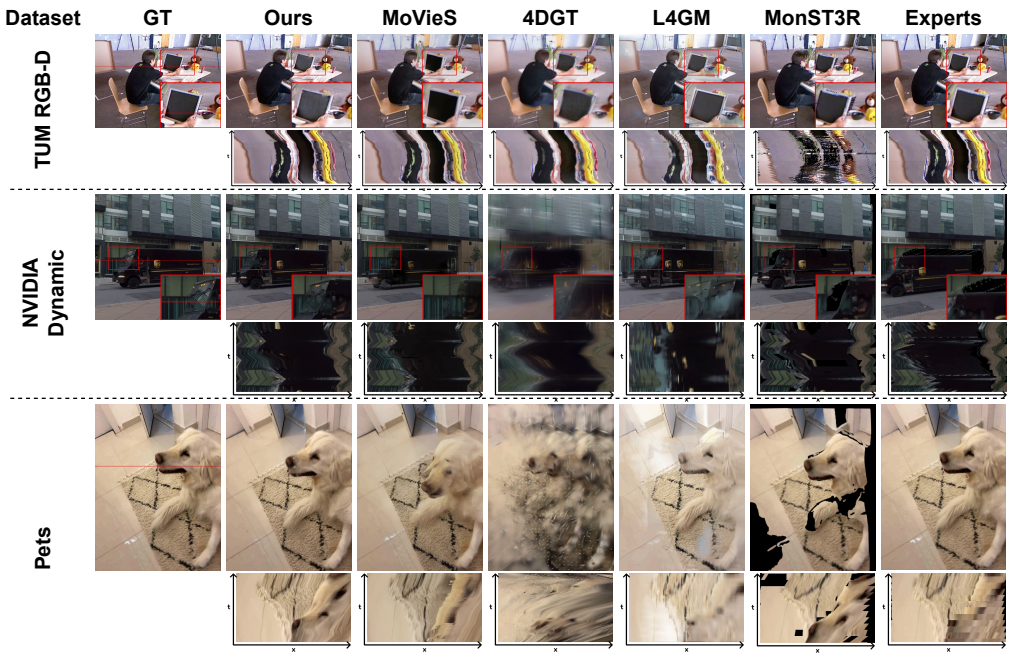

Figure 3: **Qualitative comparison on dynamic scenes.** For each scene we show a single rendered frame, with a zoomed-in, and an x–t slice along the red line marked in GT, which reveals temporal behavior across the whole video. TRACKSTO4DGS yields the best image fidelity and the smoothest motion over time. MonST3R and Experts look plausible in a single frame, but their x–t slices reveal jumpy, stepped video. Videos of all the scenes: `https://tracksto4dgs.github.io`.

We also regularize opacity and scale using the terms in Appendix B.3. At inference, trajectory construction and attribute prediction produce the scene without per-video optimization. At a query time between two input frames, we linearly interpolate each Gaussian's center between its positions at those frames; all other attributes are constant.

4 EXPERIMENTS

We evaluate TRACKSTO4DGS on six common benchmarks for dynamic reconstruction from monocular video, spanning real and synthetic capture, egocentric and handheld cameras, and object-centric and scene-level content. We compare against feed-forward dynamic Gaussian methods under a common protocol, separate temporal interpolation from novel-view synthesis where a benchmark allows it, and ablate the design choices of Section 3.

Datasets. We evaluate on **ADT** (Pan et al., 2023), synthetic egocentric scenes with exact ground truth; egocentric **Aria** recordings from AEA (Lv et al., 2024), Nymeria (Ma et al., 2024), Ego-Exo4D (Grauman et al., 2024), and HOT3D (Banerjee et al., 2024); **TUM-RGBD** (Sturm et al., 2012), handheld indoor sequences with moving people; **DyCheck** (Gao et al., 2022), handheld iPhone captures with held-out validation cameras, scored with covisibility masks; **NVIDIA Dynamic Scenes** (Yoon et al., 2020), handheld scenes from a camera rig evaluated monocularly; and the **pet test set** of TracksTo4D (Kasten et al., 2024), casual videos of cats and dogs under non-rigid deformable motion. We use the same train–test split as 4DGT (Xu et al., 2025): training uses egocentric Aria recordings only, so ADT and Aria are in-domain and the remaining four datasets test transfer, with NVIDIA and pets furthest from the training data in both camera motion and content.

Protocol. We follow the evaluation protocol of 4DGT (Xu et al., 2025). For ADT, Aria, TUM-RGBD, and DyCheck we reuse its published input and held-out frames; for NVIDIA and pets, which 4DGT does not evaluate, we split the frames in the same way. Each recording is divided into fixed windows; within a window, a temporally interleaved half of the frames is given as input and the rest

Table 2: DyCheck results by type of target view. *Int.*: held-out frames along the input camera path (interpolation in time). *NV*: views from separate test cameras placed away from the capture path (wide-baseline novel views). *Comb.*: all targets. **Best**, second best and third best are highlighted.

	PSNR ↑			SSIM ↑			LPIPS ↓		
Method	Int.	NV	Comb.	Int.	NV	Comb.	Int.	NV	Comb.
4DGT	20.36	13.31	15.17	0.589	0.523	0.540	0.252	0.510	0.441
Experts	19.78	8.37	11.35	0.607	0.385	0.441	0.244	0.652	0.543
MonST3R	16.74	9.00	11.03	0.573	0.406	0.448	0.309	0.620	0.538
L4GM	18.95	9.67	12.11	0.543	0.511	0.517	0.342	0.616	0.543
MoVieS	18.43	14.10	15.22	0.581	0.543	0.551	0.276	0.397	0.365
TRACKSTO4DGS (ours)	**22.71**	**15.31**	**17.23**	**0.787**	**0.570**	**0.626**	**0.178**	**0.382**	**0.328**

are held out and scored. On the four 4DGT benchmarks, TRACKSTO4DGS processes 51 rather than 64 inputs per window, but every held-out frame is scored by every method. NVIDIA and pet clips are short, so all methods use a 24-frame window with 12 input frames. Unlike the other baselines, MonST3R and TRACKSTO4DGS do not receive ground-truth cameras, so each held-out camera must be registered to the predicted reconstruction, and even small misregistration sharply reduces PSNR. As is common for pose-free methods (Lin et al., 2021; Ye et al., 2025a), both therefore use the held-out images to localize the held-out cameras. On the four 4DGT benchmarks, TRACKSTO4DGS does so by running VGGT-Ω jointly on input and held-out images; the held-out images reach no other component. On NVIDIA and pets, TRACKSTO4DGS does not use the held-out images at all: VGGT-Ω and AllTracker see only the input frames, and held-out frames are rendered from ground-truth cameras aligned to our reconstruction using the input cameras only. Appendix D.1 details what each method receives.

Baselines. We compare against five baselines, all run from the pretrained checkpoints released by their authors. 4DGT (Xu et al., 2025) is trained on the same egocentric recordings as TRACKSTO4DGS. L4GM (Ren et al., 2024) is an object-centric 4D reconstruction model. MoVieS (Lin et al., 2026) is trained on a more diverse mix of eight static and dynamic datasets spanning synthetic, driving, and internet video. MonST3R (Zhang et al., 2025) predicts per-frame pointmaps of dynamic scenes. The *Experts* baseline of Xu et al. (2025) unprojects the nearest input frame using Depth Anything V2 depth (Yang et al., 2024a), scale-aligned with UniDepth (Piccinelli et al., 2024).

Metrics and implementation. We report PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) as unweighted means over sequences. All TRACKSTO4DGS results come from a single checkpoint with no per-dataset tuning. MoVieS is trained with at most 13 input frames; on ADT, Aria, TUM-RGBD, and DyCheck we give it the full 64-frame windows, as for every other baseline, and report a matched 13-frame setting for both MoVieS and TRACKSTO4DGS in Appendix E.1. On NVIDIA and pets the 12-input windows fall within its training range. Because TRACKSTO4DGS and 4DGT patchify on different grids (8×8 versus 14×14), their native rendering resolutions differ slightly. For a fair comparison, each model renders at its own resolution and we bilinearly resample our renderings to the resolution used by the baselines before scoring, so all metrics are computed on identical pixel grids (Appendix D.2). Baseline setup details are in Appendix D.3.

4.1 RESULTS

Comparison to feed-forward methods. Table 1 compares all methods on the six benchmarks. TRACKSTO4DGS obtains the best average in all three metrics and ranks first or second on every dataset and metric. Two patterns stand out. First, the margin over the baselines widens on the datasets furthest from egocentric capture: 4DGT, trained on the same corpus as TRACKSTO4DGS, degrades sharply on TUM-RGBD, DyCheck, NVIDIA, and pets, whereas TRACKSTO4DGS does not. Fig. 3 shows this failure: 4DGT's novel views of both the NVIDIA and pets scenes dissolve into blur and noise. The difference is what the two models must learn: 4DGT recovers geometry and motion from rendering supervision on egocentric video, and that geometry does not transfer; TRACKSTO4DGS takes them from predictors trained on broad data and only learns appearance. Second, MoVieS is the strongest baseline on the two handheld, object-centric datasets, which is consistent with its training

Table 3: Low-resolution ablations against the control. PSNR (dB); Avg is the unweighted mean over six datasets. † Training-free baselines. ‡ Per-time geometry inputs (Appendix D.4).

	PSNR ↑						
Method	ADT	Aria	TUM	DyC	NVIDIA	Pets	Avg
TRACKSTO4DGS Ours (rank-4)	**30.181**	26.154	**21.490**	**16.610**	**21.484**	24.002	**23.320**
Rank 1 (static conditioning)	29.976	26.058	20.920	16.149	20.980	23.805	22.981
Rank 12	30.122	**26.176**	21.464	16.454	21.368	23.971	23.259
No SVD conditioning/fitting	28.776	25.091	19.820	15.115	19.885	23.006	21.949
Replicate instead of learned upsampler	28.741	25.217	20.563	15.625	21.120	23.510	22.463
DA3 depth and cameras	26.633	23.273	20.287	15.217	20.860	**25.609**	21.980
Independent pointmaps (no tracks)‡	22.613	21.205	17.356	12.121	15.843	18.392	17.922
Fixed splats: constant world radius†	22.958	21.108	16.440	13.948	19.227	20.903	19.097
Fixed splats: one-pixel birth radius†	21.810	20.112	15.872	14.535	19.328	21.150	18.801

mix: internet stereo video, driving, and synthetic dynamic scenes already cover that regime, whereas TRACKSTO4DGS never sees it. Though trained only on egocentric video, TRACKSTO4DGS remains competitive with MoVieS there, leading on Pets in PSNR, and surpasses it on every other dataset. MoVieS, however, struggles with fast-moving subjects (Fig. 3): it renders the truck's windshield transparent and ghosts the dog's head into a doubled face, while TRACKSTO4DGS keeps both intact.

Wide-baseline novel views on DyCheck. The DyCheck benchmark is the most demanding setting among our six benchmarks: its test cameras are placed away from the capture path, so each target view sees the scene from a substantially different viewpoint, and much of it was never observed in the input. Table 2 isolates these views. All methods lose several dB relative to views along the input trajectory, yet TRACKSTO4DGS leads in this setting as well. Absolute numbers nonetheless remain low for all methods: feed-forward monocular novel-view synthesis under such large viewpoint changes is far from solved.

4.2 ABLATIONS

We ablate the components of TRACKSTO4DGS one at a time (Table 3); the full model achieves the highest average PSNR. To test the SVD conditioning, we change its rank from 4 to 1 or 12, or remove it entirely. To test the learned upsampler, we replace it with simple replication of each seed. To test the choice of geometry predictor, we replace the depth and cameras of VGGT-Ω with those of Depth Anything 3 (Lin et al., 2025b). To test the role of tracks, we replace them with independent per-frame pointmaps, which carry no correspondence across time. Finally, to test the learned Gaussian attributes, two training-free variants assign fixed attributes instead: the color of the pixel where each track starts, a fixed opacity, and either a constant or a depth-scaled radius. Appendix D.4 details the protocols.

4.3 LIMITATIONS

TRACKSTO4DGS inherits the failure modes of its frozen predictors: depth errors, lost tracks, and camera drift propagate directly into the trajectories, though it also benefits directly as these models improve. Attributes are constant over each window, so temporally varying appearance is not modeled. We leave learned trajectory corrections and time-varying attributes to future work.

5 CONCLUSION

We presented TRACKSTO4DGS, a feed-forward model that renders dynamic scenes from monocular video by decoupling geometry and motion from appearance: Gaussian trajectories come from frozen tracking and reconstruction models, and a sparse-bottleneck Attribute Network learns only how to render them. Trained solely on unlabeled egocentric video, a single model generalizes across six diverse benchmarks, improving average PSNR by 2.0 dB over the strongest baseline. More broadly, our results suggest that feed-forward 4D reconstruction need not learn motion from scratch, and can instead build on foundation models and improve as they do.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing, for retrieval and discovery of related work, for research execution, and to draft sections of the paper. We have not used generative AI tools for generating synthetic datasets, and proving mathematical claims is not applicable to this work. Specifically, LLMs were used to edit and polish the text; to search for related work; to assist with implementation, experiment scripts, and the preparation of qualitative comparisons (Producing visualizations such as zoomed insets, and x–t slices from our rendered outputs); and to draft initial versions of sections and figure captions, which the authors then revised. We have reviewed all AI-assisted work. All cited works suggested by LLMs were checked by the authors against the original publications. All AI-assisted code was reviewed and tested by the authors, and all reported numbers come from our own evaluation runs. Qualitative examples proposed by AI tools were inspected and selected by the authors, and all AI-drafted text was verified against our experimental results and rewritten where needed. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The main paper specifies the representation, geometric construction, network, renderer, and training objective (Section 3), as well as the datasets, train–test split, windowing and input selection, baselines, and metrics (Section 4). All results come from a single checkpoint with no per-dataset tuning. The appendix provides the remaining method details (Appendix B), architectural dimensions and optimization settings (Appendix C), the resolution alignment across methods (Appendix D.2), baseline configurations (Appendix D.3), and the ablation protocols (Appendix D.4). The supplementary material contains renderings of every evaluation scene for all methods (Appendix A). Code, trained weights, baseline adapters, and evaluation scripts will be publicly released.

REFERENCES

Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Nonrigid structure from motion in trajectory space. In *NeurIPS*, pp. 41–48, 2008.

Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Fan Zhang, Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, Jakob Julian Engel, and Tomas Hodan. Introducing HOT3D: An egocentric dataset for 3D hand and object tracking. *arXiv preprint arXiv:2406.09598*, 2024.

Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *CVPR*, volume 2, pp. 690–696, 2000.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, pp. 19457–19467, 2024.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3d gaussian splatting from sparse multi-view images. In *ECCV*, pp. 370–386, 2024.

Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. TAPIR: Tracking any point with per-frame initialization and temporal refinement. In *ICCV*, pp. 10061–10072, 2023.

Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. In *NeurIPS*, volume 35, pp. 33768–33780, 2022.

Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, et al. Ego-Exo4D: Understanding skilled human activity from first- and third-person perspectives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Adam W. Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *ECCV*, pp. 59–75, 2022.

Adam W. Harley, Yang You, Xinglong Sun, Yang Zheng, Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Pavel Tokmakov, Suya You, Rares Ambrus, Katerina Fragkiadaki, and Leonidas J. Guibas. Alltracker: Efficient dense point tracking at high resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. In *ECCV*, pp. 18–35, 2024.

Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. In *ICCV*, pp. 6013–6022, 2025.

Yoni Kasten, Wuyue Lu, and Haggai Maron. Fast encoder-based 3D from casual videos via point track processing. In *Advances in Neural Information Processing Systems*, 2024.

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.

Jiahui Lei, Yijia Weng, Adam W. Harley, Leonidas Guibas, and Kostas Daniilidis. MoSca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. In *CVPR*, pp. 6165–6177, 2025.

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, pp. 6498–6508, 2021.

Hanxue Liang, Jiawei Ren, Ashkan Mirzaei, Antonio Torralba, Ziwei Liu, Igor Gilitschenski, Sanja Fidler, Cengiz Oztireli, Huan Ling, Zan Gojcic, and Jiahui Huang. Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. *arXiv preprint arXiv:2412.03526*, 2024.

Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. BARF: Bundle-adjusting neural radiance fields. In *ICCV*, 2021.

Chenguo Lin, Yuchen Lin, Panwang Pan, Yifan Yu, Tao Hu, Honglei Yan, Katerina Fragkiadaki, and Yadong Mu. MoVieS: Motion-aware 4d dynamic view synthesis in one second. In *CVPR*, 2026. arXiv:2507.10065.

Chieh Hubert Lin, Zhaoyang Lv, Songyin Wu, Zhen Xu, Thu Nguyen-Phuoc, Hung-Yu Tseng, Julian Straub, Numair Khan, Lei Xiao, Ming-Hsuan Yang, et al. DGS-LRM: Real-time deformable 3d gaussian reconstruction from monocular videos. In *NeurIPS*, 2025a. arXiv:2506.09997.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025b.

Qingming Liu, Yuan Liu, Jiepeng Wang, Xianqiang Lyv, Peng Wang, Wenping Wang, and Junhui Hou. MoDGS: Dynamic gaussian splatting from casually-captured monocular videos with depth priors. In *ICLR*, 2025.

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, pp. 800–809, 2024.

Zhaoyang Lv, Nicholas Charron, Pierre Moulon, Alexander Gamino, Cheng Peng, Chris Sweeney, Edward Miller, Huixuan Tang, Jeff Meissner, Jing Dong, et al. Aria everyday activities dataset. *arXiv preprint arXiv:2402.13349*, 2024.

Lingni Ma, Yuting Ye, Fangzhou Hong, Vladimir Guzov, Yifeng Jiang, Rowan Postyeni, Luis Pesqueira, Alexander Gamino, Vijay Baiyya, Hyo Jin Kim, et al. Nymeria: A massive collection of multimodal egocentric daily motion in the wild. In *Proceedings of the European Conference on Computer Vision*, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Yuheng Carl Ren. Aria digital twin: A new benchmark dataset for egocentric 3D machine perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, pp. 5865–5874, 2021a.

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

Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. UniDepth: Universal monocular metric depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10106–10116, 2024.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, pp. 10318–10327, 2021.

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

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *CVPR*, pp. 5294–5306, 2025a.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Qianqian Wang, Vickie Ye, Hang Gao, Weijia Zeng, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. In *ICCV*, pp. 9660–9672, 2025b.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *CVPR*, pp. 10510–10522, 2025c.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3d vision made easy. In *CVPR*, pp. 20697–20709, 2024.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics*, 38(5), 2019.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *CVPR*, pp. 20310–20320, 2024.

Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. SpatialTracker: Tracking any 2d pixels in 3d space. In *CVPR*, 2024.

Yuxi Xiao, Jianyuan Wang, Nan Xue, Nikita Karaev, Yuri Makarov, Bingyi Kang, Xing Zhu, Hujun Bao, Yujun Shen, and Xiaowei Zhou. SpatialTrackerV2: Advancing 3d point tracking with explicit camera motion. In *ICCV*, pp. 6726–6737, 2025.

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

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. In *Advances in Neural Information Processing Systems*, 2024a.

Yuxue Yang, Lue Fan, Ziqi Shi, Junran Peng, Feng Wang, and Zhaoxiang Zhang. NeoVerse: Enhancing 4d world model with in-the-wild monocular videos. *arXiv preprint arXiv:2601.00393*, 2026.

Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *ICLR*, 2024b. arXiv:2310.10642.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *CVPR*, pp. 20331–20341, 2024c.

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. In *ICLR*, 2025a. arXiv:2410.24207.

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, and Angjoo Kanazawa. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26, 2025b.

Daheng Yin, Isaac Ding, Yili Jin, Jianxin Shi, and Jiangchuan Liu. TrackerSplat: Exploiting point tracking for fast and robust dynamic 3d gaussians reconstruction. In *SIGGRAPH Asia Conference Papers*, 2025. arXiv:2604.02586.

Jae Shin Yoon, Kihwan Kim, Orazio Gallo, Hyun Soo Park, and Jan Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K. Barral, Raia Hadsell, Zoubin Ghahramani, Andrew Zisserman, Junlin Zhang, and Mehdi S. M. Sajjadi. Efficiently reconstructing dynamic scenes one d4rt at a time. In *CVPR*, 2026.

Junyi Zhang, Charles Herrmann, Junhwa Hur, Varun Jampani, Trevor Darrell, Forrester Cole, Deqing Sun, and Ming-Hsuan Yang. MonST3R: A simple approach for estimating geometry in the presence of motion. In *International Conference on Learning Representations*, 2025.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. GS-LRM: Large reconstruction model for 3d gaussian splatting. In *ECCV*, 2024.

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/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

A SUPPLEMENTARY VIDEOS

Videos of all evaluation scenes across the six benchmarks are available on an anonymized web page at `https://tracksto4dgs.github.io`. The same page is included offline in the supplementary material (`index.html`, which opens locally in any browser). For each scene, TRACKSTO4DGS and all baselines are rendered along the same novel camera path and shown side by side, next to the nearest input frame for reference. The scenes are uncurated: every evaluation sequence is included.

B METHOD: ADDITIONAL DETAILS

Throughout, a *parent* is a coarse AllTracker track (one per 8×8 tracker cell), its 64 *children* are the full-resolution convex tracks inside that cell, and dense track (j, p) is the child born at keyframe j and pixel p.

B.1 GEOMETRY GAUGE AND TRAJECTORY COMPLETION

Predicted depth and camera translation can differ by a global scale. At the parent resolution, we robustly triangulate a reference point $\bar{\mathbf{x}}_n$ for each parent from its 2D observations and the predicted cameras. Over the set $\mathcal{A}$ of visible observations of well-triangulated tracks, we estimate the positive depth-to-camera scale

$$\gamma = \mathrm{median}_{(n,t)\in\mathcal{A}} \frac{\mathbf{e}_3^\top \mathbf{R}_t^\top (\bar{\mathbf{x}}_n - \mathbf{c}_t)}{\mathbf{D}_t(\mathbf{u}_{n,t})}, \tag{10}$$

the ratio of triangulated to predicted camera-frame depth, and rescale all depth maps by γ. We then normalize every camera and lifted point with the same similarity transform

$$\mathcal{S}(\mathbf{x}) = \eta\, \mathbf{R}_1^\top (\mathbf{x} - \bar{\mathbf{c}}), \quad \bar{\mathbf{c}} = \frac{1}{T}\sum_t \mathbf{c}_t, \quad \eta = \left(\frac{1}{T} \sum_t \|\mathbf{c}_t - \bar{\mathbf{c}}\|_2 \right)^{-1}. \tag{11}$$

This places the camera centroid at the origin, aligns the world axes with the first camera, and sets the mean camera radius to one. For nearly stationary cameras, where this radius is degenerate, the robust extent of the triangulated point cloud sets the scale instead. Held-out cameras (Section 4) are mapped into this frame by the same transform.

For the motion factorization, each parent's reference position is its robust triangulation when available, and otherwise the median of its valid depth-lifted positions. Dense depth is aligned to this normalized geometry by a single positive scale (no shift), fitted by least squares at the valid parent birth samples. The full-resolution child tracks are then unprojected with the aligned depth. A sample serves as an anchor only if the tracker marks it visible and its position and sampled depth are finite, positive, and in bounds. We linearly interpolate missing samples between anchors in world space and hold the nearest anchor outside that interval. A trajectory with no valid anchor is discarded.

B.2 TEMPORAL COVERAGE OWNERSHIP

For parent n, the tracker supplies visibility $v_{n,t}$ and two confidence values $p_{n,t}^{(1)}$ and $p_{n,t}^{(2)}$. We define the per-frame and per-track reliabilities

$$r_{n,t} = v_{n,t}\, p_{n,t}^{(1)}\, p_{n,t}^{(2)}, \qquad \rho_n = \frac{1}{T}\sum_t r_{n,t}. \tag{12}$$

Parents are split at the median of ρ_n over all parents in the window. Tracks strictly above the median are processed first across keyframes, in keyframe birth order; all remaining tracks follow in a second birth-ordered pass. Let $\mathcal{O}_t$ denote the screen locations already owned at time t, and let $\nu_{j,p,t} = 1$ indicate that dense track (j, p) is visible, finite, and in bounds at t. A child is kept if it reaches an unowned location at some valid time:

$$k_{j,p} = \mathbb{1}\left[\exists t :\ \nu_{j,p,t} = 1 \ \wedge \ \min_{\mathbf{y}\in\mathcal{O}_t} \|\tilde{\mathbf{u}}_{j,p,t} - \mathbf{y}\|_\infty > \varepsilon \right], \qquad \varepsilon = 0.25 \text{ px}. \tag{13}$$

The test is evaluated on a supersampled occupancy grid. Each kept child adds its valid observations to $\mathcal{O}_t$; occluded observations claim no coverage. Pooling the child decisions within each 8×8 cell gives the parent mask used for the SVD and point-set decoding.

B.3 TOKEN INPUTS, DECODER, AND OBJECTIVE

Before 8×8 patchification, images are padded to the tracker grid so that boundary cells preserve exact tracker-to-pixel alignment. Each Gaussian-parent token has 268 scalar inputs: a flattened 8×8 RGB patch (192), a flattened log-depth patch (64), the 3D anchor (3), and the $K - 1 = 3$ spatial

bases (9). The separate 3D anchor intentionally duplicates the zeroth (static) spatial basis. A learned projection maps this vector to the transformer width. Spatial, keyframe, temporal, and token-type embeddings are added to the corresponding token families.

The point-set decoder implements its $64\times$ expansion with a transposed 1D convolution of kernel size and stride 64 and a learned child-identity embedding. Equivalently, parent feature $\mathbf{f}_i$ yields, for child slot r,

$$\mathbf{y}_{i,r} = \mathbf{W}_r \mathbf{f}_i + \mathbf{e}_r^{\text{child}}, \qquad r \in \{1, \dots, 64\}. \tag{14}$$

In addition to the photometric loss of Equation 9, we regularize opacity and scale over all G Gaussians:

$$\begin{aligned} \mathcal{L}_\alpha &= \frac{1}{G} \sum_g \alpha_g, \\ \mathcal{L}_s &= \frac{1}{G} \sum_g \max\left(0, \max_{d \in \{x,y,z\}} \log s_{g,d} + 1\right), \\ \mathcal{L} &= \mathcal{L}_{\text{photo}} + 10^{-4}\mathcal{L}_\alpha + 10^{-2}\mathcal{L}_s. \end{aligned} \tag{15}$$

$\mathcal{L}_s$ penalizes only Gaussians whose largest axis exceeds e^{-1} in the normalized units of Appendix B.1. Optimization settings are given in Appendix C.

C Implementation Details

Architecture. The reference configuration processes $T = 51$ frames and seeds bidirectional tracks every 10 frames, including the final frame, yielding $J = 6$ keyframes. It uses $K = 4$ trajectory components (one static component and 3 dynamic modes), an eight-layer transformer of width 512 with eight attention heads and four register tokens, a 16-nearest-neighbor motion graph with two graph-refinement blocks, and 64 Gaussian children per retained parent. TracksTo4DGS has 45.5M trainable parameters; VGGT-Omega and AllTracker are kept frozen.

Training. We train for 100K iterations at a maximum image side of 320, then fine-tune for 20K iterations at 512×512 with a freshly initialized optimizer. Both stages use Adam with learning rate 10^{-4}, a 1,000-step linear warmup, cosine decay to 10^{-5}, and global gradient clipping at 1.0. Training runs on 16 NVIDIA H100 (80GB) GPUs with one 51-frame sample per GPU (an effective batch size of 16) in FP32 precision. A single final checkpoint is used for all benchmarks, without per-dataset tuning.

Code release. Code, trained weights, baseline adapters, and evaluation scripts will be made publicly available.

D Evaluation Protocol

D.1 Benchmarks and Metrics

We evaluate on the 4DGT suite (Xu et al., 2025) (4 ADT, 7 Aria, 3 TUM, and 6 DyCheck sequences), all 8 NVIDIA Dynamic Scenes sequences, and 21 pet clips. None of them appear in training. On ADT, Aria, TUM, and DyCheck, TracksTo4DGS takes 51 interleaved inputs per 101-frame window; following 4DGT, baselines receive 64 inputs per 128-frame window. All methods are scored on the same held-out targets. On NVIDIA and Pets, all methods use 12 inputs per 24-frame window and are scored on the held-out odd frames. DyCheck additionally evaluates held-out camera views. We report PSNR, SSIM, and LPIPS, using DyCheck covisibility masks for held-out views and full images otherwise. Metrics are computed on the fixed per-dataset grids of Appendix D.2. Scenes are weighted equally within each dataset, and the overall average weights the six datasets equally. Videos of every evaluation sequence are provided in the supplementary material (Appendix A).

Held-out cameras. Evaluating a pose-free method on novel views involves a trade-off. 4DGT, MoVieS, L4GM, and Experts receive ground-truth cameras for both input and held-out frames: their held-out views are aligned with their reconstruction by construction, and on several benchmarks these

ground-truth cameras are themselves estimated from the full recording, held-out frames included. MonST3R and TRACKSTO4DGS predict their own cameras, so each held-out camera must be registered to the predicted reconstruction; unlike point-cloud metrics, PSNR drops sharply under even small camera misalignment, even when the reconstruction itself is accurate. As is common for pose-free methods (Lin et al., 2021; Ye et al., 2025a), both use the held-out images to localize the held-out cameras. MonST3R does so by PnP followed by photometric refinement against the held-out image. On ADT, Aria, TUM, and DyCheck, TRACKSTO4DGS runs VGGT-Ω jointly on the input and held-out images of each window (on DyCheck, in chunks that include the held-out camera views), which places all cameras in one shared frame. We render held-out frames from the predicted held-out cameras and build trajectories from the input-frame depth and cameras. The held-out images reach no other component: AllTracker and the attribute network never see them, and they contribute no color. Because VGGT-Ω is a joint multi-view model, the held-out images may affect its predicted depth and cameras. Since most baselines already receive ground-truth cameras, they still have advantage with in camera reconstructions, and any resulting advantage over them is limited to depth. NVIDIA and Pets test TRACKSTO4DGS without this effect, since there it uses no held-out images (below): on both, it still ranks first or second in every metric and outperforms 4DGT by 3.4 and 9.2 dB PSNR, although 4DGT receives ground-truth cameras (Table 1).

NVIDIA and Pets: no held-out images. On NVIDIA and Pets, TRACKSTO4DGS evaluates without this trade-off and uses no held-out image in any component. VGGT-Ω, AllTracker, and the attribute network see only the input frames. Held-out frames are rendered from the same ground-truth cameras the baselines receive, mapped into the normalized frame of Equation 11 by a similarity transform fitted between the ground-truth and predicted input cameras only. The held-out images are used solely to compute the metrics.

D.2 NATIVE AND EVALUATION RESOLUTIONS

4DGT patchifies with the 14×14 grid of its DINOv2 encoder (Oquab et al., 2024), so its native resolutions are multiples of 14 per side, whereas TRACKSTO4DGS renders at multiples of 16. Preserving the source aspect ratio under these constraints yields slightly different native grids for the two models. We compute all metrics on the 4DGT-native grid (Table 4). Our renderings are resampled bilinearly to this grid, while 4DGT renderings are scored directly at their native resolution. On Pets, the evaluation grid is larger than our rendering grid, so our renderings are upsampled; if anything, this disadvantages TRACKSTO4DGS.

Table 4: Rendering grid of TRACKSTO4DGS, evaluation grid used for all main results (4DGT-native), and evaluation grid of the low-resolution ablations (Appendix D.4). All grids are height×width.

Benchmark	TRACKSTO4DGS render	Evaluation	Ablation
ADT, Aria	512×512	504×504	320×320
TUM	528×704	504×672	256×320
DyCheck	640×448	560×392	320×256
NVIDIA	512×512	504×504	224×448
Pets	576×432	672×504	320×256

D.3 BASELINES

4DGT and MoVieS use their released weights with the supplied input and target cameras. For MoVieS, cameras are normalized with context-only VGGT, and images are resized to a maximum side of 518. 4DGT's adapters for MonST3R, Experts, and L4GM are unpublished, so we reimplemented them from public code. MonST3R runs its native global alignment (300 iterations), interpolates geometry in time with SEA-RAFT flow, and localizes each target camera by PnP followed by camera-only photometric refinement. Experts scale-aligns Depth Anything V2-Large depth to UniDepth V2-Large, adds StableNormal normals, and splats the nearest input frame into the target view. L4GM-Calibrated runs L4GM on full frames at 256×256, maps its canonical Gaussians to the input cameras with an estimated scale, and renders from the target cameras. Only MonST3R and TRACKSTO4DGS predict their own cameras; held-out camera localization is described in Appendix D.1.

D.4 Low-Resolution Ablation Protocol

To keep the ablations affordable, all variants in Table 3 are trained and evaluated at reduced resolution. The control is retrained from scratch for 100K updates on eight GPUs with one sample per GPU. Like the main model, it uses rank-4 SVD conditioning ($K = 4$); it keeps the learned point-set expansion. All learned variants share the control's seed and training settings, including a cosine learning-rate schedule defined over 1M steps, and differ only in the intervention described below.

Every variant is scored on the same target frames at the ablation grids of Table 4. Table 3 reports ADT, Aria, TUM, DyCheck (DyC), NVIDIA, and Pets separately; its NVIDIA column averages the seven scenes other than *DynamicFace*. Each dataset score is an unweighted mean over its scenes, and Avg is the unweighted mean of the six dataset scores, computed before rounding.

Conditioning interventions. The rank-1 variant keeps only the static SVD component as conditioning. The no-SVD variant skips SVD fitting, which removes the SVD-derived spatial inputs, the coefficient tokens, and the motion-graph affinity; its graph is built from mean valid-frame track positions instead. In both variants, rendering still uses the time-varying dense tracked centers, so neither makes the scene static: they remove motion information only from the network's conditioning.

Independent pointmaps. This variant replaces persistent trajectories with an independent pointmap at every time. The same six keyframes still provide appearance features to the attribute network. At every other time, the pointmap supplies 3D positions and a fixed time coordinate, and the decoder predicts colors for a separate Gaussian cloud with no identity shared across frames. Because each pointmap is built from the image at its own time, the geometry provider sees the images at held-out times in the temporal-interpolation benchmarks. On DyCheck it uses the training-view image at each time and never the held-out novel view. The track-based gauge of Appendix B.1 is replaced by the provider's joint depth–camera gauge with track-free normalization. The variant therefore compares scene representations under full-time geometry input; it is not a held-out-target test and does not isolate identity alone.

DA3 depth and cameras. Source depth and source cameras are both replaced by DA3 predictions. On the 4DGT suite and DyCheck, target cameras are also predicted by DA3 and aligned to the source gauge through shared source anchors. NVIDIA and Pets keep the ground-truth target cameras with similarity alignment, as in the main protocol. The variant therefore measures the combined effect of the geometry provider.

Training-free fixed splats. Two heuristic baselines keep the control's dense moving centers and coverage rule but replace the learned attribute network entirely. They use isotropic scales, identity rotations, opacity $\mathrm{sigmoid}(2)$, birth-pixel RGB, and no view-dependent color. They differ only in the radius: one sets every world-space scale to $\exp(-3)$, and the other sets each Gaussian's scale to a one-pixel footprint in its birth view (from birth depth and focal length), held fixed over time. No trained weights are used.

E Additional Results

E.1 MoVieS in Its Native Setting

MoVieS is trained with at most 13 input frames, so evaluating it on 64-frame windows may put it at a disadvantage. To rule this out, Table 5 evaluates both methods on 13-frame chunks of the same windows, scored on common targets. MoVieS improves in this setting, as expected. TracksTo4DGS also improves, although it was not trained with this input configuration, and remains ahead on the suite average in all three metrics.

E.2 Runtime

Table 6 reports the wall-clock time to reconstruct and render one window on a single NVIDIA H100 (80GB) GPU, measured on one scene per benchmark. Timings include all per-window computation (tracking, depth and camera estimation, reconstruction, rendering, and I/O) and exclude model

Table 5: Matched-input comparison with MoVieS on the 4DGT evaluation suite: both methods reconstruct from 13-input chunks and are scored on the same targets. Avg weights all 20 sequences equally. Best in **bold**.

	PSNR ↑					SSIM ↑					LPIPS ↓				
Method	ADT	Aria	TUM	DyC	Avg	ADT	Aria	TUM	DyC	Avg	ADT	Aria	TUM	DyC	Avg
MoVieS	32.46	27.62	20.30	13.70	23.31	0.904	0.856	0.765	0.525	0.753	**0.042**	**0.138**	0.185	0.419	0.210
Ours	**32.68**	**29.62**	**25.15**	**16.31**	**25.57**	**0.937**	**0.881**	**0.860**	**0.606**	**0.806**	0.051	0.147	**0.157**	**0.348**	**0.190**

Table 6: Wall-clock time (seconds) to reconstruct and render one window on an NVIDIA H100, excluding model loading. One scene per benchmark (DyCheck: *apple*; NVIDIA: *Balloon1*); single run, no warm-up.

Method	DyCheck	NVIDIA	Pets
4DGT	70.7	37.2	37.7
Experts	49.2	12.2	13.3
MonST3R	829.3	118.4	112.9
L4GM	42.2	15.9	14.6
MoVieS	48.6	3.6	3.8
TRACKSTO4DGS	54.9	36.3	33.9

loading and data staging. Each entry is a single run without warm-up, so the numbers indicate the cost of each pipeline rather than a precise speed ranking.

On NVIDIA and Pets, all methods process identical windows, with 12 input frames and 11 target views. On DyCheck, window lengths follow each method's evaluation setup (Appendix D.1). NVIDIA and Pets share the same window size across all methods for direct comparison. On these two benchmarks, TracksTo4DGS reconstructs a window in about 35 seconds, comparable to 4DGT and about 3× faster than the optimization-based MonST3R. The lighter pipelines of Experts, L4GM, and MoVieS are faster.