# GAUSSTRACE: PERSISTENT 3D TRAJECTORIES FOR DYNAMIC GAUSSIAN RECONSTRUCTION FROM MONOCULAR VIDEO

**Anonymous authors**
Paper under double-blind review

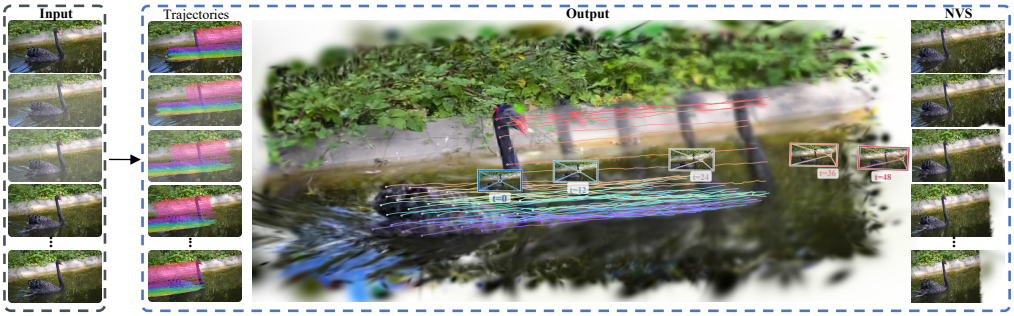


Figure 1: GaussTrace refines persistent 3D trajectories from a single feed-forward prediction, achieving high-quality dynamic Gaussian reconstruction with efficient 2–4 minute optimization.

## ABSTRACT

Reconstructing dynamic 3D scenes from casually captured monocular video requires temporally coherent motion under sparse, ambiguous observations, yet existing 4D Gaussian Splatting (4DGS) pipelines rely on fragmented preprocessing or expensive scene-specific optimization. We present **GaussTrace**, a framework built around persistent 3D trajectories that converts a single feed-forward 4D prediction into unified priors for dynamic Gaussian reconstruction. *Trajectory-Basis Motion Modeling* (TBMM) groups similar trajectory bases and learns shared translation corrections with local residuals, while *Motion-Aware Routing and Temporal Coreset* (MRTC) combines tri-state motion routing with motion-aware time selection to allocate motion freedom and focus refinement on representative observations. *Progressive Gaussian Optimization* (PGO) separates scene stabilization from motion refinement before jointly optimizing Gaussian attributes and motion corrections. Experiments on NVIDIA Dynamic Scenes, iPhone DyCheck, and DAVIS show strong reconstruction quality, sharper dynamic boundaries, and substantially more efficient optimization than representative monocular 4DGS systems.

## 1 INTRODUCTION

Real-time reconstruction of dynamic 3D scenes from video streams, commonly referred to as 4D reconstruction, is crucial for applications including robotics (Huang et al., 2023; 2025), augmented and virtual reality (AR/VR) (Carmigniani & Furht, 2011; Cruz-Neira et al., 1992), and autonomous driving (Sun et al., 2020). For monocular input, 4D Gaussian Splatting (4DGS) provides a particularly appealing representation by extending explicit 3D Gaussian Splatting (3DGS) primitives (Kerbl et al., 2023) to model time-varying scenes while retaining efficient rasterization and high-quality novel-view synthesis (Yang et al., 2024c; Wu et al., 2024). The central challenge, nevertheless, is to enforce temporally consistent motion rather than fit each frame as an independent 3D scene.

Existing monocular 4DGS methods broadly follow two paradigms: per-scene optimization and feed-forward prediction. Among optimization-based methods, Shape of Motion (Wang et al., 2025) represents each point trajectory as a linear combination of compact SE(3) motion bases to recover



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.

coherent long-range 3D motion. MoSca (Lei et al., 2025) constructs a 4D motion scaffold that compactly encodes scene deformation and anchors Gaussians for joint geometry and appearance reconstruction. MoDGS (Liu et al., 2025) leverages monocular depth priors, 3D-aware initialization, and robust depth supervision to stabilize reconstruction under weak multi-view constraints. SplineGS (Park et al., 2025) models continuous Gaussian trajectories with cubic Hermite splines and adaptively prunes control points according to motion complexity. Despite these advances, per-scene optimization makes reconstruction slow. In the feed-forward direction, StreamSplat (Wu et al., 2026) predicts online dynamic Gaussians from uncalibrated video streams using probabilistic Gaussian sampling, bidirectional deformation, and adaptive Gaussian fusion. It substantially accelerates reconstruction, but its single-pass prediction generally yields lower reconstruction quality than per-scene optimization.

Recent feed-forward 4D reconstruction models, including D4RT (Zhang et al., 2026a), 4RC (Luo et al., 2026), Track4World (Lu et al., 2026), and OmniX (Jiang et al., 2026), directly recover geometry, camera parameters, spatiotemporal correspondences, or dense 3D trajectories from monocular video. Inspired by these advances, we seek to leverage the coordinate-aligned geometry, camera estimates, dense 3D trajectories, and long-range motion context encoded by feed-forward 4D models as unified priors for dynamic Gaussian reconstruction. This leads to a central question: *Can we effectively combine feed-forward 4D prediction with per-scene optimization to build an accurate and efficient monocular 4DGS system?* However, effectively combining the two paradigms raises two key challenges: (1) how to organize corrections to dense feed-forward trajectories through a compact group-shared branch without sacrificing point-wise flexibility; and (2) how to reliably distinguish static and dynamic regions under monocular ambiguity while accounting for uncertain motion, so that each Gaussian receives an appropriate degree of motion freedom.

To address these challenges, we present **GaussTrace**, a framework built around persistent 3D trajectories that constructs an explicit dynamic Gaussian scene from a single feed-forward 4D prediction. First, *Trajectory-Basis Motion Modeling* (TBMM) treats the predicted trajectories and trajectory bases as a fixed motion scaffold, groups bases with similar full-sequence motion, and learns a compact group-wise translation-correction branch complemented by point-wise residuals. Second, *Motion-Aware Routing and Temporal Coreset* (MRTC) combines tri-state motion-confidence routing with motion-aware time selection: a smooth gate assigns static, transitional, or dynamic motion freedom to each Gaussian, while displacement-and-velocity descriptors select representative times for efficient long-sequence refinement. Third, *Progressive Gaussian Optimization* (PGO) first stabilizes Gaussian coverage, appearance, cameras, and topology, and then fixes topology to jointly refine Gaussian attributes and shared and local trajectory corrections under stage-specific objectives. Experiments on NVIDIA Dynamic Scenes and iPhone DyCheck (Yoon et al., 2020; Gao et al., 2022) show that GaussTrace delivers strong and consistent reconstruction quality across diverse capture settings while requiring only 2–4 minutes of scene-specific optimization.

Our contributions are threefold:

- We introduce a monocular 4DGS framework that compiles persistent 3D trajectories predicted by a feed-forward spatiotemporal query model into unified priors for dynamic Gaussian reconstruction.
- We propose TBMM and MRTC to preserve long-range trajectory structure, organize learned corrections into group-shared and point-wise paths, assign motion freedom under monocular ambiguity, and concentrate long-video optimization on representative times.
- We develop PGO to separate scene stabilization from motion refinement, enabling strong reconstruction quality together with efficient optimization and rendering.

## 2 Related Work

### 2.1 Dynamic Neural Radiance Fields

Dynamic NeRF methods extend static NeRF (Mildenhall et al., 2022) with scene flow, canonical deformation, time-conditioned radiance, or spatiotemporal factorization; representative examples include D-NeRF, Nerfies, and HyperNeRF (Li et al., 2021; Pumarola et al., 2021; Park et al., 2021a;b; Wang et al., 2021; Xian et al., 2021). For casually captured monocular videos, RoDynRF, MonoN-

eRF, DynIBaR, MoBluRF, and CTNeRF improve robustness through temporal observations, camera handling, or learned priors (Liu et al., 2023; Tian et al., 2023; Li et al., 2023; Bui et al., 2025; Miao et al., 2024). Casual-FVS instead synthesizes casual-video views by warping information from neighboring frames (Lee et al., 2024b). These implicit representations motivate an explicit dynamic representation under monocular ambiguity.

### 2.2 Dynamic 3D Gaussian Splatting

Dynamic Gaussian methods extend 3DGS (Kerbl et al., 2023) through deformation fields, space-time features, polynomial trajectories, particles, or compact motion factors; 4D Gaussian Splatting (4DGS) (Wu et al., 2024) and Deformable 3D Gaussians (Yang et al., 2024c) are distinct examples alongside other variants (Liang et al., 2025; Li et al., 2024; Lin et al., 2024; Lee et al., 2024a; Kratimenos et al., 2024; Huang et al., 2024; Yang et al., 2024b; Duan et al., 2024; Luiten et al., 2024). Dynamic Gaussian Marbles targets robust monocular reconstruction with an explicit Gaussian scene (Stearns et al., 2024). Recent monocular systems structure motion with shared 3D bases, scaffolds, depth priors, adaptive spline trajectories, or continuous $\mathrm{SE}(3)$ B-splinegs (Wang et al., 2025; Lei et al., 2025; Liu et al., 2025; Park et al., 2025; Zhang et al., 2026b). Their pipelines commonly combine camera estimation, monocular depth, dynamic masks, optical flow, and long-range 2D tracking. SpeeDe3DGS further studies efficient dynamic Gaussian modeling through pruning and motion grouping (Tu et al., 2026). StreamSplat directly predicts dynamic Gaussians from uncalibrated video streams in an online feed-forward setting (Wu et al., 2026). In contrast, GaussTrace receives mutually aligned depth-ray geometry, dense 3D point trajectories, camera parameters, and trajectory transformation bases from one video model, then restricts optimization to residual corrections supported by the video.

### 2.3 Feed-forward Video Reconstruction and Tracking

Feed-forward geometric models such as DUSt3R, MonST3R, and Align3R directly infer video geometry for dynamic-reconstruction initialization (Wang et al., 2024; Zhang et al., 2025; Lu et al., 2025). Long-range trackers such as CoTracker provide video correspondences for downstream reconstruction and tracking (Karaev et al., 2024). Recent query and world-tracking models, including D4RT, 4RC, Track4World, OmniX, ST4RTrack, SpatialTrackerV2, and 4DGT, predict geometry, cameras, or persistent 3D trajectories directly from video (Zhang et al., 2026a; Luo et al., 2026; Lu et al., 2026; Jiang et al., 2026; Feng et al., 2025; Xiao et al., 2025; Xu et al., 2025). Unlike fully feed-forward renderers, GaussTrace uses OmniX as a unified prior and retains short constrained optimization to adapt canonical Gaussian attributes and correct residual camera–motion errors.

## 3 Preliminaries

### 3.1 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) represents a static scene with anisotropic Gaussian primitives $\mathcal{G} = \{G_i\}_{i=1}^{N}$. Each $G_i$ has center $\mu_i \in \mathbb{R}^3$, covariance $\Sigma_i = R_i S_i S_i^\top R_i^\top$, opacity $\alpha_i$, and color $c_i$, where $R_i$ and $S_i$ denote rotation and scale. The spatial influence of $G_i$ at a 3D location $\mathbf{x}$ is given by

$$G_i(\mathbf{x}) = \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i)\right]. \tag{1}$$

After projection and depth sorting, alpha compositing yields

$$\hat{C} = \sum_{i\in\mathcal{N}} c_i \widetilde{\alpha}_i \prod_{j<i}(1-\widetilde{\alpha}_j), \tag{2}$$

where $\mathcal{N}$ contains contributing splats and $\widetilde{\alpha}_i$ combines learned opacity with the projected 2D Gaussian value at the pixel.

**Extension to dynamic scenes.** Static 3DGS shares $\mathcal{G}$ across observations. Dynamic methods introduce time $t$ and construct

$$\mathcal{G}_t = \{G_{t,i} = (\mu_{t,i}, \Sigma_{t,i}, \alpha_{t,i}, c_{t,i})\}_{i=1}^{N}, \tag{3}$$

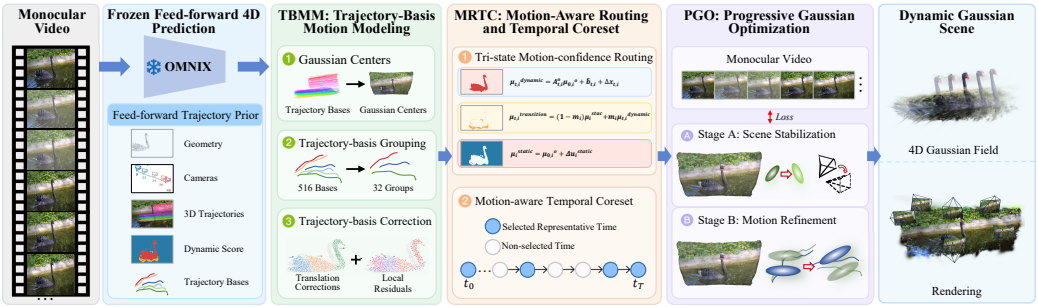


Figure 2: GaussTrace pipeline. Frozen OmniX priors initialize geometry, cameras, trajectories, scores, and bases; PGO first freezes topology, after which TBMM corrects grouped trajectories and MRTC applies tri-state routing and selects representative times for joint refinement.

by making selected attributes time-dependent, typically through canonical-to-time deformation or explicit trajectories (Luiten et al., 2024; Yang et al., 2024c; Wu et al., 2024). Each $\mathcal{G}_t$ uses the static-3DGS projection and alpha compositing.

# 4 Method

## 4.1 Overview

Figure 2 shows GaussTrace, which combines feed-forward 4D priors with per-scene residual optimization for monocular dynamic Gaussian reconstruction. A frozen pass initializes aligned geometry, cameras, trajectories, dynamic scores, and motion bases. PGO first stabilizes Gaussian coverage, appearance, cameras, and topology; after topology is fixed, TBMM groups bases and learns shared translation corrections plus local residuals, while MRTC assigns motion freedom and selects representative times for joint refinement. Together, the modules preserve long-range motion while correcting prediction errors from RGB supervision.

## 4.2 Trajectory-Basis Motion Modeling

Learning independent time-varying motion for every Gaussian from monocular RGB is underconstrained and inefficient, while an unrefined feed-forward trajectory retains geometric errors. *Trajectory-Basis Motion Modeling* (TBMM) treats the feed-forward trajectory field as a fixed scaffold and organizes its correction into group-shared translations and point-wise residuals. It comprises (1) a *Feed-forward trajectory prior* decoding persistent Gaussian centers from predicted bases; (2) *Trajectory-basis grouping* of bases with similar full-sequence motion; and (3) *Trajectory-basis correction* with shared translations and point-wise residuals.

*Feed-forward trajectory prior.* We choose OmniX (Jiang et al., 2026) because its strong 4D-reconstruction and long-range-tracking performance provides reliable motion initialization. One forward pass jointly predicts geometry, cameras, persistent 3D trajectories, dynamic scores, and trajectory bases in a common coordinate system, avoiding heterogeneous-estimator alignment while preserving long-range identities. Given a monocular video $\mathcal{I} = \{I_t\}_{t=0}^{T-1}$, a frozen pass yields

$$\left\{C_t^o, \mu_{t,i}^o, q_{t,i}, S_t, \tau_{t,k}^o, a_{i,k}\right\}. \tag{4}$$

Here $c = 0$ is the canonical source timestamp; $C_t^o = (R_t^o, t_t^o, K_t^o)$, $\mu_{t,i}^o \in \mathbb{R}^3$, $q_{t,i} \in [0,1]$, and $S_t$ denote camera pose, persistent-point center, confidence, and dynamic-score map. OmniX also provides $K = 516$ bases $\tau_{t,k}^o = (r_{t,k}^o, b_{t,k}^o) \in \mathbb{R}^6 \times \mathbb{R}^3$ and sparse normalized coefficients $a_{i,k} \in [0,1]$ with $\sum_k a_{i,k} = 1$. Taking $\mu_{0,i}^o$ as the reference center, we decode

$$\tau_{t,i}^o = \sum_{k=1}^{K} a_{i,k}\tau_{t,k}^o \longmapsto (A_{t,i}^o, b_{t,i}^o), \qquad \mu_{t,i}^o = A_{t,i}^o \mu_{0,i}^o + b_{t,i}^o. \tag{5}$$

where the 6D rotation of $\tau^o_{t,i}$ maps to $A^o_{t,i} \in \mathrm{SO}(3)$ and $b^o_{t,i} \in \mathbb{R}^3$ is its translation. Feed-forward outputs remain fixed; after topology stabilizes, $a_{i,k}$ is constructed once for the current Gaussian set and vanishes outside point $i$'s sparse basis support.

*Trajectory-basis grouping.* Several of the 516 bases can encode similar full-sequence motion, making an independent correction for every basis unnecessarily redundant. We concatenate source-relative rotation-matrix and translation trajectories into a full-sequence descriptor, separately normalized by their global standard deviations over active bases and timestamps. After Stage-A topology stabilization, the fixed tri-state routing in Section 4.3 defines the dynamic Gaussian set $\mathcal{D}$; its basis support weight is $w_k = \sum_{i \in \mathcal{D}} a_{i,k}$. We apply one-time deterministic weighted $k$-means (MacQueen, 1967) to this descriptor augmented with frozen token semantics, using $w_k$ to produce $G = 32$ groups. We denote the resulting assignment of basis $k$ by $\pi_k \in \{1, \ldots, G\}$. This grouping uses only frozen outputs and sparse interpolation support; assignments stay fixed and receive no RGB-loss gradients.

*Trajectory-basis correction.* Using these fixed assignments, each group $g$ receives a learnable source-relative translation correction $\Delta\tau_{t,g} \in \mathbb{R}^3$ at time $t$. For Gaussian $i$, the fixed weight of group $g$ is $\rho_{i,g} = \sum_{k:\, \pi_k = g} a_{i,k}$, with $\sum_{g=1}^{G} \rho_{i,g} = 1$. Writing the blended prior as $\tau^o_{t,i} = (r^o_{t,i}, b^o_{t,i})$, its translation-corrected trajectory transform is

$$\hat{\tau}_{t,i} = \left( r^o_{t,i},\ b^o_{t,i} + \sum_{g=1}^{G} \rho_{i,g} \Delta\tau_{t,g} \right) \longmapsto \left( A^o_{t,i}, \hat{b}_{t,i} \right). \tag{6}$$

Thus, $\hat{\tau}_{t,i}$ preserves prior rotation and updates only translation, compressing the group-shared correction branch from $T \times 516 \times 3$ to $T \times 32 \times 3$. For $N$ active Gaussians, this changes the shared-correction decode and its backward reduction from an $N \times 516$ mixture to an $N \times 32$ mixture, reducing the computation and coefficient-buffer traffic of this branch by $16.1\times$ while leaving the dense local residual state intact. To retain per-Gaussian correction freedom, we additionally introduce a point-wise trajectory residual $\Delta x_{t,i} \in \mathbb{R}^3$. Applying $\hat{\tau}_{t,i}$ to the reference center and adding this residual gives the dynamic center

$$\mu^{\mathrm{dynamic}}_{t,i} = A^o_{t,i} \mu^o_{0,i} + \hat{b}_{t,i} + \Delta x_{t,i}. \tag{7}$$

Together, the shared and point-wise paths provide complementary correction capacity: the former ties corrections across Gaussians with common basis support, while the latter retains point-specific adjustments. Thus, grouping makes only the shared correction branch compact; it does not impose a low-dimensional restriction on the final per-Gaussian motion field.

### 4.3 Motion-Aware Routing and Temporal Coreset

Monocular motion priors are ambiguous near dynamic boundaries, and long videos contain redundant observations. *Motion-Aware Routing and Temporal Coreset* (MRTC) addresses both by assigning static, transitional, or dynamic motion freedom from point evidence and concentrating updates on representative temporal controls while preserving the full output sequence.

*Tri-state motion-confidence routing.* At its canonical projection $p_{0,i}$, each persistent point receives the fixed score $\theta_i = S_0(p_{0,i}) \in [0, 1]$. This forward estimate is a motion prior, not ground truth; RGB supervision refines only the corrections permitted by the fixed routing below. Because foreground masks are not provided as input, we use this fixed dynamic score to partition Gaussians into static, uncertain, and dynamic states with thresholds $\theta_s = 0.10$ and $\theta_d = 0.65$. Following the cubic Hermite smoothstep interpolation for continuous threshold-based blending (Perlin, 1985; Günther et al., 2011), we define the normalized dynamic score and motion gate as

$$r_i = \mathrm{clamp}\left( \frac{\theta_i - \theta_s}{\theta_d - \theta_s}, 0, 1 \right), \qquad m_i = r_i^2 (3 - 2 r_i). \tag{8}$$

Here $r_i$ is the score normalized to $[0, 1]$ between thresholds, and $m_i$ its smooth motion gate. Thus $m_i = 0$ for static points, $m_i = 1$ for dynamic points, and uncertain points receive an intermediate motion gate.

Each Gaussian has time-shared appearance and shape parameters $\boldsymbol{\psi}_i$ and a static displacement $\Delta u_i^{\text{static}} \in \mathbb{R}^3$. We explicitly define its static center and its smooth transition between the static and dynamic hypotheses as

$$\mu_i^{\text{static}} = \mu_{0,i}^o + \Delta u_i^{\text{static}}, \qquad \mu_{t,i}^{\text{transition}} = (1 - m_i)\mu_i^{\text{static}} + m_i \mu_{t,i}^{\text{dynamic}}. \tag{9}$$

The final center $\mu_{t,i}$ is $\mu_i^{\text{static}}$ for $\theta_i \le \theta_s$, $\mu_{t,i}^{\text{transition}}$ for $\theta_s < \theta_i < \theta_d$, and $\mu_{t,i}^{\text{dynamic}}$ for $\theta_i \ge \theta_d$. Static Gaussians thus remain time-invariant, uncertain ones interpolate between hypotheses, and dynamic ones receive full grouped and point-wise corrections. Moreover, tri-state routing replaces hard foreground-mask partitioning: when complex motion leaves moving background outside a foreground mask, uncertain support remains available for trajectory-driven refinement rather than being blurred as static. All three states share one depth-ordering and alpha-compositing pass rather than separate scene models. We deliberately keep Gaussian covariance time-shared and learn it from all frames rather than transporting it by frozen per-frame basis rotations; this is a fixed-shape approximation that favors a stable shared shape model, not a claim of explicit framewise covariance rotation.

*Motion-aware temporal coreset.* For long videos, sparse temporal support can represent adjacent motion while reaching the calibrated quality criterion with fewer Stage-B updates than full control. We retain all $T$ outputs but allocate renderer and backward pass only to representative $\mathcal{S} \subset \{0, \dots, T-1\}$. Let $\mathcal{U}$ denote the eligible dynamic support and $\kappa_0(i) = \arg\max_{1 \le k \le K} a_{i,k}$ the dominant basis; reindex the $M$ nonempty basis values on $\mathcal{U}$ as $\kappa(i) \in \{1, \dots, M\}$, with $\mathcal{Q}_j = \{i \in \mathcal{U} \mid \kappa(i) = j\}$. For each nonempty component, averaging $\mu_{t,i}^o$ over $i \in \mathcal{Q}_j$ gives prior-trajectory centroid $\mathbf{z}_{t,j}$, while normalizing $\sum_{i \in \mathcal{Q}_j} m_i$ across components gives importance $\nu_j$. We smooth $\mathbf{z}_{t,j}$ temporally with the fixed binomial kernel $\frac{1}{16}[1, 4, 6, 4, 1]$ to obtain $\bar{\mathbf{z}}_{t,j}$, then compute its source-relative displacement $\mathbf{d}_{t,j} = \bar{\mathbf{z}}_{t,j} - \bar{\mathbf{z}}_{0,j}$ and velocity $\mathbf{v}_{t,j} = \nabla_t \bar{\mathbf{z}}_{t,j}$. Concatenating these component motions and projecting them onto their leading principal directions gives

$$\mathbf{y}_t = \operatorname{vec}\Big( \big\{ \sqrt{\nu_j}\, \mathbf{d}_{t,j}, \sqrt{\nu_j}\, \mathbf{v}_{t,j} \big\}_{j:\, \mathcal{Q}_j \neq \emptyset} \Big), \qquad \boldsymbol{\phi}_t = \mathbf{V}_{d_\phi}^{\mathsf{T}} (\mathbf{y}_t - \bar{\mathbf{y}}). \tag{10}$$

Here $\bar{\mathbf{y}} = T^{-1} \sum_{t=0}^{T-1} \mathbf{y}_t$, and $\mathbf{V}_{d_\phi}$ contains the leading PCA directions that retain $99.5\%$ of the descriptor variance. Starting from $\mathcal{S} = \{0, T-1\}$, we greedily insert the time least explained by neighboring representatives. For each $t \notin \mathcal{S}$, let $\ell_t = \max\{s \in \mathcal{S} \mid s < t\}$ and $u_t = \min\{s \in \mathcal{S} \mid s > t\}$ denote its nearest selected times. Using the normalized temporal coordinate $\lambda_t = (t - \ell_t)/(u_t - \ell_t)$, we measure the interpolation error as

$$e_t = \left\| \boldsymbol{\phi}_t - [(1 - \lambda_t)\boldsymbol{\phi}_{\ell_t} + \lambda_t \boldsymbol{\phi}_{u_t}] \right\|_2. \tag{11}$$

We repeatedly add $t^\star = \arg\max_{t \notin \mathcal{S}} e_t$ until $\max_{t \notin \mathcal{S}} e_t \le \epsilon$, where $\epsilon = \text{median}_t \|\mathbf{y}_t^{\text{raw}} - \mathbf{y}_t\|_2$ estimates high-frequency trajectory noise by comparing the descriptors before and after temporal smoothing.

### 4.4 Progressive Gaussian Optimization

Early projection and topology errors can distort prematurely refined trajectories. *Progressive Gaussian Optimization* (PGO) separates *Scene stabilization* and *Motion refinement*, releasing residual motion only after Gaussian coverage and camera geometry stabilize. Input cameras are refined with rotation and translation residuals $(\omega_t, \Delta t_t)$ around $C_t^o$.

*Stage A: Scene stabilization.* Stage A optimizes Gaussian appearance, shape, and cameras with static and trajectory corrections fixed at zero; splitting and pruning occur only here. The stage objective is

$$\mathcal{L}_{\text{A}} = \mathcal{L}_{\text{rgb}} + \lambda_{\text{shape}} \mathcal{L}_{\text{shape}} + \lambda_{\text{opacity}} \mathcal{L}_{\text{opacity}} + \lambda_{\text{cam}} \mathcal{L}_{\text{cam}} + \lambda_{\text{smooth}} \mathcal{L}_{\text{smooth}}. \tag{12}$$

Here $\mathcal{L}_{\text{rgb}} = \lambda_{\ell_1} \mathcal{L}_{\ell_1} + \lambda_{\text{ssim}} \mathcal{L}_{\text{ssim}}$ supervises rendered RGB with standard $\ell_1$ reconstruction and SSIM (Kerbl et al., 2023; Wang et al., 2004); $\lambda_{\ell_1}$ and $\lambda_{\text{ssim}}$ weight the terms and $\mathcal{L}_{\text{ssim}}$ is differentiable SSIM dissimilarity. Inspired by Gaussian anisotropy constraints (Xie et al., 2024), $\mathcal{L}_{\text{shape}}$ bounds normal-to-tangential scale ratios to prevent degenerate shapes, while $\mathcal{L}_{\text{opacity}}$ sparsifies

opacity to suppress redundant Gaussians (Kheradmand et al., 2024). The camera terms $\mathcal{L}_{\text{cam}}$ and $\mathcal{L}_{\text{smooth}}$ penalize residual magnitude and temporal acceleration, respectively:

$$\mathcal{L}_{\text{cam}} = \operatorname*{mean}_t\big(\|\omega_t\|_2^2 + \|\Delta t_t\|_2^2\big), \qquad \mathcal{L}_{\text{smooth}} = \operatorname*{mean}_{1\le t<T-1}\big(\|\nabla_t^2\omega_t\|_2^2 + \|\nabla_t^2\Delta t_t\|_2^2\big), \tag{13}$$

where $\nabla_t^2 y_t = y_{t+1} - 2y_t + y_{t-1}$. We then freeze topology, associate each Gaussian with its nearest canonical persistent prior, transfer frozen trajectory, confidence, score, and coefficients, and construct motion gates and basis groups.

*Stage B: Motion refinement.* With fixed topology and, for long videos, a constructed coreset, Stage B jointly refines Gaussian attributes, $\Delta u_i^{\text{static}}$, $\Delta\tau_{t,g}$, $\Delta x_{t,i}$, and cameras by minimizing

$$\mathcal{L}_{\text{B}} = \mathcal{L}_{\text{A}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}} + \lambda_{\text{motion}}\mathcal{L}_{\text{motion}}. \tag{14}$$

Here $\mathcal{L}_{\text{A}}$ retains Stage-A reconstruction and regularization; $\mathcal{L}_{\text{reg}}$ limits correction magnitude by confidence-weighting local residuals, while $\mathcal{L}_{\text{motion}}$ penalizes local and grouped temporal acceleration:

$$\begin{aligned}\mathcal{L}_{\text{reg}} &= \frac{1}{TN}\sum_{t,i}\max(q_{t,i},\varepsilon_q)\|\Delta x_{t,i}\|_1 + \eta_{\text{shared}}\frac{1}{TG}\sum_{t,g}\|\Delta\tau_{t,g}\|_1,\\ \mathcal{L}_{\text{motion}} &= \operatorname*{mean}_{t,i}\|\nabla_t^2\Delta x_{t,i}\|_1 + \eta_{\text{shared}}\operatorname*{mean}_{t,g}\|\nabla_t^2\Delta\tau_{t,g}\|_1,\end{aligned} \tag{15}$$

where $N$ is the Gaussian count, $\varepsilon_q > 0$ floors confidence weights, and $\eta_{\text{shared}}$ balances grouped and local corrections. These penalties favor small, temporally smooth use of both paths, rather than identifying a unique physical shared–local decomposition.

# 5 Experiments

## 5.1 Experimental Setup

**Implementation details.** We use Adam with exponential learning-rate decay and the two-stage PGO schedule of Sec. 4.4. All quantitative experiments, including every baseline reproduction, use a single NVIDIA RTX 4090 GPU and the same predefined convergence criterion under the corresponding protocol. Using held-out HyperNeRF validation-rig views, we select the smallest Stage-A/B budget whose further extension fails to improve PSNR by at least $0.005$ dB; full GaussTrace consequently uses fixed $4{,}000$/$3{,}000$ updates on every reported benchmark, while each ablation configuration follows this selection criterion. For the RGB reconstruction term, we set $\lambda_{\ell_1} = 0.8$ and $\lambda_{\text{ssim}} = 0.2$ for the $\ell_1$ and SSIM components, respectively. The remaining Stage-A weights are $\lambda_{\text{shape}} = 10^{-2}$, $\lambda_{\text{opacity}} = \lambda_{\text{cam}} = 10^{-3}$, and $\lambda_{\text{smooth}} = 10^{-2}$; the Stage-B weights are $\lambda_{\text{reg}} = \lambda_{\text{motion}} = 10^{-4}$. We use $\eta_{\text{shared}} = 0.5$ and $\varepsilon_q = 10^{-6}$.

**Datasets.** We evaluate dynamic novel-view synthesis on NVIDIA Dynamic Scenes, iPhone DyCheck, and DAVIS. NVIDIA Dynamic Scenes (Yoon et al., 2020) contains seven synchronized multi-camera sequences and supports source-to-test evaluation under camera-moving and fixed-camera full-sequence protocols. iPhone DyCheck (Gao et al., 2022) provides casually captured sequences with non-rigid motion and synchronized static views for monocular training and test-view evaluation. We report PSNR, SSIM (Wang et al., 2004), LPIPS (Zhang et al., 2018), and Training Time, excluding prior extraction and preprocessing for all methods. DAVIS (Perazzi et al., 2016) provides in-the-wild monocular videos averaging 70 frames but no ground-truth novel views, so we use it for qualitative comparison only.

**Baselines.** We compare against 4D Gaussian Splatting (4DGS) (Wu et al., 2024), RoDynRF (Liu et al., 2023), MoSca (Lei et al., 2025), Shape of Motion (Wang et al., 2025), SplineGS (Park et al., 2025), $\mathrm{SE}(3)$ B-SplineGS (Zhang et al., 2026b), MotionScale (Zhou & Lee, 2026), Gaussian Flow (Lin et al., 2024), Gaussian Marbles (Stearns et al., 2024), and StreamSplat (Wu et al., 2026). Every quantitative baseline entry is reproduced by us from its publicly released implementation using the protocol-specific split and the common convergence criterion above.

## 5.2 Comparisons

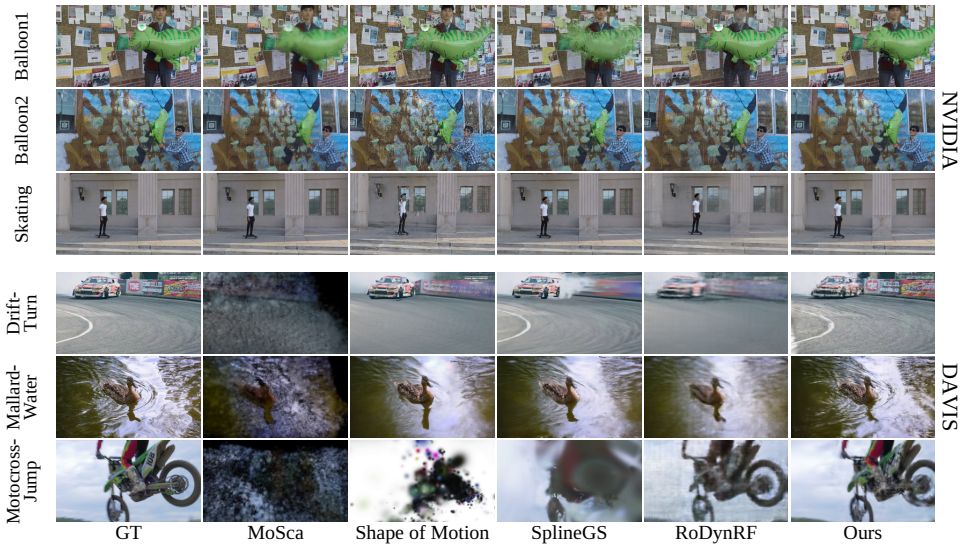


Figure 3: Qualitative comparison on NVIDIA Dynamic Scenes and DAVIS. GaussTrace yields sharper object boundaries, more coherent motion, and more faithful dynamic details than competing methods.

**NVIDIA Dynamic Scenes.** Under the 12-frame RoDynRF protocol (Liu et al., 2023), we train on camera 0 at $t = 0, \ldots, 11$ and evaluate synchronized held-out camera 1 frames. Table 1 reports best PSNR/SSIM, third-lowest LPIPS, and second-fastest optimization; Figure 3 shows sharper boundaries, more accurate motion, and more faithful dynamic details.

Table 1: Twelve-frame NVIDIA comparison (camera $0 \rightarrow 1$) under the RoDynRF protocol (Liu et al., 2023).

| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Training Time ↓ |
|---|---|---|---|---|
| 4DGS | 23.76 | 0.748 | 0.113 | 9 min |
| RoDynRF | 25.50 | 0.782 | 0.069 | 11.2 h |
| MoSca | 26.73 | 0.813 | <u>0.059</u> | 17 min |
| Shape of Motion | 23.49 | 0.761 | 0.092 | 9 min |
| SplineGS | <u>27.33</u> | <u>0.882</u> | **0.043** | 1.7 h |
| SE(3) B-SplineGS | 20.28 | 0.672 | 0.107 | 20 min |
| MotionScale | 25.74 | 0.793 | 0.096 | 2 h |
| StreamSplat | 14.67 | 0.552 | 0.658 | **10 s** |
| **Ours** | **27.64** | **0.894** | 0.060 | <u>2 min</u> |

Under the full-sequence Gaussian Marbles protocol (Stearns et al., 2024), camera 0 supplies the monocular sequence and synchronous camera 1 defines NVS evaluation. Table 2 reports best PSNR/SSIM and optimizer time under both fixed-camera protocols, with second-lowest LPIPS; consistency across sparse-frame and full-sequence protocols shows that the trajectory prior remains effective under limited and dense temporal coverage.

**iPhone DyCheck.** On iPhone DyCheck (Gao et al., 2022), we reconstruct the released monocular training videos and evaluate the released held-out static views; GaussTrace attains best PSNR/SSIM and fastest optimization while retaining second-lowest LPIPS.

**DAVIS.** DAVIS provides no ground-truth novel views for these monocular videos, so we use it only qualitatively. On DAVIS (Perazzi et al., 2016), Mallard-Water contains non-rigid background ripples, while Drift-Turn and Motocross-Jump combine rapid camera motion and large object displacements; GaussTrace preserves their dynamic structures and boundaries more coherently than methods with fixed or independently estimated motion support.

### 5.3 Ablation Study

We ablate TBMM, MRTC, and PGO on NVIDIA Dynamic Scenes (Yoon et al., 2020) under the Gaussian Marbles protocol (Stearns et al., 2024); Table 3 and Figure 4 report the quantitative and qualitative results.

Table 2: Full-sequence comparison on NVIDIA (camera $0 \rightarrow 1$) under the Gaussian Marbles protocol (Stearns et al., 2024) and iPhone DyCheck under its official split.

| Method | NVIDIA | | | | iPhone DyCheck | | | |
|---|---|---|---|---|---|---|---|---|
| | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Training Time ↓ | PSNR ↑ | SSIM ↑ | LPIPS ↓ | Training Time ↓ |
| Gaussian Flow | 21.91 | 0.747 | 0.086 | 19 min | 16.44 | 0.601 | 0.321 | 27 min |
| Gaussian Marbles | 23.70 | 0.802 | **0.059** | 4 h | 16.84 | 0.605 | 0.408 | 6 h |
| MoSca | 22.54 | 0.766 | 0.084 | 1 h | 19.44 | 0.716 | **0.254** | 50 min |
| SE(3) B-SplineGS | 19.57 | 0.711 | 0.102 | 30 min | 16.67 | 0.591 | 0.301 | 23 min |
| MotionScale | 22.43 | 0.731 | 0.075 | 2 h | 18.34 | 0.622 | 0.275 | 2 h |
| **Ours** | **23.91** | **0.821** | 0.070 | **4 min** | **19.68** | **0.721** | 0.261 | **4 min** |

**The effect of TBMM.** We ablate trajectory-basis grouping (TBG) and correction (TBC). Without TBG, each of the 516 bases receives its own shared translation correction instead of the 32-group correction. TBG preserves comparable visual quality (Table 3; Figure 4) while making the shared decoder $16.1\times$ narrower; this lowers its mixing, gradient-reduction, and coefficient-buffer cost, although rasterization and dense local residuals remain the dominant end-to-end workload. Removing TBC degrades reconstruction quality and motion alignment, confirming the need for learnable correction beyond frozen feed-forward trajectories. Supplementary decompositions remove the group-shared correction or point-wise residual in turn and establish their empirical complementarity, rather than a unique physical factorization of motion.

Table 3: Ablations on NVIDIA Dynamic Scenes; each configuration uses its HyperNeRF-selected 0.005-dB budget.

| Variant | PSNR↑ | SSIM↑ | LPIPS↓ | Training Time ↓ |
|---|---|---|---|---|
| w/o TBG | 23.82 | 0.813 | 0.072 | 5 min |
| w/o TBC | 17.24 | 0.598 | 0.248 | **4min** |
| w/o TMCR | 20.48 | 0.715 | 0.094 | **4min** |
| w/o MATC | 23.86 | 0.820 | **0.070** | 7min |
| w/o PGO | 21.39 | 0.749 | 0.088 | **4min** |
| Full | **23.91** | **0.821** | **0.070** | **4min** |

**The effect of MRTC.** We ablate MRTC's tri-state motion-confidence routing (TMCR) and motion-aware temporal coreset (MATC). Replacing TMCR with binary static/dynamic routing degrades reconstruction (Table 3; Figure 4), showing that the transitional state preserves uncertain motion at ambiguous dynamic boundaries. For the reported long-sequence NVS evaluation, full-frame optimization reaches the same HyperNeRF-selected 0.005 dB convergence criterion only with a 4,000/5,000 schedule, raising training time from 4 to 7 minutes; MATC thus removes temporal redundancy empirically without material held-out quality loss.

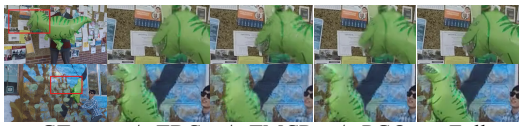


Figure 4: Qualitative ablations on Balloon1/2.

**The effect of PGO.** Replacing PGO with one-stage joint updates of Gaussian attributes, cameras, and static, group-wise, and point-wise trajectory corrections from the first iteration degrades results (Table 3; Figure 4), showing that stabilizing scene structure before motion refinement prevents camera and trajectory corrections from absorbing early geometry and appearance errors.

## 6 Conclusion

GaussTrace is a framework built around persistent 3D trajectories that constructs an explicit dynamic Gaussian scene from a single feed-forward 4D prediction followed by short, constrained per-scene refinement. TBMM makes its shared trajectory-basis correction branch compact while retaining local residual freedom; MRTC assigns motion freedom and selects representative times; and PGO stabilizes scene structure before joint refinement. Together, these modules preserve long-range motion coherence, recover sharper dynamic boundaries and non-rigid details, and achieve strong reconstruction quality with only 2–4 minutes of scene-specific optimization across NVIDIA Dynamic Scenes, iPhone DyCheck, and DAVIS.

### AI USE STATEMENT

We used generative AI tools for language editing and code-assistance during development. All authors reviewed the resulting text, code, experiments, and figures and take responsibility for the final manuscript and artifacts.

### ETHICS STATEMENT

This work evaluates dynamic-scene reconstruction on the publicly available NVIDIA Dynamic Scenes, iPhone DyCheck, and DAVIS benchmarks. We use these data only for research, collect no new human-subject data, and release no personally identifying information.

### REPRODUCIBILITY STATEMENT

The main paper specifies the input prior, optimization schedule, loss weights, renderer, protocol-specific source/test views and frame sampling, per-scene metrics, and optimizer-only timing boundary. OmniX is an ECCV work with publicly released code and pretrained weights; we use its released weights unchanged as the frozen input prior. The anonymous submission package contains `ICLR19403_core_code`, `ICLR19403_Video.mp4`, and a single `main.pdf`. The core-code artifact implements the paper-specific TBMM, MRTC, and PGO operations for method inspection, while `main.pdf` provides algorithmic details, experimental configurations, and additional quantitative and qualitative analyses. The supplementary video presents novel-view synthesis across NVIDIA, iPhone, and DAVIS scenes, tri-state motion-score visualizations against masks, trajectory changes before and after refinement, and comparisons on challenging background-motion and large-displacement cases. The core code is deliberately a method-level supplement rather than a runnable end-to-end release; `main.pdf` specifies the associated protocols and hyperparameters.

## REFERENCES

Minh-Quan Viet Bui, Jongmin Park, Jihyong Oh, and Munchurl Kim. MoBluRF: Motion deblurring neural radiance fields for blurry monocular video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(9):7752–7770, 2025.

Julie Carmigniani and Borko Furht. Augmented reality: An overview. In Borko Furht (ed.), *Handbook of Augmented Reality*, pp. 3–46. Springer, 2011.

Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. DeFanti, Robert V. Kenyon, and John C. Hart. The CAVE: Audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72, 1992.

Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. TAP-Vid: A benchmark for tracking any point in a video. In *Advances in Neural Information Processing Systems*, volume 35, pp. 13610–13626, 2022.

Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, João Carreira, and Andrew Zisserman. TAPIR: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10061–10072, 2023.

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*, 2024. Article 87.

Haiwen Feng, Junyi Zhang, Qianqian Wang, Yufei Ye, Pengcheng Yu, Michael J. Black, Trevor Darrell, and Angjoo Kanazawa. St4RTrack: Simultaneous 4D reconstruction and tracking in the world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8503–8513, 2025.

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

Tobias Günther, Kai Bürger, Rüdiger Westermann, and Holger Theisel. A view-dependent and interframe coherent visualization of integral lines using screen contribution. In *VMV 2011 – Vision, Modeling and Visualization*, pp. 215–222. The Eurographics Association, 2011.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D value maps for robotic manipulation with language models. In *Proceedings of the 7th Conference on Robot Learning*, volume 229, pp. 540–562, 2023.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270, pp. 4573–4602, 2025.

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 (CVPR)*, pp. 4220–4230, 2024.

Yanqin Jiang, Tengfei Wang, Zhengwei Wang, Chenjie Cao, Junta Wu, Wenhan Luo, Weiming Hu, Jin Gao, and Chunchao Guo. OmniX: Any-view and any-time 4D reconstruction via feed-forward trajectory fields. In *European Conference on Computer Vision (ECCV)*, pp. 295–312, 2026.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is better to track together. In *European Conference on Computer Vision (ECCV)*, pp. 18–35, 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):139, 2023.

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D gaussian splatting as markov chain monte carlo. In *Advances in Neural Information Processing Systems*, volume 37, pp. 80965–80986, 2024.

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 (ECCV)*, pp. 252–269, 2024.

Junoh Lee, Changyeon Won, Hyunjun Jung, Inhwan Bae, and Hae-Gon Jeon. Fully explicit dynamic gaussian splatting. In *Advances in Neural Information Processing Systems*, volume 37, pp. 5384–5409, 2024a.

Yao-Chih Lee, Zhoutong Zhang, Kevin Blackburn-Matzen, Simon Niklaus, Jianming Zhang, Jia-Bin Huang, and Feng Liu. Fast view synthesis of casual videos with Soup-of-Planes. In *European Conference on Computer Vision (ECCV)*, pp. 278–296, 2024b.

Jiahui Lei, Yijia Weng, Adam W. Harley, Leonidas Guibas, and Kostas Daniilidis. MoSca: Dynamic gaussian fusion from casual videos via 4D motion scaffolds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6165–6177, 2025.

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

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6498–6508, 2021.

Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. DynIBaR: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4273–4284, 2023.

Yiqing Liang, Numair Khan, Zhengqin Li, Thu H. Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. GauFRe: Gaussian deformation fields for real-time dynamic novel view synthesis. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pp. 2642–2652, 2025.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. In *International Conference on Learning Representations (ICLR)*, 2026.

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 (CVPR)*, pp. 21136–21145, 2024.

Qingming Liu, Yuan Liu, Jiepeng Wang, Xianqiang Lyu, Peng Wang, Wenping Wang, and Junhui Hou. MoDGS: Dynamic gaussian splatting from casually-captured monocular videos with depth priors. In *International Conference on Learning Representations (ICLR)*, 2025.

Yu-Lun Liu, Chen Gao, Andréas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust dynamic radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13–23, 2023.

Jiahao Lu, Tianyu Huang, Peng Li, Zhiyang Dou, Cheng Lin, Zhiming Cui, Zhen Dong, Sai-Kit Yeung, Wenping Wang, and Yuan Liu. Align3R: Aligned monocular depth estimation for dynamic videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22820–22830, 2025.

Jiahao Lu, Jiayi Xu, Wenbo Hu, Ruijie Zhu, Chengfeng Zhao, Sai-Kit Yeung, Ying Shan, and Yuan Liu. Track4World: Feedforward world-centric dense 3D tracking of all pixels. In *European Conference on Computer Vision (ECCV)*, pp. 164–183, 2026.

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

Yihang Luo, Shangchen Zhou, Yushi Lan, Xingang Pan, and Chen Change Loy. 4RC: 4D reconstruction via conditional querying anytime and anywhere. In *International Conference on Machine Learning (ICML)*, 2026.

James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pp. 281–297, 1967.

Xingyu Miao, Yang Bai, Haoran Duan, Fan Wan, Yawen Huang, Yang Long, and Yefeng Zheng. CT-NeRF: Cross-time transformer for dynamic neural radiance field from monocular video. *Pattern Recognition*, 156:110729, 2024.

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. *Communications of the ACM*, 65(1):99–106, 2022.

Tuan Duc Ngo, Peiye Zhuang, Chuang Gan, Evangelos Kalogerakis, Sergey Tulyakov, Hsin-Ying Lee, and Chaoyang Wang. DELTA: Dense efficient long-range 3D tracking for any video. In *International Conference on Learning Representations (ICLR)*, 2025.

Jongmin Park, Minh-Quan Viet Bui, Juan Luis Gonzalez Bello, Jaeho Moon, Jihyong Oh, and Munchurl Kim. SplineGS: Robust motion-adaptive spline for real-time dynamic 3D gaussians from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26866–26875, 2025.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (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*, 40(6): 238, 2021b.

Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 724–732, 2016.

Ken Perlin. An image synthesizer. *ACM SIGGRAPH Computer Graphics*, 19(3):287–296, 1985.

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 (CVPR)*, pp. 10318–10327, 2021.

Colton Stearns, Adam W. Harley, Mikaela Uy, Florian Dubost, Federico Tombari, Gordon Wetzstein, and Leonidas Guibas. Dynamic Gaussian Marbles for novel view synthesis of casual monocular videos. In *SIGGRAPH Asia 2024 Conference Papers*, 2024. Article 30.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2446–2454, 2020.

Fengrui Tian, Shaoyi Du, and Yueqi Duan. MonoNeRF: Learning a generalizable dynamic radiance field from monocular videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17903–17913, 2023.

Allen Tu, Haiyang Ying, Alex Hanson, Yonghan Lee, Tom Goldstein, and Matthias Zwicker. SpeeDe3DGS: Speedy deformable 3D gaussian splatting with temporal pruning and motion grouping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26083–26093, 2026.

Chaoyang Wang, Ben Eckart, Simon Lucey, and Orazio Gallo. Neural trajectory fields for dynamic novel view synthesis. *arXiv preprint arXiv:2105.05994*, 2021.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9660–9672, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20697–20709, 2024.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

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

Zike Wu, Qi Yan, Xuanyu Yi, Lele Wang, and Renjie Liao. StreamSplat: Towards online dynamic 3D reconstruction from uncalibrated video streams. In *International Conference on Learning Representations (ICLR)*, 2026.

Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9421–9431, 2021.

Yuxi Xiao, Qianqian Wang, Shangzhan Zhang, Nan Xue, Sida Peng, Yujun Shen, and Xiaowei Zhou. SpatialTracker: Tracking any 2D pixels in 3D space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20406–20417, 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6726–6737, 2025.

Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. PhysGaussian: Physics-integrated 3D gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4389–4398, 2024.

Zhen Xu, Zhengqin Li, Zhao Dong, Xiaowei Zhou, Richard Newcombe, and Zhaoyang Lv. 4DGT: Learning a 4D gaussian transformer using real-world monocular videos. In *Advances in Neural Information Processing Systems*, volume 38, pp. 47576–47602, 2025.

Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10371–10381, 2024a.

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

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 (CVPR)*, pp. 20331–20341, 2024c.

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 (CVPR)*, pp. 5336–5345, 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7382–7392, 2026a.

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 (ICLR)*, 2025.

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 (CVPR)*, pp. 586–595, 2018.

Xuankai Zhang, Junjin Xiao, Shangwei Huang, Wei-shi Zheng, and Qing Zhang. Learning explicit continuous motion representation for dynamic gaussian splatting from monocular videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 33291–33300, 2026b.

Haoran Zhou and Gim Hee Lee. MotionScale: Reconstructing appearance, geometry, and motion of dynamic scenes with scalable 4D gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11860–11870, 2026.

# A APPENDIX

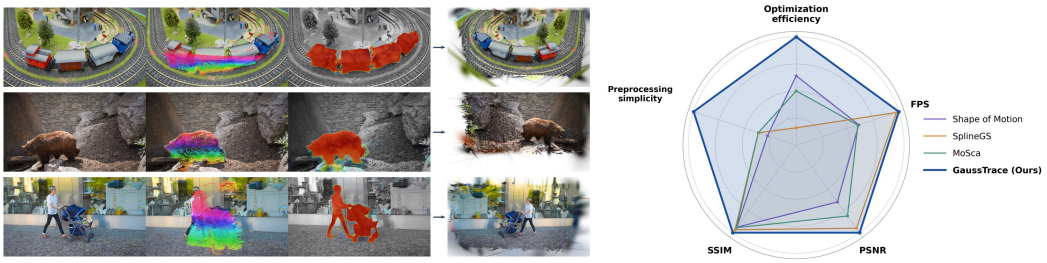


Figure 5: Appendix teaser of GaussTrace. A single OmniX preprocessing pass supplies depth–ray geometry, dense 3D point trajectories, dynamic-score maps, and trajectory transformation bases that guide novel-view synthesis (left); optimizing these unified priors improves optimization speed, preprocessing simplicity, rendering throughput, and reconstruction quality over representative monocular 4DGS systems (right).

This appendix covers unified feed-forward preprocessing; experiments on motion robustness, tracking, grouping, and alternative priors; algorithmic details; and limitations and long-sequence OmniX execution, with per-scene renderings, metrics, and launch configurations for NVIDIA experiments.

## A.1 NVIDIA AND IPHONE EVALUATION PROTOCOLS

We reproduce every NVIDIA and iPhone quantitative entry, including all baselines, on the same RTX 4090 using the released synchronized split and the corresponding predefined convergence criterion. NVIDIA results use one of two established benchmark protocols. For every scene, a tuple is written as $(t, c)$ for timestamp $t$ and camera ID $c$: camera 0 RGB is the only optimization image source, and camera 1 RGB is evaluation-only. The 12-frame setting uses exactly $\{(t, 0) \mid t = 0, \ldots, 11\}$ for training and $\{(t, 1) \mid t = 0, \ldots, 11\}$ for testing; the full-sequence setting replaces this prefix by all released timestamps of the same scene. The released per-frame calibration supplies intrinsics and extrinsics for NVS queries, while held-out RGB never enters optimization, model selection, or hyperparameter selection. The released RGB height is $496$ pixels, and its width is $940$ pixels except for Balloon2 $(944)$ and Playground $(1028)$. PSNR, SSIM, and LPIPS are averaged over all full-frame RGB test images without foreground masks. All 12-frame NVIDIA results therefore use the same camera 0-to-camera 1 test set, and all full-sequence NVIDIA results use the corresponding complete synchronized sequences.

For iPhone DyCheck, we follow the released official split: the monocular training video is the only optimization input, and the provided held-out static-view images and calibrations are used only for novel-view evaluation. Metrics are averaged over all released held-out images for each sequence without foreground masking.

## A.2 UNIFIED FEED-FORWARD PREPROCESSING

**Unified preprocessing interface.** One OmniX forward pass supplies coordinate-aligned camera initialization, dense 3D point trajectories, dynamic scores, and trajectory transformation bases (Jiang et al., 2026). Unlike fragmented pipelines that assemble camera recovery, monocular depth, motion segmentation, optical flow, and long-range 2D tracks from independently trained estimators (Wang et al., 2025; Park et al., 2025; Liu et al., 2025; Lei et al., 2025), GaussTrace obtains the required priors from this single frozen query model in a shared coordinate system. The predicted camera parameters initialize the input views, and the accompanying depth–ray geometry supplies the 3D support used to initialize Gaussian primitives; hence, no separate pose-recovery or depth-to-point-cloud alignment stage is invoked (Jiang et al., 2026). The persistent dense 3D trajectories directly provide cross-frame identities and 3D motion, avoiding the usual chain of 2D tracking, track association, lifting, and camera-dependent triangulation (Karaev et al., 2024; Wang et al., 2025; Park et al., 2025). The pixel-level dynamic scores are compiled into fixed per-Gaussian motion evidence

Table 4: Evaluation manifests. $\mathcal{T}_s$ is the released full timestamp set of NVIDIA scene $s$; NVIDIA metrics use synchronized held-out camera 1 RGB only, and HyperNeRF uses its official validation-rig split from `dataset.json`.

| Dataset and protocol | Optimization inputs | NVS inputs | Scenes and sequence length |
|---|---|---|---|
| NVIDIA, RoDynRF 12-frame | $\{(t,0) \mid t = 0, \ldots, 11\}$ | $\{(t,1) \mid t = 0, \ldots, 11\}$ | Balloon1, Balloon2, Jumping, Playground, Skating, Truck, Umbrella; $T = 12$ |
| NVIDIA, Gaussian Marbles full sequence | $\{(t,0) \mid t \in \mathcal{T}_s\}$ | $\{(t,1) \mid t \in \mathcal{T}_s\}$ | Balloon1/2: 192; Jumping: 96; Playground: 144; Skating: 132; Truck: 204; Umbrella: 180 |
| HyperNeRF, official validation-rig | Released `dataset.json` training split | Released `dataset.json` held-out split | Selected `vrig_*` development sequences; official novel-view setting |
| iPhone DyCheck, official split | Released monocular training video | Released held-out static-view RGB and calibration | Seven casually captured dynamic sequences; released full sequence |

Table 5: Preprocessing dependencies of monocular dynamic reconstruction pipelines.

| Method | Required preprocessing |
|---|---|
| Shape of Motion | Camera-pose estimation, monocular depth, long-range 2D tracking, and dynamic segmentation |
| SplineGS | Monocular depth, long-range 2D tracking, and dynamic segmentation |
| MoDGS | Calibrated camera poses, monocular depth, and optical flow |
| MoSca | Monocular depth, long-range 2D tracking, optical flow, and semantic/motion cues |
| SpeeDe3DGS | A pretrained dynamic-Gaussian initialization and motion grouping |
| GaussTrace | One unified OmniX forward pass |

for static, uncertain, and dynamic routing, so neither an external foreground mask nor an additional motion-segmentation network is required (Jiang et al., 2026). Likewise, the trajectory transformation bases and their Deformable Trajectory Sampling Head (DTSH) interpolation coefficients are predicted together with the trajectories, allowing group-shared translation corrections to act on motion support that is already aligned with geometry and identity instead of separately registered optical flow (Jiang et al., 2026). This simplification removes external preprocessing modules rather than treating the feed-forward outputs as ground truth: the subsequent two-stage Gaussian optimization still makes regularized residual corrections from the input video.

**Topology-stable prior association.** Stage A may split or prune Gaussians, so after topology stabilizes, each current Gaussian $i$ is associated with the nearest source persistent point in canonical space, $j(i) = \arg\min_j \|\mathbf{x}^{\mathrm{A}}_{c,i} - \mathbf{x}^{o}_{c,j}\|_2$. Here $\mathbf{x}^{\mathrm{A}}_{c,i}$ and $\mathbf{x}^{o}_{c,j}$ are, respectively, the canonical centers of the current Stage-A Gaussian and the source prior point. This correspondence transfers the frozen source trajectory, confidence, dynamic score, and sparse basis coefficients to the current Gaussian; thereafter we write these attributes with index $i$. For a split child, the base trajectory is initialized from its parent by $\mathbf{x}^{\mathrm{child}}_t = \mathbf{x}^{\mathrm{parent}}_t + \boldsymbol{\delta}_c$ for all $t$, whereas pruning removes the associated point-indexed state. Lifecycle-maintained trajectory-basis coefficient rows are preserved when compatible with the stabilized topology and otherwise reconstructed from the associated prior; all Stage-B correction variables are initialized to zero. Table 5 contrasts this unified interface with the external preprocessing dependencies of representative monocular dynamic reconstruction pipelines (Wang et al., 2025; Park et al., 2025; Liu et al., 2025; Lei et al., 2025; Tu et al., 2026).

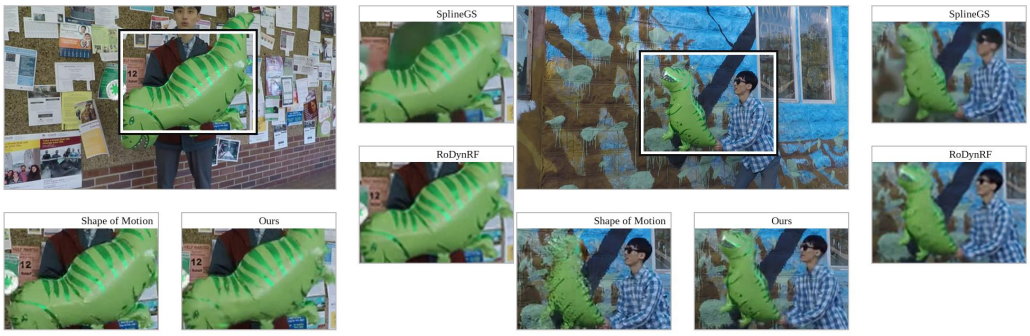


Figure 6: Motion-focused ROI comparison on Balloon1 and Balloon2 at frame 08. The ground-truth overview in each panel marks a fixed balloon-centered ROI shared by all methods. GaussTrace preserves sharper balloon contours and stripe details while reducing foreground–background mixing.

**Motion-focused ROI comparison.** Figure 6 visualizes the practical benefit of the unified motion interface described above by comparing identical balloon-centered regions in the held-out Balloon1 and Balloon2 views at frame 08. The regions are selected from the ground truth once and then applied unchanged to every method, so the comparison directly tests reconstruction fidelity on the moving foreground rather than a method-specific crop. Conventional motion modeling either learns scene deformation from image observations, as in RoDynRF (Liu et al., 2023), or assembles motion support from separately estimated depth, tracks, and masks, as in SplineGS (Park et al., 2025) and Shape of Motion (Wang et al., 2025). In contrast, GaussTrace starts from coordinate-aligned persistent 3D identities and native dynamic scores, then uses tri-state routing to prevent static, uncertain, and dynamic evidence from being indiscriminately mixed during motion refinement. Accordingly, GaussTrace reconstructs the moving balloons more accurately, preserving sharper boundaries and coherent green stripe patterns while producing fewer foreground–background mixing artifacts than the three conventional motion models. These motion-focused details demonstrate that the unified prior is not only simpler to obtain, but also supplies a more reliable spatiotemporal motion scaffold for downstream Gaussian reconstruction.

### A.3 Supplementary Experiments

This section supplements the main evaluation with a DAVIS motion-robustness study, long-range 3D tracking, and an inspection of the optimized motion and geometry representation.

#### A.3.1 Per-Scene NVIDIA Dynamic Scenes Comparison

Table 6 expands the 12-frame camera 0-to-camera 1 NVIDIA comparison with per-scene PSNR, LPIPS, and rendering FPS.

#### A.3.2 Robust Motion Modeling on DAVIS

We qualitatively evaluate motion modeling on the DAVIS benchmark (Perazzi et al., 2016) using sixteen consecutive frames from Drift-Turn, Mallard-Water, and Motocross-Jump. Each figure shows the same frames in two temporal rows for GT, SplineGS (Park et al., 2025), Shape of Motion (Wang et al., 2025), and GaussTrace. The conventional pipelines construct their dynamic support from strict motion masks and independently estimated preprocessing cues, making their motion representation sensitive to mask boundaries and errors.

Mallard-Water exposes this issue when water ripples introduce non-rigid motion throughout the nominal background. When a binary mask fails to represent this mixed support precisely, the baselines cannot consistently reconstruct the water background and its interaction with the duck. In Drift-Turn and Motocross-Jump, rapid camera motion and rapid object motion compound this error:

Table 6: Twelve-frame NVIDIA Dynamic Scenes comparison (camera $0 \to 1$) under the RoDynRF protocol. Higher PSNR and FPS, and lower LPIPS and training time, are better.

| Method | Balloon1 | | Balloon2 | | Jumping | | Playground | | Skating | | Truck | | Umbrella | | Average | | FPS↑ | Training Time↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P↑ | L↓ | P↑ | L↓ | P↑ | L↓ | P↑ | L↓ | P↑ | L↓ | P↑ | L↓ | P↑ | L↓ | P↑ | L↓ | | |
| MonoNeRF | 22.01 | 0.119 | 27.48 | 0.042 | 24.38 | 0.081 | 22.73 | 0.120 | 32.18 | 0.034 | 27.68 | 0.105 | 23.74 | 0.170 | 25.74 | 0.096 | 0.05 | 11.7 h |
| 4DGS | 22.01 | 0.143 | 24.97 | 0.071 | 22.49 | 0.168 | 21.48 | 0.079 | 26.84 | 0.074 | 26.05 | 0.087 | 22.48 | 0.168 | 23.76 | 0.113 | 95 | 9 min |
| RoDynRF | 21.93 | 0.112 | 25.70 | 0.054 | 24.39 | 0.090 | 25.32 | 0.042 | 28.83 | 0.036 | 28.97 | 0.056 | 23.37 | 0.094 | 25.50 | 0.069 | 0.45 | 11.2 h |
| MoSca | 23.39 | 0.081 | **29.13** | 0.032 | 25.55 | 0.070 | 23.35 | 0.048 | 32.74 | 0.023 | 28.41 | 0.076 | 24.52 | 0.081 | 26.73 | 0.059 | 35 | 17 min |
| Shape of Motion | 24.02 | **0.048** | 24.03 | 0.064 | 20.20 | 0.141 | 16.99 | 0.158 | 28.03 | 0.037 | 27.77 | **0.038** | 23.39 | 0.157 | 23.49 | 0.092 | 40 | 9 min |
| SplineGS | **24.55** | 0.058 | 28.49 | **0.022** | **25.62** | **0.058** | 24.31 | **0.037** | 33.84 | 0.021 | 28.78 | 0.046 | **25.73** | **0.061** | 27.33 | **0.043** | 400 | 1.7 h |
| **Ours** | 23.82 | 0.104 | 28.80 | 0.041 | 24.94 | 0.102 | **26.36** | **0.038** | **34.86** | **0.012** | **29.84** | 0.050 | 24.87 | 0.076 | **27.64** | 0.060 | **560** | **2 min** |

Table 7: Tri-state threshold calibration under the official HyperNeRF validation-rig protocol using held-out views. Higher PSNR/SSIM and lower LPIPS are better.

| $\theta_s$ | $\theta_d$ | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|---|
| 0.10 | 0.65 | 25.86 | 0.83 | 0.18 |
| 0.10 | 0.55 | 24.17 | 0.78 | 0.20 |
| 0.10 | 0.75 | 24.23 | 0.80 | 0.23 |
| 0.20 | 0.65 | 23.84 | 0.78 | 0.21 |
| 0.30 | 0.65 | 24.13 | 0.79 | 0.22 |

inaccurate foreground support misassigns observations across time and causes severe appearance degradation.

GaussTrace does not convert the OmniX dynamic score into another foreground mask; instead, it compiles the score into a per-Gaussian routing signal that selects the motion freedom used during optimization. Low-score Gaussians use only a time-invariant static offset, high-score Gaussians receive the full group-shared translation correction and point-wise trajectory residual, and intermediate-score Gaussians smoothly interpolate between these two hypotheses. The uncertain state is therefore not a softened foreground label: it retains mixed or boundary support and lets image evidence refine correction magnitude within its fixed gate, rather than irrevocably classifying that support as foreground motion or static background. Together with persistent 3D trajectories and group-shared translation corrections, this routing aggregates consistent cross-frame evidence for the duck, ripples, and their boundaries; it consequently preserves a moving background in Mallard-Water and avoids the severe cross-time misassignment caused by large coupled motion in Drift-Turn and Motocross-Jump.

*Threshold calibration and sensitivity.* We use the official HyperNeRF validation-rig protocol (Park et al., 2021b) as a disjoint development benchmark for the routing thresholds, group count, and update budget. For every selected HyperNeRF validation-rig sequence, we use its official released train/test split: reconstruction uses only the training views, whereas PSNR, SSIM, and LPIPS are computed only on the official held-out views. We select all settings from these held-out metrics and freeze them before every NVIDIA Dynamic Scenes, iPhone DyCheck, and DAVIS evaluation; no HyperNeRF development sequence contributes to a reported benchmark average. The threshold pairs below follow this protocol and are evaluated only on the official held-out views. All HyperNeRF development metrics are produced by our own runs under this fixed split, hardware, and convergence procedure. We adopt $\theta_s = 0.10$ and $\theta_d = 0.65$ after this calibration and retain them for all reported benchmarks. Figure 7 visualizes the fixed partition on HyperNeRF, while Figure 8 illustrates why the conservative static boundary preserves ambiguous support for tri-state refinement.

**Native and alternative gate comparison.** OmniX's native continuous score gate is one reason we select it as the feed-forward backbone; GaussTrace does not claim continuous static–dynamic blending itself as a new primitive. For all gates here, the input is the gate-free trajectory decoded from the frozen bases in Eq. 5, and the score is applied only once to the static and residual-corrected dynamic Gaussian hypotheses. To isolate the gate itself, we additionally compare OmniX's native linear gate, a binary gate, and our thresholded smoothstep gate under the NVIDIA Gaussian Marbles protocol. Every row uses the same gate-free OmniX trajectories, fixed dynamic scores, grouped and point-wise residual parameterization, PGO schedule, and held-out views; only the fixed gate

Table 9: iPhone long-range 3D and projected 2D tracking comparison. Lower EPE and higher $\delta_{3D}$, AJ, $\delta_{\text{avg}}$, and OA are better.

| Method | 3D tracking | | | Projected 2D tracking | | |
|---|---|---|---|---|---|---|
| | EPE↓ | $\delta_{3D}^{0.05}$↑ | $\delta_{3D}^{0.10}$↑ | AJ↑ | $\delta_{\text{avg}}$↑ | OA↑ |
| HyperNeRF | 0.182 | 28.4 | 45.8 | 10.1 | 19.3 | 52.0 |
| DynIBaR | 0.252 | 11.4 | 24.6 | 5.4 | 8.7 | 37.7 |
| Deformable-3D-GS | 0.151 | 33.4 | 55.3 | 14.0 | 20.9 | 63.9 |
| DynMF | 0.188 | 22.9 | 53.8 | 5.5 | 9.5 | 60.5 |
| CoTracker + monocular depth | 0.202 | 34.3 | 57.9 | 24.1 | 33.9 | 73.0 |
| TAPIR + monocular depth | 0.114 | 38.1 | 63.2 | 27.8 | 41.5 | 67.4 |
| DELTA (world) | 0.159 | 32.5 | 55.3 | 24.7 | 34.1 | 68.9 |
| SpatialTracker (world) | 0.125 | 37.7 | 63.9 | 24.9 | 36.9 | 73.5 |
| Shape of Motion | 0.082 | 43.0 | 73.3 | 34.4 | 47.0 | 86.6 |
| MotionScale | **0.070** | 47.0 | **76.4** | 37.7 | **50.6** | **87.4** |
| **OmniX** | 0.162 | 35.1 | 58.2 | 26.8 | 38.2 | 75.1 |
| **GaussTrace (ours)** | **0.070** | **47.2** | 75.9 | **37.8** | 50.1 | **87.4** |

$m_i$ changes. This control distinguishes the proposed Gaussian-level routing from both inherited continuous prior blending and hard static/dynamic assignment.

Table 8: Gate-function comparison on full-sequence NVIDIA Dynamic Scenes under the Gaussian Marbles protocol. All rows share OmniX priors, residual parameterization, PGO schedule, and evaluation views; higher PSNR/SSIM and lower LPIPS are better.

| Gate | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| OmniX native linear ($m_i = \theta_i$) | 22.57 | 0.775 | 0.085 |
| Binary ($m_i = \mathbb{I}[\theta_i \geq \theta_d]$) | 20.48 | 0.715 | 0.094 |
| **Thresholded smoothstep (ours)** | **23.91** | **0.821** | **0.070** |

### A.3.3 iPHONE LONG-RANGE 3D AND 2D TRACKING

We additionally evaluate long-range 3D tracking on iPhone/DyCheck in metric-scale world coordinates. Ground-truth tracks are obtained by lifting annotated 2D keypoints with LiDAR depth, excluding occluded points and points with invalid depth. The direct dense 3D point trajectories predicted by OmniX provide global temporal identities but remain imperfect under challenging motion, depth, and camera ambiguities. GaussTrace initializes every dynamic Gaussian with the OmniX prior center $\mu_{t,i}^{o}$ and corrects it through the trajectory-basis shared translation correction and local residual defined in the main paper, using the multi-view image evidence accumulated by Gaussian optimization. This trajectory-first refinement retains the forward model's long-range correspondence while improving the geometric accuracy of the optimized trajectories. Table 9 jointly reports metric-scale 3D tracking through EPE and $\delta_{3D}$, and projected 2D tracking through TAP-Vid Average Jaccard (AJ), average position accuracy ($\delta_{\text{avg}}$), and occlusion accuracy (OA) (Doersch et al., 2022).

**Baselines.** We compare against reconstruction-based HyperNeRF (Park et al., 2021b), DynIBaR (Li et al., 2023), Deformable-3D-GS (Yang et al., 2024c), DynMF (Kratimenos et al., 2024), Shape of Motion (Wang et al., 2025), and MotionScale (Zhou & Lee, 2026), as well as tracking-based CoTracker (Karaev et al., 2024) and TAPIR (Doersch et al., 2023) lifted with Depth Anything (Yang et al., 2024a), DELTA (Ngo et al., 2025), and SpatialTracker (Xiao et al., 2024). Every tracking result is reproduced by us from the corresponding public implementation under the same iPhone split, hardware, and convergence procedure.

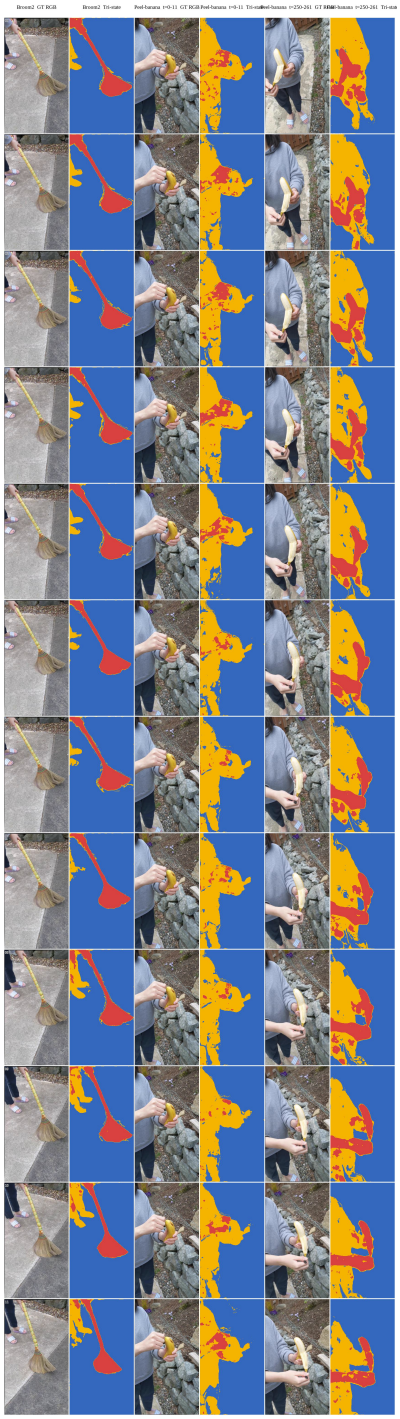

Figure 7: Fixed tri-state partitions on HyperNeRF. Each paired column shows RGB and the corresponding routing over consecutive frames: Broom2 and two distant temporal windows of the Peel Banana validation-rig sequence. Broom2 is shown for qualitative visualization only; all numerical calibration uses the selected `vrig_*` sequences. Blue, yellow, and red denote static, transitional, and dynamic support, respectively.

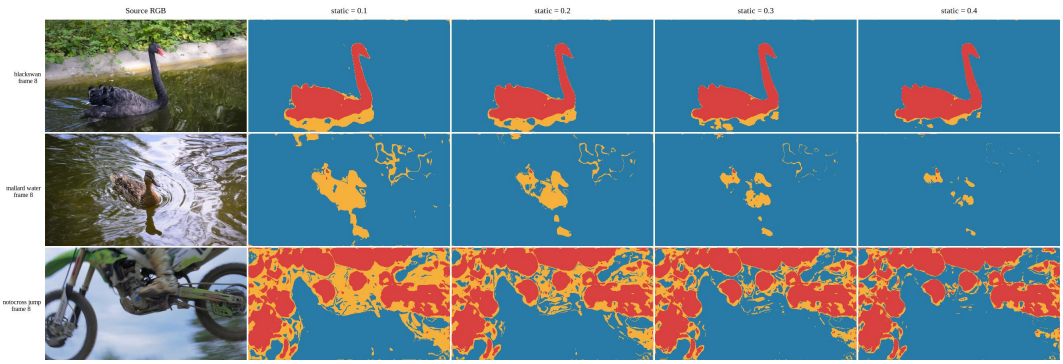


Figure 8: Effect of the static routing threshold on DAVIS score partitions. Blue, yellow, and red denote static, transitional, and dynamic support, respectively; increasing $\theta_s$ progressively freezes ambiguous support that the tri-state formulation otherwise retains for refinement.




Figure 9: Frame-wise OmniX dynamic-score partitions on DAVIS. For Rollerblade, Blackswan, and Motocross-Jump, each left column visualizes the fixed tri-state partition compiled from OmniX dynamic scores across sixteen consecutive frames, and the adjacent right column shows the DAVIS foreground annotation for post-hoc evaluation only.




Figure 10: DAVIS Drift-Turn qualitative comparison over sixteen consecutive frames. Each method occupies two temporal rows; GaussTrace better preserves the fast-moving car and track appearance under coupled camera and object motion.

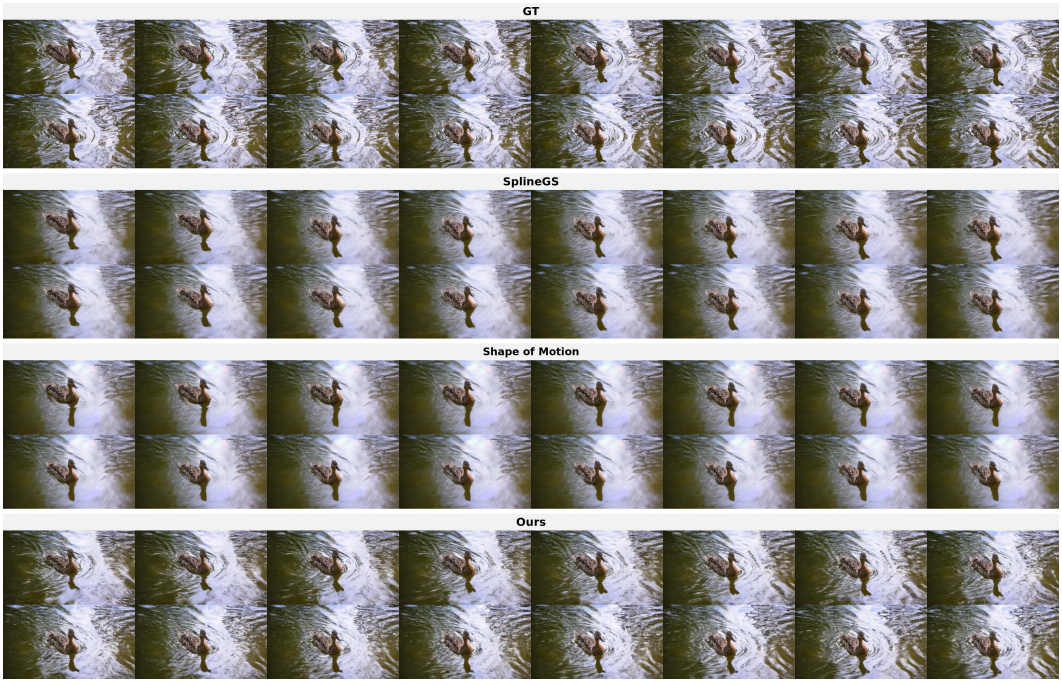


Figure 11: DAVIS Mallard-Water qualitative comparison over sixteen consecutive frames. The mixed water-ripple background challenges strict binary motion support; GaussTrace retains both the duck and the surrounding water appearance.




Figure 12: DAVIS Motocross-Jump qualitative comparison over sixteen consecutive frames. Under rapid viewpoint and articulated object motion, trajectory-first refinement maintains more coherent motorcycle structure and appearance.

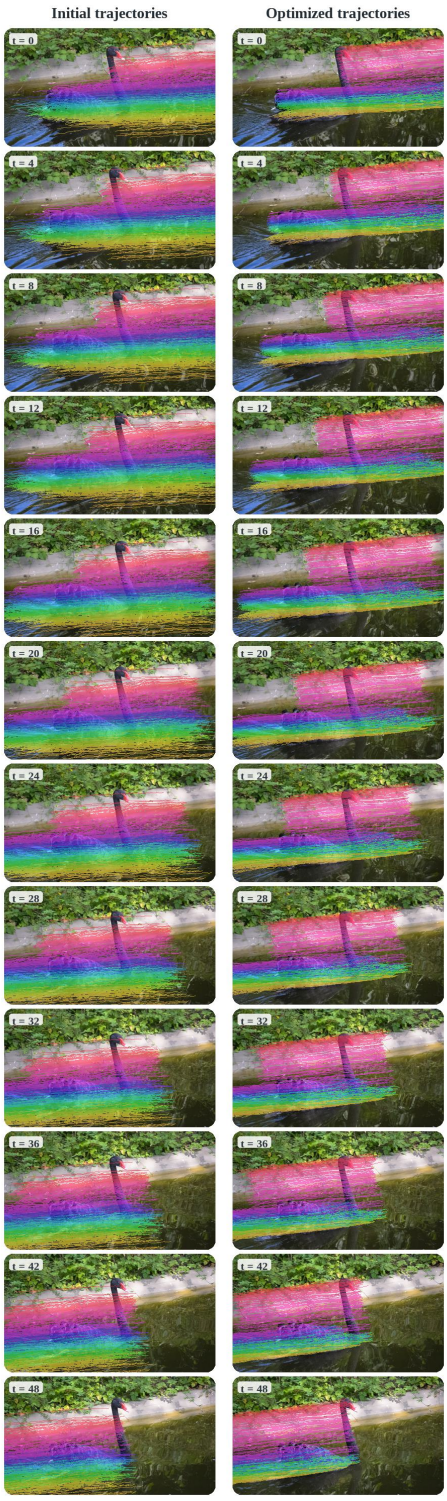


Figure 13: Long-range trajectory refinement on DAVIS Blackswan. Each row is a timestamp, with frozen feed-forward trajectories on the left and trajectories after grouped and point-wise GaussTrace correction on the right. The refined field remains coherent on the swan and surrounding water while correcting accumulated initialization error.

Figure 14 visualizes the dense long-range 3D trajectory fields used by GaussTrace across eight iPhone/DyCheck sequences. Each overlaid curve corresponds to a persistent point identity transported through time, providing substantially denser motion support than a sparse set of isolated tracks. The trajectories cover both articulated foreground regions and their surrounding scene context, enabling the shared translation correction to exploit coherent motion while retaining local residual flexibility at motion boundaries.

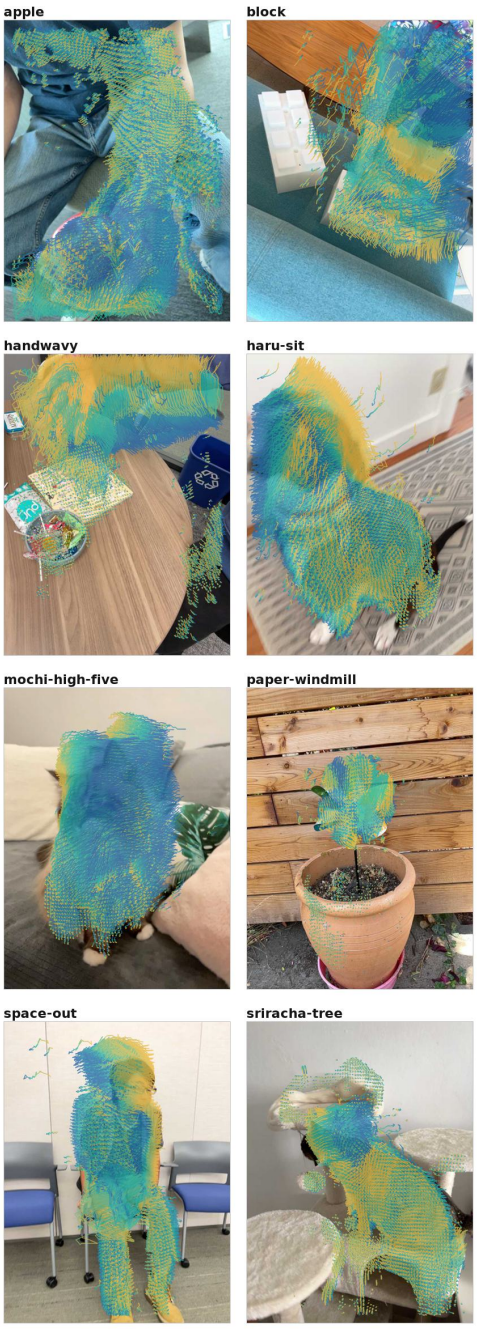


Figure 14: Dense long-range trajectory visualization on eight iPhone/DyCheck sequences. The overlaid trajectories provide persistent motion support across articulated humans, animals, and objects.

Table 10: Stage-B TBMM parameter and memory accounting under the 12-frame NVIDIA camera 0-to-camera 1 setting. $T = 12$ and $G = 32$ for every scene. The local training state includes the FP32 parameter, gradient, and two Adam moments.

| Scene | $N$ | Local $3TN$ | Shared $3TG$ | Local FP32 / training state (MiB) | Peak GPU allocation (GiB) |
|---|---|---|---|---|---|
| Balloon1 | 71,917 | 2,589,012 | 1,152 | 9.88 / 39.51 | 1.036 |
| Balloon2 | 47,013 | 1,692,468 | 1,152 | 6.46 / 25.82 | 0.704 |
| Jumping | 50,718 | 1,825,848 | 1,152 | 6.97 / 27.86 | 0.755 |
| Playground | 66,224 | 2,384,064 | 1,152 | 9.09 / 36.38 | 0.949 |
| Skating | 41,035 | 1,477,260 | 1,152 | 5.64 / 22.54 | 0.626 |
| Truck | 54,922 | 1,977,192 | 1,152 | 7.54 / 30.17 | 0.811 |
| Umbrella | 64,770 | 2,331,720 | 1,152 | 8.90 / 35.58 | 0.936 |

### A.3.4 TBMM Implementation and Resource Accounting

For reproducibility, TBMM constructs the full-sequence motion descriptor of each active trajectory basis by flattening its source-relative rotation-matrix and translation trajectories in temporal order and concatenating the two blocks. We normalize the rotation and translation blocks separately by their global standard deviations over all active bases, timestamps, and corresponding components, without mean centering, per-basis normalization, or whitening. A frozen token descriptor concatenates an $\ell_2$-normalized source-reference token with its temporal mean and is reduced to 16 dimensions by PCA without whitening. We combine this descriptor with the motion feature using candidate semantic weights $\{0.1, 0.25, 0.5, 1.0\}$, retain candidates whose motion weighted SSE is at most $1.1\times$ that of motion-only grouping, and choose the remaining candidate with the smallest semantic weighted SSE. Weighted $k$-means uses the dynamic-support weights defined in the main paper, deterministic weighted farthest-point initialization, and at most 50 weighted Lloyd updates; the resulting $G = 32$ assignments are fixed before RGB optimization.

The temporal coreset changes the long-sequence optimization schedule but not dense residual storage. Stage B stores $3TN$ local and $3TG$ shared scalar entries; these are storage counts, whereas under MATC the effective temporal controls are only the representative entries and all remaining entries are overwritten by interpolation after each update. Independently, grouping reduces the per-time shared-correction decode and its gradient reduction from $O(NK)$ to $O(NG)$, with $K = 516$ and $G = 32$, so this branch is $16.1\times$ narrower in both coefficient-buffer traffic and arithmetic. This does not make the complete motion state $16.1\times$ smaller: the $O(TN)$ local residual and renderer activations remain intact and dominate the total state and wall time. The projection is a hard constraint on parameter values after each update: dense Adam states are implementation storage, not additional temporal controls, because non-representative values cannot persist independently after projection. All local residuals use FP32 and Adam, so the local tensor itself occupies $12TN$ bytes and its parameter, gradient, and two moment buffers occupy approximately $48TN$ bytes during optimization. Table 10 reports this accounting under the 12-frame camera 0-to-camera 1 NVIDIA setting; peak GPU allocation includes the Gaussian representation, residuals, optimizer states, renderer activations, and cameras, whereas the preceding frozen OmniX forward pass is excluded.

### A.3.5 Trajectory-Basis Grouping Efficiency

We vary only the number of groups $G$ used for shared trajectory correction to assess the compactness-quality trade-off of that branch. The $G = 516$ setting optimizes one shared translation correction for every original trajectory basis and therefore serves as the ungrouped reference. We calibrate $G$ only under the official HyperNeRF validation-rig protocol and freeze $G = 32$ before every reported benchmark evaluation. Table 11 reports this development-set group-count sweep.

### A.3.6 Shared and Local Motion-Correction Ablation

We further decompose TBMM by removing either the group-shared translation correction $\Delta\tau_{t,g}$ or the point-wise trajectory residual $\Delta x_{t,i}$, while retaining the other correction and all remaining components. This comparison tests their empirical complementarity without assuming that the ren-

Table 11: Trajectory-basis group-count calibration under the official HyperNeRF validation-rig protocol. $G = 516$ denotes direct shared translation correction without clustering; wall-clock time includes the unchanged renderer and dense local state, and therefore does not isolate the shared-decoder saving.

| **Groups $G$** | **PSNR↑** | **Training Time↓** |
|---|---|---|
| 32 (ours) | 25.86 | 4min |
| 64 | 25.77 | 4min |
| 516 (no grouping) | 25.84 | 4min |

Table 12: TBMM correction decomposition on 12-frame NVIDIA (camera $0 \rightarrow 1$). Higher PSNR/SSIM and lower LPIPS/training time are better.

| **Configuration** | **PSNR↑** | **SSIM↑** | **LPIPS↓** | **Training Time↓** |
|---|---|---|---|---|
| w/o group-shared translation correction | 24.51 | 0.799 | 0.081 | 2min |
| w/o point-wise trajectory residual | 19.56 | 0.692 | 0.121 | 2min |
| **GaussTrace** | 27.64 | 0.894 | 0.060 | 2min |

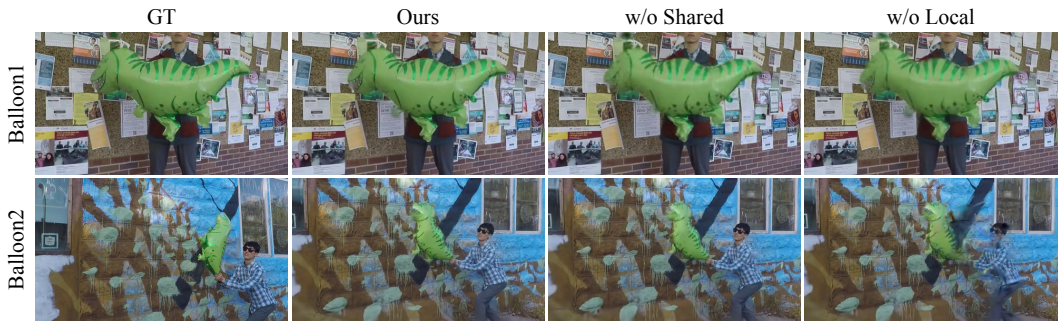


Figure 15: Qualitative TBMM correction decomposition on 12-frame NVIDIA (camera $0 \rightarrow 1$). Columns are GT, Ours, w/o group-shared translation correction, and w/o point-wise trajectory residual; rows are Balloon1 and Balloon2.

dered motion has a unique shared-versus-local decomposition. Table 12 and Figure 15 report the quantitative and qualitative results under the 12-frame NVIDIA camera 0-to-camera 1 setting.

Removing the group-shared translation correction lowers PSNR from 27.64 to 24.51, establishing the value of tying corrections across common basis support beyond point-wise residuals alone. Removing the point-wise trajectory residual causes a larger drop to 19.56 PSNR, showing that shared corrections alone do not supply the required per-Gaussian flexibility. Figure 15 shows the corresponding appearance failures on Balloon1 and Balloon2.

### A.3.7 Random-Seed Stability

To quantify optimization variability, we repeat GaussTrace and every main-paper ablation with three independent optimizer seeds under the NVIDIA Dynamic Scenes Gaussian Marbles protocol. All repetitions retain the same frozen OmniX prior, HyperNeRF-selected hyperparameters, input views, held-out cameras, and optimization schedule. For each seed, we first average the held-out metrics over all NVIDIA scenes, and Table 13 reports the mean and standard deviation across the three resulting dataset averages.

The pronounced degradations from removing TBC, TMCR, or PGO substantially exceed the observed seed variation, supporting the reliability of their quality gains. In contrast, the 0.09-PSNR TBG and 0.05-PSNR MATC differences are comparable to the seed variation; accordingly, we re-

Table 13: Random-seed stability on NVIDIA Dynamic Scenes under the Gaussian Marbles protocol. Entries report mean ± standard deviation over three independent optimizer seeds; higher PSNR/SSIM and lower LPIPS are better.

| Variant | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| **GaussTrace** | 23.91 ±0.05 | 0.821 ± 0.002 | 0.070 ± 0.008 |
| w/o trajectory-basis grouping (TBG) | 23.82 ± 0.03 | 0.813 ± 0.004 | 0.072 ± 0.003 |
| w/o trajectory-basis correction (TBC) | 17.24 ± 0.08 | 0.598 ± 0.010 | 0.248 ± 0.006 |
| w/o tri-state motion-confidence routing (TMCR) | 20.48 ± 0.06 | 0.715 ± 0.011 | 0.094 ± 0.011 |
| w/o motion-aware temporal coreset (MATC) | 23.86 ± 0.04 | 0.820 ± 0.007 | 0.070 ± 0.003 |
| w/o progressive Gaussian optimization (PGO) | 21.39 ± 0.07 | 0.749 ± 0.008 | 0.088 ± 0.005 |

Table 14: Alternative-prior study on 12-frame NVIDIA ($\mathrm{camera}\ 0 \rightarrow 1$). Entries are seven-scene means; higher PSNR and SSIM, lower LPIPS, and lower training time are better.

| Configuration | PSNR↑ | SSIM↑ | LPIPS↓ | Training Time (s)↓ |
|---|---|---|---|---|
| DA3 + CoTracker | 20.59 | 0.675 | 0.103 | 218 |
| 4RC initialization | 25.74 | 0.799 | 0.085 | 191 |
| OmniX + Shape of Motion | 23.47 | 0.732 | 0.096 | 331 |
| **GaussTrace (ours)** | 27.64 | 0.894 | 0.060 | 140 |

gard these variants as reconstruction-quality ties and attribute the 4-versus-7-minute preference to their efficiency benefits rather than to a material quality advantage.

### A.3.8 Alternative Priors and Downstream Refinement

We evaluate robustness to alternative prior interfaces while keeping the renderer, image losses, RoDynRF protocol, and held-out camera evaluation fixed. This is an end-to-end configuration study, not a factor-isolated attribution: a source without a native dynamic score necessarily cannot run the score-dependent routing unchanged.

**Alternative initializations.** For *DA3 + CoTracker*, we lift CoTracker (Karaev et al., 2024) observations with Depth Anything 3 (DA3) (Lin et al., 2026) to initialize the 3D support and point trajectories. For *4RC initialization*, we instead initialize geometry, cameras, and correspondences from the estimates predicted by 4RC (Luo et al., 2026). Neither source provides a native pixel-wise dynamic score, so its corresponding run removes tri-state motion-confidence routing rather than inserting a separately estimated mask. The first two rows therefore assess usable end-to-end alternatives with this paired change, rather than assigning any gap to the prior alone.

For *OmniX + Shape of Motion*, we retain OmniX's native dynamic score to form dynamic support, but replace our trajectory-basis initialization and refinement with the motion-basis construction of Shape of Motion (Wang et al., 2025). Concretely, we cluster vectorized OmniX 3D-trajectory velocities, initialize each shared $\mathrm{SE}(3)$ basis through weighted Procrustes alignment in its cluster, and initialize each point's basis coefficients from its spatial distances to the cluster centers. Thus, this row retains the same OmniX dynamic evidence but jointly changes the downstream motion representation and its initialization. The final row is full **GaussTrace**, which combines OmniX's native dynamic score and trajectory bases with tri-state routing, group-shared translation corrections, local residuals, and progressive optimization.

The DA3 + CoTracker and 4RC rows yield recognizable reconstructions, showing that the downstream objective remains usable with non-OmniX cues; their difference reflects both the prior and the available routing interface. OmniX + Shape of Motion holds the OmniX dynamic evidence




Figure 16: Same-frame qualitative comparison for the alternative-prior study. Columns show Balloon1, Balloon2, and Skating at frame 08; rows show DA3 + CoTracker, 4RC initialization, OmniX + Shape of Motion, and GaussTrace from top to bottom. GaussTrace preserves the moving foreground and background appearance more faithfully across all three sequences.

fixed but jointly changes the subsequent representation and refinement, attaining 23.47 PSNR in 331 seconds versus 27.64 PSNR in 140 seconds for GaussTrace. Together, these results support the effectiveness of the complete trajectory-first configuration, while the controlled gate comparison above isolates the contribution of our routing function.

### A.3.9 Optimized Motion and Geometry Visualization

Figure 17 visualizes the final optimized representation rather than the raw OmniX prediction. For every dynamic Gaussian indexed by $i$, we construct a temporal correction descriptor by stacking the optimized displacement applied to its forward trajectory over the $T$ observed timestamps,

$$\mathbf{g}_i = \mathrm{vec}\left(\left\{ m_i \left( \sum_{g=1}^{G} \rho_{i,g} \Delta\tau_{t,g} + \Delta x_{t,i} \right) \right\}_{t=0}^{T-1}\right). \tag{16}$$

Here $\mathrm{vec}(\cdot)$ concatenates the time-ordered 3D corrections into one vector; $m_i$, $\rho_{i,g}$, $\Delta\tau_{t,g}$, and $\Delta x_{t,i}$ retain the definitions given in the main paper. We project $\{\mathbf{g}_i\}$ onto three principal components and map them to RGB colors in the *PCA Coef.* column. Points with coherent colors share similar optimized temporal corrections, whereas sharp color transitions expose locations where the motion field changes between articulated parts or between dynamic foreground and static support. The *Sparse Trajectories* column shows sampled OmniX prior paths, while the *Dense Trajectories* column shows the per-Gaussian trajectories after applying both the trajectory-basis shared translation correction and the local residual. The *Pred. Depth* column is rendered from the final Gaussian

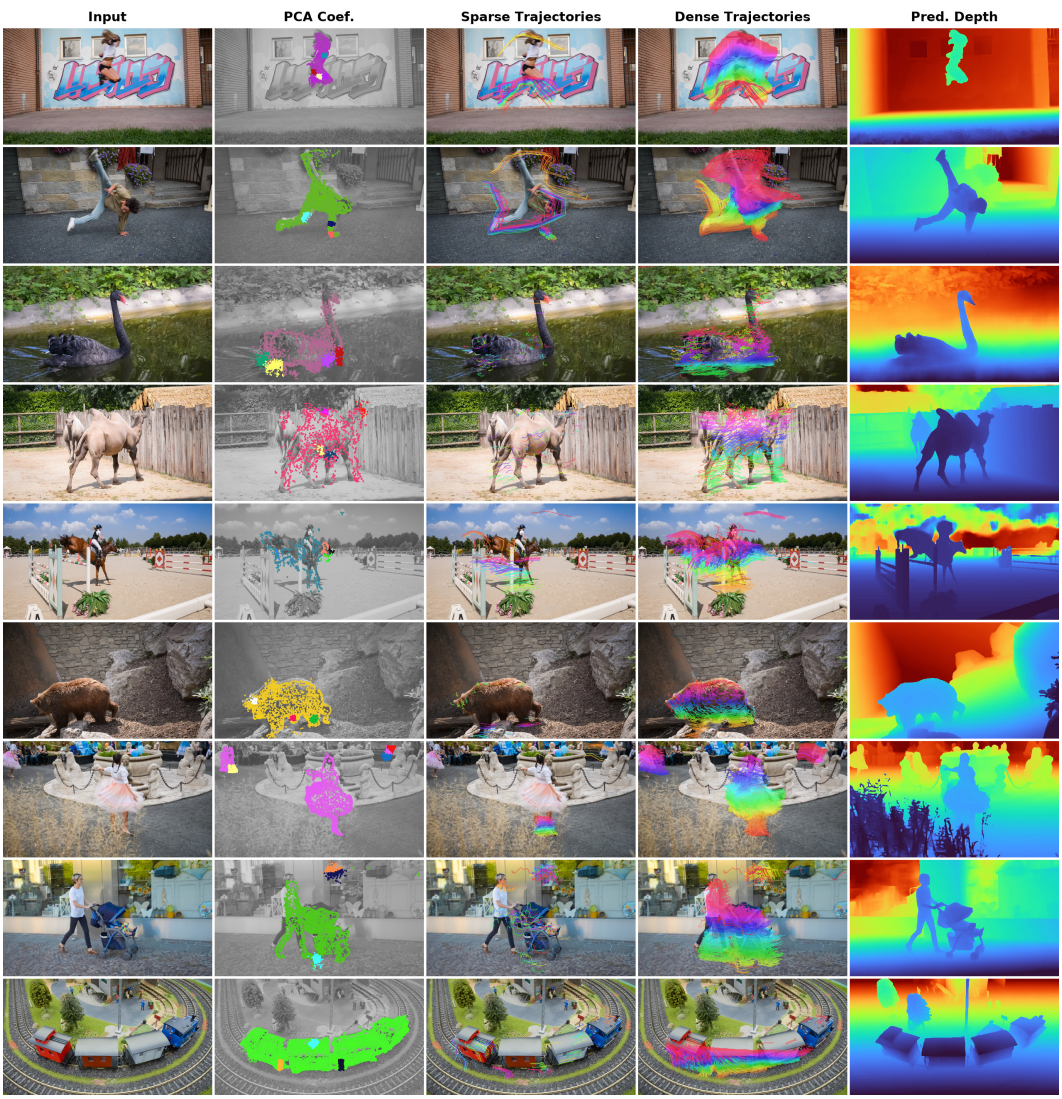


Figure 17: PCA visualization of optimized trajectory corrections and geometry for selected dynamic sequences. The PCA colors encode the optimized temporal correction descriptor in Eq. (16); sparse and dense trajectories respectively show sampled forward paths and final corrected Gaussian trajectories, while predicted depth visualizes the final geometry and camera configuration.

centers and optimized camera poses, and therefore diagnoses the optimized geometry rather than the initial depth map alone.

In Fig. 17, the PCA decomposition separates the regions with the most complex optimized motion, including the skater's legs and torso and the dancer's articulated limbs and contact region, from regions undergoing nearly rigid motion. These localized color changes visualize where the optimized correction differs across points; they do not assert a unique physical separation between shared and local motion.

## A.4 Additional Algorithmic Details

### A.4.1 Optimization-Budget Calibration

Table 15 calibrates nearby Stage-A + Stage-B budgets at 500-update increments for full GaussTrace with MATC, only under the official HyperNeRF validation-rig protocol. It selects the shortest full-model budget whose further extension fails to improve held-out PSNR by at least 0.005 dB; the resulting $4{,}000 + 3{,}000$ schedule is fixed for all reported target scenes. Table 16 applies the identical selection rule to full-control optimization without MATC, and Table 17 records the selected schedules for every main-paper ablation.

Table 15: Full GaussTrace-with-MATC budget calibration under the official HyperNeRF validation-rig protocol. The selected schedule is used on NVIDIA Gaussian Marbles. Higher PSNR/SSIM and lower LPIPS are better.

| Stage-A + Stage-B updates | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| $4{,}000 + 3{,}000$ | 25.86 | 0.83 | 0.18 |
| $3{,}500 + 2{,}500$ | 24.18 | 0.78 | 0.21 |
| $4{,}000 + 2{,}500$ | 25.77 | 0.81 | 0.19 |
| $4{,}000 + 3{,}500$ | 25.80 | 0.82 | 0.18 |
| $4{,}500 + 3{,}000$ | 25.83 | 0.80 | 0.19 |

Table 16: Full-control no-MATC budget calibration under the official HyperNeRF validation-rig protocol. The selected $4{,}000 + 5{,}000$ schedule is the shortest one whose further Stage-B extension improves held-out PSNR by less than 0.005 dB. Higher PSNR/SSIM and lower LPIPS are better.

| Stage-A + Stage-B updates | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| $4{,}000 + 3{,}000$ | 24.65 | 0.77 | 0.23 |
| $4{,}000 + 3{,}500$ | 24.82 | 0.78 | 0.21 |
| $4{,}000 + 4{,}000$ | 24.84 | 0.79 | 0.21 |
| $4{,}000 + 4{,}500$ | 25.05 | 0.79 | 0.20 |
| $4{,}000 + 5{,}000$ | 25.26 | 0.80 | 0.20 |
| $4{,}000 + 5{,}500$ | 25.25 | 0.80 | 0.19 |

Table 17 records the fixed HyperNeRF-selected schedules used for the NVIDIA Gaussian Marbles ablations. For full control, only Stage B extends from 3,000 to 5,000 updates while retaining the same Stage-A stabilization, accounting for the 7-minute no-MATC result. With MATC, extending Stage B beyond 3,000 updates does not improve the held-out PSNR in Table 15; without MATC, the score continues from 25.05 at $4{,}000 + 4{,}500$ to 25.26 at $4{,}000 + 5{,}000$ before saturating at 25.25.

Table 17: Fixed HyperNeRF-selected optimization schedules for the NVIDIA Gaussian Marbles ablations. A dash denotes the one-stage PGO-free optimization.

| Configuration | Stage-A updates | Stage-B updates | Training Time |
|---|---|---|---|
| Full GaussTrace | 4,000 | 3,000 | 4 min |
| w/o TBG | 4,000 | 4,000 | 5 min |
| w/o TBC | 4,000 | 3,000 | 4 min |
| w/o TMCR | 4,000 | 3,000 | 4 min |
| w/o MATC (full control) | 4,000 | 5,000 | 7 min |
| w/o PGO | – | 6,500 | 4 min |

### A.4.2 Shared Frame-0 Projection-Gauge Calibration Details

For completeness, we provide the bounded residual parameterization used in an auxiliary shared frame-0 projection-gauge calibration experiment. This calibration is a standalone auxiliary experi-

ment, not one of the two PGO optimization stages. Let $\mathcal{C}_0 = (R_0, t_0, K_0)$ be the optimized frame-0 camera and let $(\bar{\omega}, \overline{\Delta t}, z_f, \mathbf{z}_p)$ be the four residual variables. The shared projection-camera extrinsics and intrinsic offsets are parameterized as

$$\begin{aligned} R^{\mathrm{q}} &= \mathcal{E}(\bar{\omega}) R_0, & t^{\mathrm{q}} &= t_0 + \overline{\Delta t}, \\ \bar{f} &= \exp\big(\log(f_{\max}) \tanh z_f\big), & \overline{\mathbf{p}} &= (\kappa W, \kappa H) \odot \tanh(\mathbf{z}_p). \end{aligned} \tag{17}$$

Here $\mathcal{E}$ maps the three-dimensional rotation residual $\bar{\omega}$ to a rotation matrix, $f_{\max} > 1$ bounds the focal-length multiplier, $\kappa > 0$ bounds the principal-point displacement, and $\odot$ denotes element-wise multiplication. If $(f_{x,0}, f_{y,0}, p_{x,0}, p_{y,0})$ are the focal lengths and principal point in $K_0$, then $K^{\mathrm{q}}$ uses focal lengths $(\bar{f} f_{x,0}, \bar{f} f_{y,0})$ and principal point $(p_{x,0}, p_{y,0}) + \overline{\mathbf{p}}$. The bounded-residual prior is

$$\mathcal{R}_{\mathrm{gauge}}(\mathcal{C}^{\mathrm{q}}, \mathcal{C}_0) = \frac{1}{3}\|\bar{\omega}\|_2^2 + \frac{1}{3}\|\overline{\Delta t}\|_2^2 + z_f^2 + \frac{1}{2}\|\mathbf{z}_p\|_2^2. \tag{18}$$

Let $\hat{I}_0^{\mathrm{q}}$ denote the image rendered from the frozen scene with the resulting shared camera $\mathcal{C}^{\mathrm{q}} = (R^{\mathrm{q}}, t^{\mathrm{q}}, K^{\mathrm{q}})$; the calibration minimizes

$$\mathcal{L}_{\mathrm{gauge}} = \frac{1}{3HW}\left\|\hat{I}_0^{\mathrm{q}} - I_0\right\|_2^2 + \lambda_{\mathrm{gauge}} \mathcal{R}_{\mathrm{gauge}}(\mathcal{C}^{\mathrm{q}}, \mathcal{C}_0), \tag{19}$$

where $H$ and $W$ are image height and width, $I_0$ is the input frame-0 RGB image, and we use the fixed calibration weight $\lambda_{\mathrm{gauge}} = 10^{-4}$. The calibration is run after all scene variables are frozen and is shared across output timestamps, so it cannot compensate frame-specific reconstruction errors with a time-varying query camera.

### A.4.3 Motion-Aware Temporal Coreset Construction

This section gives the complete deterministic construction of the temporal coreset summarized in the main paper. It is constructed once after Stage-A topology stabilization and before Stage-B optimization, uses neither RGB reconstruction errors nor optimization gradients, and thus cannot tune selected timestamps to a held-out view.

**Dynamic trajectory-component trajectories.** We first assign each Gaussian to its dominant native basis, $\kappa_0(i) = \arg\max_{1 \le k \le K} a_{i,k}$, and retain non-static support

$$\mathcal{U} = \{i \mid m_i > 0\}. \tag{20}$$

Let $\mathcal{B} = \{\kappa_0(i) \mid i \in \mathcal{U}\}$ be the set of nonempty native-basis assignments and $M = |\mathcal{B}|$ its cardinality; relabeling the elements of $\mathcal{B}$ as $j \in \{1, \dots, M\}$ gives the fixed component assignment $\kappa(i)$ used below. Thus, $M$ is video-dependent rather than a fixed hyperparameter, and this assignment is derived from the DTSH interpolation coefficients $a_{i,k}$ solely for temporal selection. If $\mathcal{U} = \emptyset$ or the descriptor has zero variance, we retain all $T$ timestamps. For every trajectory component $j$, we define its eligible support as

$$\mathcal{Q}_j = \{i \in \mathcal{U} \mid \kappa(i) = j\}. \tag{21}$$

Here $m_i$ is the tri-state continuous motion gate and $\emptyset$ the empty set; only nonempty components enter the schedule, so static support cannot affect temporal selection while transitional support retains its fractional gate value. For each remaining component, we compute the prior-trajectory centroid and its normalized importance:

$$\mathbf{z}_{t,j} = \frac{1}{|\mathcal{Q}_j|}\sum_{i \in \mathcal{Q}_j} \mu_{t,i}^{o}, \qquad \widetilde{\nu}_j = \sum_{i \in \mathcal{Q}_j} m_i, \qquad \nu_j = \frac{\widetilde{\nu}_j}{\sum_{\ell : \mathcal{Q}_\ell \neq \emptyset} \widetilde{\nu}_\ell}. \tag{22}$$

The importance $\nu_j$ is the tri-state-gated support mass of a component: it combines support extent with fractional transition confidence, preventing a small collection of weakly supported points from dominating selection.

**Descriptor and noise calibration.** We temporally smooth every centroid using the fixed five-tap binomial kernel $\mathbf{h} = \frac{1}{16}[1, 4, 6, 4, 1]$:

$$\bar{\mathbf{z}}_{t,j} = \sum_{\delta=-2}^{2} h_{\delta+2}\, \mathbf{z}_{\mathrm{clip}(t+\delta,0,T-1),j}, \qquad \mathrm{clip}(x, a, b) = \min\{b, \max\{a, x\}\}. \tag{23}$$

We then form source-relative displacement $\mathbf{d}_{t,j} = \bar{\mathbf{z}}_{t,j} - \bar{\mathbf{z}}_{0,j}$ and temporal velocity $\mathbf{v}_{t,j} = \nabla_t \bar{\mathbf{z}}_{t,j}$, where $\nabla_t$ is a first-order finite difference with one-sided boundary differences. The time-$t$ descriptor concatenates the weighted displacement and velocity of all nonempty components:

$$\mathbf{y}_t = \mathrm{vec}\Big( \big\{ \sqrt{\nu_j}\mathbf{d}_{t,j}, \sqrt{\nu_j}\mathbf{v}_{t,j} \big\}_{j:\mathcal{Q}_j \neq \emptyset} \Big). \tag{24}$$

The square-root factor makes squared descriptor distances additive with respect to component importance. Thus, $\mathbf{y}_t$ is a scene-level translation-motion state rather than a per-Gaussian quantity: it concatenates each eligible component's displacement and velocity at time $t$. It is a temporal control descriptor, not a surrogate for RGB or NVS error and not a complete representation of within-component rotation, visibility, or appearance; those effects remain in the frozen trajectory prior and the subsequent image-supervised refinement. In particular, for two times $t$ and $t'$, the squared Euclidean descriptor distance is the importance-weighted sum of squared component displacement and velocity differences:

$$\|\mathbf{y}_t - \mathbf{y}_{t'}\|_2^2 = \sum_{j:\mathcal{Q}_j \neq \emptyset} \nu_j \Big( \|\mathbf{d}_{t,j} - \mathbf{d}_{t',j}\|_2^2 + \|\mathbf{v}_{t,j} - \mathbf{v}_{t',j}\|_2^2 \Big). \tag{25}$$

Therefore, dominant moving components contribute more strongly to frame selection. Applying the same construction before smoothing gives $\mathbf{y}_t^{\mathrm{raw}}$; the difference $\mathbf{y}_t^{\mathrm{raw}} - \mathbf{y}_t$ is precisely the high-frequency trajectory variation removed by the temporal filter. We use the median magnitude of this removed variation as a sequence-specific forward-noise tolerance:

$$\epsilon = \mathrm{median}_{0 \leq t < T} \, \|\mathbf{y}_t^{\mathrm{raw}} - \mathbf{y}_t\|_2 . \tag{26}$$

The median makes this tolerance robust to a small number of genuine rapid-motion frames, and it is not a ground-truth noise measurement or a learned parameter. This uses the trajectory field itself to distinguish meaningful rapid motion from high-frequency forward jitter, rather than imposing a manually chosen number of retained frames.

**PCA and greedy knot insertion.** We center the descriptor matrix and compute its singular value decomposition:

$$\mathbf{Y} = \begin{bmatrix} (\mathbf{y}_0 - \bar{\mathbf{y}})^\mathsf{T} \\ \vdots \\ (\mathbf{y}_{T-1} - \bar{\mathbf{y}})^\mathsf{T} \end{bmatrix} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}, \qquad \bar{\mathbf{y}} = \frac{1}{T}\sum_{t=0}^{T-1} \mathbf{y}_t. \tag{27}$$

We retain the smallest rank $d_\phi$ satisfying $\sum_{j=1}^{d_\phi} \sigma_j^2 / \sum_j \sigma_j^2 \geq 0.995$, denote the corresponding right singular vectors by $\mathbf{V}_{d_\phi}$, and set $\boldsymbol{\phi}_t = \mathbf{V}_{d_\phi}^\mathsf{T}(\mathbf{y}_t - \bar{\mathbf{y}})$. The low-dimensional $\boldsymbol{\phi}_t$ removes redundant descriptor directions without changing the temporal variation that drives selection.

We initialize $\mathcal{S} = \{0, T-1\}$. For an unselected timestamp $t$, let $l(t)$ and $r(t)$ be its nearest selected timestamps to the left and right, respectively, and define

$$\lambda_t = \frac{t - l(t)}{r(t) - l(t)}, \qquad \widehat{\boldsymbol{\phi}}_t = (1 - \lambda_t)\boldsymbol{\phi}_{l(t)} + \lambda_t \boldsymbol{\phi}_{r(t)}, \qquad e_t = \Big\| \boldsymbol{\phi}_t - \widehat{\boldsymbol{\phi}}_t \Big\|_2 . \tag{28}$$

We insert $t^\star = \arg\max_{t \notin \mathcal{S}} e_t$ and repeat until $\max_{t \notin \mathcal{S}} e_t \leq \eta\epsilon$, where $\eta = 1$ is the default noise multiplier. For $T \leq 16$, the procedure directly returns all timestamps because compression would provide little throughput benefit while increasing temporal aliasing risk; hence all 12-frame NVIDIA sequences, including the rotating Umbrella sequence, use full temporal control. Umbrella therefore evaluates the time-shared covariance approximation under object rotation without conflating it with temporal selection.

**Optimization-time allocation and projection.** The following pseudocode specifies the use of $\mathcal{S}$ during optimization.

1. Assign every full timestamp to its nearest representative, $o(t) = \arg\min_{s \in \mathcal{S}} |t - s|$, and count $n_s = |\{t \mid o(t) = s\}|$.
2. Form a length-$T$ smooth weighted round-robin schedule in which each representative $s$ occurs exactly $n_s$ times.

Table 18: Temporal-selector sweep on the 192-frame NVIDIA Balloon1 sequence under the Gaussian Marbles protocol. Coreset rows use $4{,}000/3{,}000$ updates; full control uses $4{,}000/5{,}000$, selected by the same HyperNeRF $0.005$-dB criterion. Higher speedup and smaller-magnitude $\Delta$PSNR are better; $\Delta$PSNR is held-out NVS PSNR in dB relative to full control, where negative values denote a loss.

| **Selection** | **Speedup** | **$\Delta$PSNR (dB)** |
|---|---|---|
| Full control (baseline) | $1.0\times$ | $0.00$ |
| $0.5$ | $1.55\times$ | $-0.0176$ |
| $0.75$ | $1.76\times$ | $-0.0287$ |
| $1.0$ (default) | $1.34\times$ | $-0.0268$ |
| $1.25$ | $1.83\times$ | $-0.0762$ |
| $1.5$ | $2.08\times$ | $-0.0861$ |
| $2.0$ | $2.03\times$ | $-0.1201$ |
| $3.0$ | $3.54\times$ | $-0.1339$ |
| Random sampling (same cardinality as $\eta = 1.0$) | $1.35\times$ | $-3.2617$ |

3. At each update, draw representatives from this schedule, render their image batch, and back-propagate only through that batch.
4. After the optimizer step, interpolate all non-representative time parameters from their two bracketing representatives.

More specifically, for every time-dependent parameter sequence $\boldsymbol{\xi}_t$, including $\Delta\tau_{t,g}$, $\Delta x_{t,i}$, $\omega_t$, and $\Delta t_t$, the final step applies

$$\boldsymbol{\xi}_t \leftarrow (1-\lambda_t)\boldsymbol{\xi}_{l(t)} + \lambda_t \boldsymbol{\xi}_{r(t)}, \qquad t \notin \mathcal{S}. \tag{29}$$

Consequently, every timestamp remains defined in the final scene, but only representative nodes carry independent temporal degrees of freedom and repeatedly invoke rendering and back-propagation.

**Selector trade-off on a long sequence.** We evaluate the temporal-selector noise multiplier $\eta$ on the 192-frame Balloon1 sequence of NVIDIA Dynamic Scenes under the Gaussian Marbles protocol (Stearns et al., 2024), holding the frozen OmniX prior and all other settings constant; camera 0 supplies the optimization video and synchronized camera 1 provides held-out NVS. Table 18 reports the resulting quality–efficiency trade-off and uses $\eta = 1$ as the default operating point. All coreset rows use the fixed $4{,}000/3{,}000$ full-model budget, whereas full control uses the $4{,}000/5{,}000$ budget selected by the same HyperNeRF $0.005$ dB criterion; reported speedups are measured optimization wall-clock ratios. The random baseline retains exactly the same number of representative timestamps as the default $\eta = 1.0$ selector, differing only in how those timestamps are chosen. At this matched cardinality, random selection loses $3.2617$ dB held-out NVS PSNR, whereas MATC loses only $0.0268$ dB at essentially identical speedup, showing that the gain comes from motion-aware timestamp selection rather than sparse optimization alone.

## A.5 Limitations and Failure Analysis

### A.5.1 Boundary Failure

Figure 18 shows this remaining failure mode at thin or weakly supported dynamic boundaries across six NVIDIA Dynamic Scenes sequences. The blurred contours are consistent with an initialization in which the forward depth–ray geometry and persistent 3D trajectories place boundary support slightly inside the foreground, background, or an intermediate depth layer. At such pixels, the point-level dynamic score can also be ambiguous; once topology has stabilized, the motion gate and the sparse interpolation coefficients $a_{i,k}$ are compiled and fixed, so a boundary Gaussian can retain an imperfect motion-support assignment. Under monocular RGB supervision, this mixed initialization can reduce the photometric loss by adjusting the coverage, opacity, or color of existing Gaussians near the boundary, rather than uniquely recovering a sharp foreground–background separation.

The grouped translation correction $\sum_g \rho_{i,g} \Delta\tau_{t,g}$ couples points with common basis support, while the local residual $\Delta x_{t,i}$ accommodates point-wise deviations not explained by that shared path; both operate within the fixed motion gate and sparse basis support and are constrained by the magnitude and temporal-smoothness regularizers defined in the main paper. The staged schedule deliberately confines topology adaptation to Stage A, because a late topology change would require assigning new Gaussians to motion support and could destabilize the forward trajectory identities. Consequently, the residual optimization can improve projection consistency around an existing boundary, but after Stage B begins it cannot introduce a new topology or motion-support assignment when the required high-frequency geometric evidence is absent from the initialization.

### A.5.2 Failure Under Degraded OmniX Priors

Our trajectory-first formulation also inherits a broader limitation from its frozen feed-forward initialization. When OmniX produces incomplete or inconsistent geometry and trajectory evidence for a challenging input sequence, the subsequent Gaussian optimization starts from an insufficient motion scaffold. Figure 19 shows three such cases, in which the final reconstruction retains some motion structure but remains incomplete, blurred, or spatially inconsistent. The RGB objective can repair modest local deviations, but under monocular supervision and the constrained residual parameterization it cannot reliably recover geometric or correspondence evidence that is absent from the initial OmniX prediction.

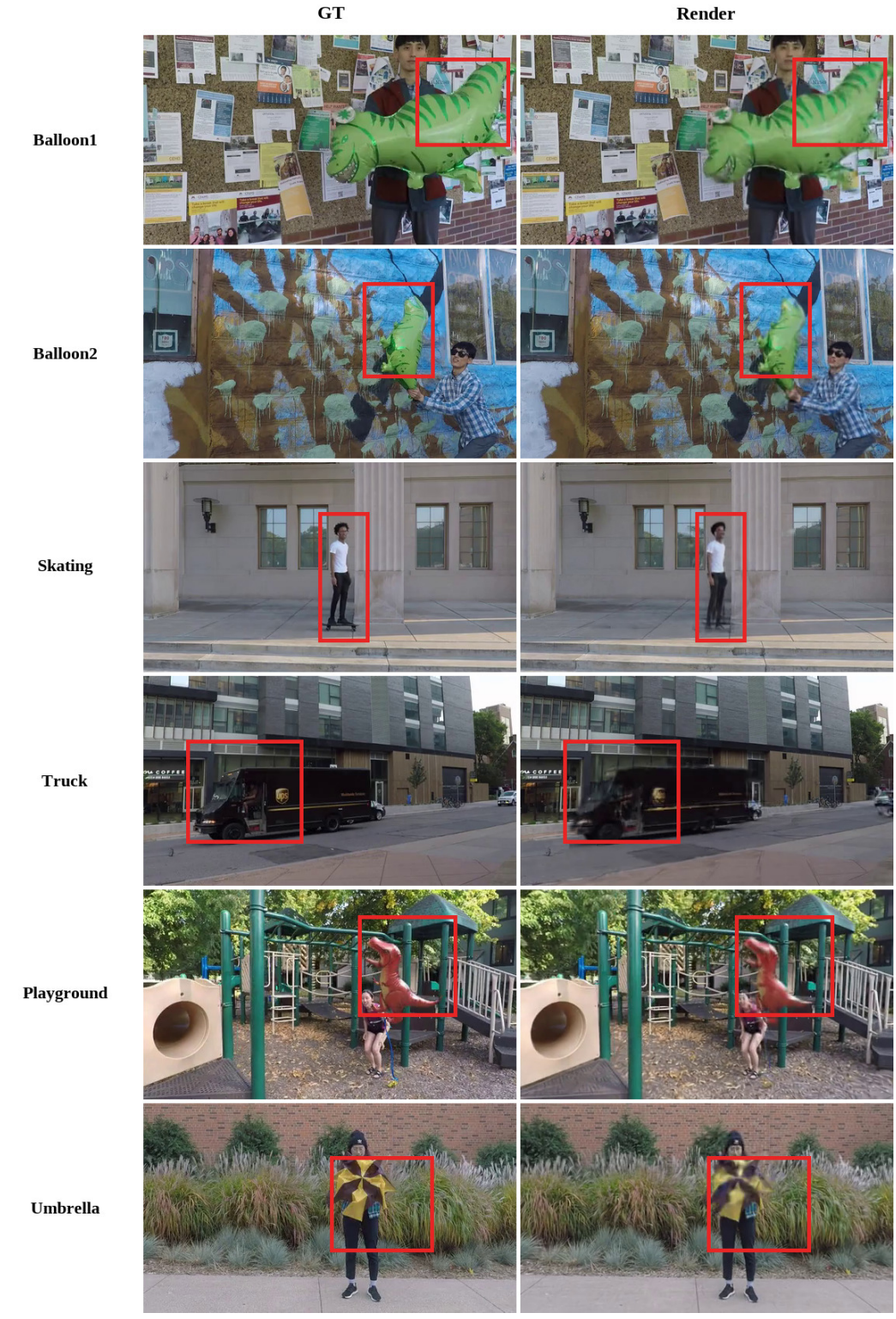


Figure 18: Boundary blur under ambiguous feed-forward geometric support. The red boxes compare ground truth and novel-view synthesis at dynamic object boundaries across six NVIDIA Dynamic Scenes sequences, where small errors in the initial depth, trajectory, or dynamic support can produce blurred contours after optimization.



Figure 19: Failure cases under degraded OmniX priors. Rows show Dog Agility, Motocross Jump, and Motocross Bumps; columns show the input frame, the corresponding OmniX initialization visualization, and the final GaussTrace reconstruction.

### A.6 Additional Details of OmniX-LONG

This section describes the Exact-Stream engineering path used to obtain long-video OmniX (Jiang et al., 2026) priors for GaussTrace. It preserves the frozen OmniX weights, trajectory semantics, per-source top-$20\%$ token-level dynamic-probability indices, and ordered complete-video context, changing only intermediate-tensor materialization and release. It is therefore a memory-lifetime implementation rather than a token-compression, learned motion, or GaussTrace optimization component.

**Motivation: one coherent video forward pass.** Our goal is to obtain the long-range OmniX priors from one coherent forward pass over the whole ordered video, rather than from independently processed temporal chunks. Splitting a video into clips changes the temporal context available to the backbone and the spatiotemporal module at clip boundaries. It consequently requires an additional cross-chunk alignment step for point identities, camera-relative motion, and trajectory coordinates; otherwise, a trajectory estimated on one chunk need not connect continuously to its counterpart in the next chunk. Such boundary-dependent alignment is particularly undesirable here, because GaussTrace consumes the exported trajectories as long-range motion priors for jointly optimizing the same persistent Gaussian identities. Exact-Stream addresses the memory bottleneck without introducing this chunk boundary: it preserves complete-video attention structure while streaming intermediate storage and registered-query readout.

**Execution contract.** Given $T$ input frames, Exact-Stream first executes one frozen OmniX backbone pass over the complete ordered video, and then reads each requested registered source pixel at all target timestamps from the prepared representation. For a source timestamp $s$ and $N_s$ registered continuous image coordinates, the returned trajectory tensor has shape $\mathbf{Y}_s \in \mathbb{R}^{1\times N_s\times T\times 3}$. The source–target tiles bound only the resident query workspace: every tile retains the original source initialization and top-$20\%$ token-level dynamic-probability indices, masked cross-attention reads the selected target-frame memory, and global cross-attention reads the complete ordered $T$-frame memory. Thus, tiling does not turn the input into independently inferred temporal clips.

**Why the native execution exceeds memory.** The released OmniX path reaches a $16.24$ GiB live allocation at the 32-frame offset-head boundary and then requests a further $8.44$ GiB, yielding a $24.68$ GiB OOM demand. This peak is caused by the overlap of the complete feature pyramid, depth and camera geometry, source–target motion fields, and dense offset or trajectory outputs. More intuitively, each of the $T$ frames contributes an $H \times W$ field of image locations, while trajectory inference must relate locations from a source frame to target timestamps. If these source–target relations are materialized densely at once, their storage has a $T \times T \times H \times W$ scale: one temporal factor chooses the source frame and the other chooses the target frame. Consequently, doubling the video length can multiply this source–target portion of the memory by roughly four, before accounting for channels, multiple feature levels, offsets, and decoder activations. The OOM is not attributed to a single unusually large operator; rather, the native path lets these quadratic-scale relations coexist with the complete feature pyramid and geometry-head outputs until the offset head requests its next workspace. Exact-Stream leaves the complete-video backbone computation intact but prevents these later states from being resident simultaneously.

**Lifetime-controlled data flow.** After the backbone exports its four feature-pyramid levels, they are transferred to CPU memory, and the frozen DPT head is evaluated on consecutive eight-frame feature slabs. This slab is an execution granularity rather than a temporal context window: complete-video context remains in the backbone and spatiotemporal module. The geometry results are retained, whereas the temporary slab is released before the next group is processed. The spatiotemporal module then projects key–value states in frame chunks and evaluates one source and a small target block at a time. For the profiled path, the source and target tile sizes are both one; the standard path permits four target frames and reduces this number only when required by available memory. Once the offset statistics for the current source are complete, they are immediately consumed by trajectory readout and released. Finally, bilinear trajectory readout is executed only for requested registered coordinates in batches, and each target-tile result is written out before the next tile is formed.

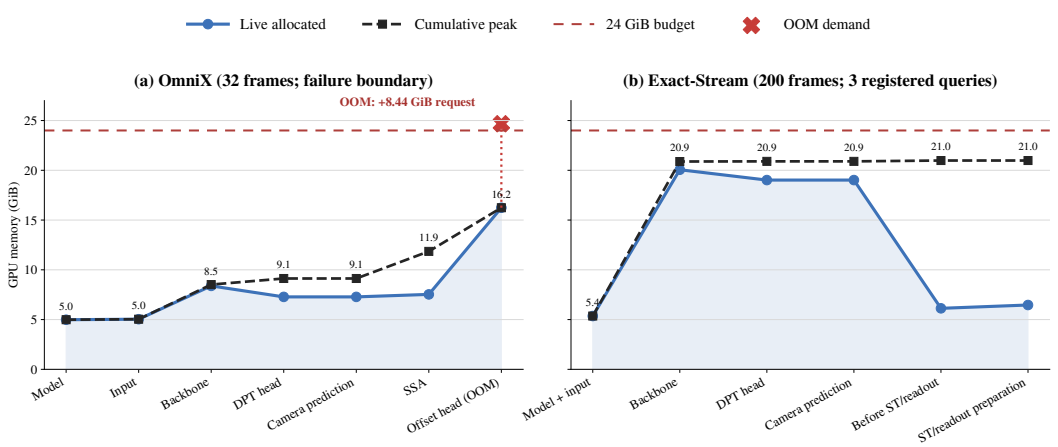


Figure 20: Stage-wise GPU-memory profiles for native OmniX and Exact-Stream. Blue curves show live PyTorch CUDA allocated memory, black dashed curves show cumulative allocated-memory peaks, and the red dashed line marks the 24-GiB device budget. Native OmniX fails at the 32-frame offset-head boundary after an additional 8.44-GiB request beyond 16.24 GiB already allocated. Exact-Stream completes a separate 200-frame execution with three registered queries; CPU feature-pyramid offload and tiled readout lower live allocation to 6.13 GiB before spatiotemporal readout, and the final readout reaches a 21.95-GiB cumulative peak.

Table 19 gives the recorded stage ledger corresponding to Fig. 20.

**Numerical consistency and output boundary.** All changes run in inference mode and reuse the original eight spatiotemporal blocks, four motion projections, four motion levels, and nine-dimensional motion features without token merging, frame subsampling, quantization, or approximate attention. The tiled attention workspace uses an online log-sum-exp reduction, which is algebraically equivalent to full softmax attention while avoiding a resident full query–key score matrix. The recorded $T = 200$, $504 \times 280$ profile reads three registered query trajectories in $261.526$ seconds with a $21.950$ GiB peak allocation; it is a streamed-query memory feasibility profile, not a measurement of a complete Gaussian-initialization query set. Short-sequence parity against the released query path yields relative $\ell_2$ differences of $4.12 \times 10^{-7}$ at $T = 8$ and $4.10 \times 10^{-6}$ at $T = 16$. Exact-Stream is therefore numerically consistent with, rather than bitwise identical to, the released query path. The validated result concerns arbitrary registered pixels, not a materialized all-source, all-pixel $T \times T \times H \times W \times 3$ trajectory volume. For GaussTrace, this path is used only to obtain the requested long-range OmniX priors; the subsequent two-stage Gaussian optimization is separate from Exact-Stream and retains the method described in the main paper.

**End-to-end iPhone runtime.** Unlike the optimization-only Training Time in the main paper, this comparison includes method-specific prior extraction and preprocessing together with per-scene optimization. Table 20 reports the aggregate $8\,\mathrm{min}$ iPhone/DyCheck end-to-end timing for GaussTrace, including requested OmniX priors and subsequent per-scene optimization; it is measured separately from the three-query feasibility profile above.

Table 19: Recorded Exact-Stream $T = 200$ GPU-memory ledger for three registered queries on a 24-GiB GPU. Allocated memory is the live PyTorch CUDA allocation at a stage boundary, and peak is the cumulative maximum from the beginning of the inference call.

| **Stage** | **Allocated (GiB)** | **Cumulative peak (GiB)** |
|---|---|---|
| Model and input | 5.356 | 5.356 |
| Backbone | 20.057 | 20.882 |
| DPT head | 19.018 | 20.898 |
| Camera prediction | 19.018 | 20.898 |
| Before ST/readout | 6.130 | 20.981 |
| After ST/readout preparation | 6.464 | 20.981 |
| After ST/readout | 21.950 | 21.950 |

Table 20: End-to-end iPhone/DyCheck runtime, including method-specific prior extraction, preprocessing, and per-scene optimization. Lower is better.

| **Method** | **Total Time↓** |
|---|---|
| Gaussian Marbles (Stearns et al., 2024) | 5–9 h |
| Gaussian Flow (Lin et al., 2024) | 23 min |
| MoSca (Lei et al., 2025) | 1 h |
| Shape of Motion (Wang et al., 2025) | 1 h |
| SE(3) B-SplineGS (Zhang et al., 2026b) | 30 min |
| SplineGS (Park et al., 2025) | 2 h |
| RoDynRF (Liu et al., 2023) | 12 h |
| MotionScale (Zhou & Lee, 2026) | 2 h |
| **GaussTrace** | 8 min |