RAYCO4D: RAY-CONDITIONED CONTINUOUS MOTION MODEL FOR MONOCULAR 4D DYNAMIC RECONSTRUCTION

Anonymous authors
Paper under double-blind review

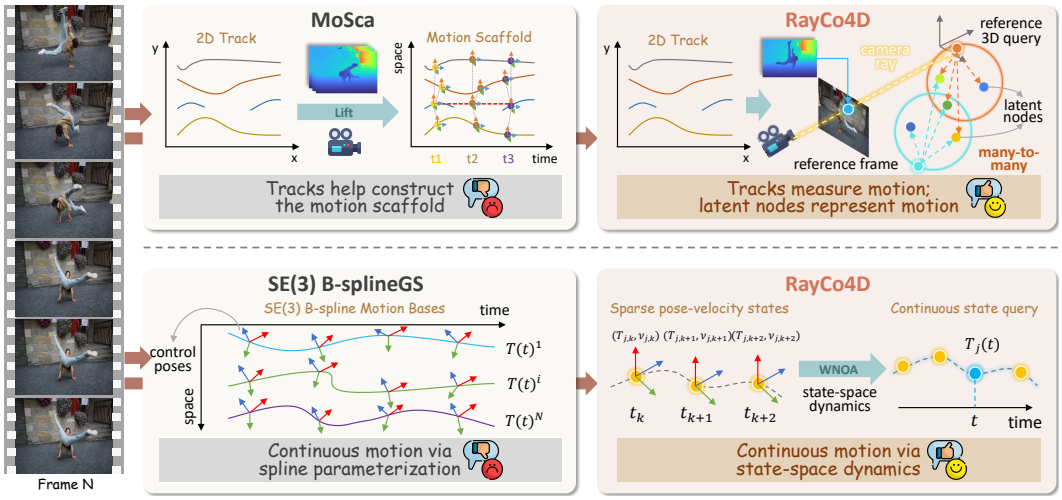


Figure 1: **RayCo4D versus prior track-driven methods. Top:** MoSca (Lei et al., 2025) initializes motion support from lifted tracks; RayCo4D treats tracks as ray-conditioned measurements of shared latent nodes, without one-to-one track–node binding. **Bottom:** SE(3) B-splineGS (Zhang et al., 2026) utilizes continuous motion with spline trajectories, whereas RayCo4D models it using sparse SE(3) pose-velocity states with a WNOA prior for continuous-time queries.

ABSTRACT

Reconstructing dynamic 3D scenes from monocular video requires inferring motion from noisy and intermittently visible observations. Long-range tracks provide valuable geometric constraints, but integrating their fragmented temporal coverage into shared, continuous motion remains challenging. We introduce **RayCo4D**, a ray-conditioned continuous motion model for monocular 4D reconstruction that explicitly separates visual measurements from latent 3D motion states. Each track is associated with a reference-time 3D query constrained to its camera ray, and its image and depth observations constrain shared latent motion nodes through local many-to-many associations. Each node maintains an SE(3) pose and body-frame velocity at sparse support times. A white-noise-on-acceleration process prior couples neighboring states and defines continuous-time interpolation, allowing observations across visibility intervals to jointly constrain the same latent state sequence. The inferred shared motion is transferred to Gaussian primitives through dual-quaternion blending, while reference-anchored local SE(3) residuals capture fine-scale non-rigid deformation beyond the shared scaffold. Experiments on diverse monocular dynamic scenes show that RayCo4D achieves competitive novel-view synthesis quality, improves structural and temporal consistency under rapid motion and incomplete tracks, and supports continuous-time rendering.

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.

1 INTRODUCTION

Reconstructing dynamic 3D scenes from casually captured monocular videos is an important step toward persistent and renderable representations of the physical world. Such representations can support applications, such as immersive content creation (Wu et al., 2025b; Yan et al., 2025), robot perception (Song et al., 2025; Jia et al., 2025), and dynamic scene understanding (Lei et al., 2025; Wang et al., 2025). With only one viewpoint available at each instant, scene geometry, camera motion, and non-rigid deformation are jointly under-constrained. Occlusions, motion blur, and tracking failures further fragment the cross-frame evidence available for motion estimation.

Recent methods build on the explicit representation and efficient rendering of 3D Gaussian Splatting (Kerbl et al., 2023), using long-range tracks (Karaev et al., 2024) and monocular depth to constrain dynamic scene reconstruction. An important modeling choice is how these geometric cues relate to the underlying motion states. MoSca (Lei et al., 2025) initializes motion-node trajectories from lifted tracks, while SE(3) B-splineGS (Zhang et al., 2026) initializes continuous spline control poses from lifted tracklets. Shape of Motion (Wang et al., 2025) uses tracks to supervise shared $\mathrm{SE}(3)$ motion bases, illustrating the value of pooling observations through a common motion representation. For incomplete tracks, the central challenge is to combine measurements across visibility intervals into locally shared motion states with coherent continuous-time evolution.

Our central perspective is that *image tracks are measurements of latent 3D motion, rather than the motion states themselves*. Based on this perspective, we introduce **RayCo4D**, a ray-conditioned continuous motion model for monocular 4D reconstruction. Fig. 1 summarizes its two central design choices: ray-conditioned measurement–state modeling and continuous pose–velocity state estimation. Using camera and depth estimates from Depth Anything 3 (Lin et al., 2025), we constrain each track with a reference-time 3D query and use its visible observations to estimate the shared motion. Track geometry helps initialize a compact motion support, but subsequent optimization does not maintain a dedicated motion state for each track. Instead, each query is associated with a local neighborhood of motion nodes. A track can constrain multiple nodes, while each node aggregates observations from multiple tracks and timestamps. This many-to-many association allows track observations with different visibility intervals to contribute to a shared motion representation.

To estimate this shared motion continuously, each node maintains an $\mathrm{SE}(3)$ pose and body-frame velocity at sparse support times. We connect consecutive states using a white-noise-on-acceleration (WNOA) process prior from continuous-time state estimation (Barfoot, 2024). The process prior couples neighboring pose–velocity states and defines consistent interpolation between support times. Track and depth measurements constrain node transformations at their observation times. As a result, observations from different frames and visibility intervals contribute to the same latent state sequence. Intervals without direct observations remain connected through the process model. Temporal continuity enters both the motion representation and its estimation.

To obtain renderable dynamic geometry, we propagate the node transformations to Gaussian primitives through dual-quaternion blending. This shared scaffold provides a common motion structure, but cannot capture every fine-scale non-rigid deformation. We therefore augment each dynamic Gaussian with a local $\mathrm{SE}(3)$ residual expressed in its reference configuration. The residual transform is exactly the identity at the Gaussian's reference time, ensuring that it introduces no local correction at that anchor. Together with regularization on residual magnitude and variation, this design encourages local corrections to complement, rather than replace, the shared motion. RayCo4D thus combines continuous shared motion with reference-anchored local refinement in a hierarchical dynamic representation.

Our main contributions are summarized as follows: (1) We introduce a **ray-conditioned measurement–state formulation** in which 2D tracks and camera–depth geometry constrain shared latent motion nodes through many-to-many associations, decoupling track identity from motion-state identity. (2) We formulate shared motion using sparse continuous-time $\mathrm{SE}(3)$ pose-velocity states, with WNOA process factors coupling neighboring supports and defining interpolation for measurement evaluation and rendering. (3) We develop a **hierarchical Gaussian motion representation** that combines shared node transport with reference-anchored local $\mathrm{SE}(3)$ residuals, capturing coherent structural motion and fine-scale non-rigid deformation. (4) Experiments demonstrate competitive novel-view synthesis and improved structural and temporal consistency under fast motion and incomplete tracks, together with continuous-time rendering within the reconstructed interval.

2 Related Work

Structured Motion for Dynamic Gaussian Splatting. Deformable 3DGS (Yang et al., 2024) and 4DGS (Wu et al., 2024) model dynamic scenes through time-conditioned Gaussian deformation. Structured representations introduce shared motion across primitives: Shape of Motion (Wang et al., 2025) fits shared $\mathrm{SE}(3)$ motion bases using track and depth supervision, while MoSca (Lei et al., 2025) initializes sparse motion nodes from lifted tracks and propagates their transformations to Gaussians. OriGS (Wu et al., 2025a), RiGS (Wu et al., 2026), and USPLAT4D (Guo et al., 2026) further introduce orientation fields, motion-type decomposition, and uncertainty-guided motion graphs, respectively. Building on shared motion representations, RayCo4D connects ray-conditioned track measurements to latent nodes through local many-to-many associations and augments their motion with reference-anchored Gaussian-level corrections.

Continuous Motion and State-Space Modeling. SplineGS (Park et al., 2025) represents Gaussian positions with cubic Hermite splines, while SE(3) B-splineGS (Zhang et al., 2026) models position and orientation using cumulative $\mathrm{SE}(3)$ B-splines initialized from lifted tracklets. GP-4DGS (Kim et al., 2026) uses variational Gaussian processes to model correlated deformations and their uncertainty. Related dynamical formulations include prediction–observation fusion over Gaussian states in Gaussians on their Way (Deng et al., 2025), learned velocity fields in EvoGS (Asiimwe & Vondrick, 2025), and latent ODE dynamics in ODE-GS (Wang et al., 2026a). RayCo4D adopts a WNOA prior from continuous-time state estimation (Barfoot et al., 2014; Barfoot, 2024) to jointly estimate shared node poses and body velocities at sparse support times. The same pose-velocity model defines temporal process factors and continuous motion queries, allowing ray-based observations to constrain a common latent state sequence. Additional discussion is provided in Appendix A.

3 Method

Given a monocular video $\{I_f\}_{f=0}^{F-1}$, RayCo4D reconstructs a dynamic Gaussian scene by separating visual measurements from shared continuous motion states (Fig. 2). The inputs comprise 2D tracks with visibility and confidence, camera and depth estimates, and epipolar cues derived from RAFT (Teed & Deng, 2020) optical flow. Our representation consists of a sparse set of latent motion nodes, reference-time Gaussian geometry, and per-Gaussian local motion corrections. Through ray-conditioned queries, tracks constrain the motion nodes, whose transformations also drive the Gaussian scene. We formulate ray-conditioned measurements, continuous pose–velocity dynamics, and reference-anchored local corrections.

3.1 Ray-Conditioned Motion Measurements

Long-range tracks connect observations across time, but their visibility, confidence, and depth support vary across frames. The goal is to convert these fragmented observations into geometric constraints on a compact, shared motion field. RayCo4D anchors each track to a camera ray in a reliable reference frame and associates its 3D query with nearby latent motion nodes. The resulting node-induced motion is then evaluated against image and depth measurements.

Ray-constrained queries (RCQ). To couple a 2D track with the 3D motion field, we first assign it a reference location in world space. For each track i, let $\mathbf{u}_{i,f}$ denote its observed pixel coordinate at frame f, and r_i a visible reference frame selected based on observation reliability. We represent the reference-time track geometry as a 3D query $\mathbf{q}_i$, whose position is constrained and optimized along the corresponding camera ray:

$$z_i = z_{\min} + (d_i^0 - z_{\min})\exp(\eta_i), \qquad \mathbf{q}_i = \mathbf{c}_{r_i} + z_i \mathbf{R}_{r_i}^c \mathbf{K}_{r_i}^{-1} \widetilde{\mathbf{u}}_{ir_i}, \tag{1}$$

where $\widetilde{\mathbf{u}}$ denotes homogeneous pixel coordinates, $z_{\min} > 0$ is a depth floor, and η_i is an optimizable scalar initialized to zero. Here, $\mathbf{c}_{r_i}$, $\mathbf{R}_{r_i}^c$, and $\mathbf{K}_{r_i}$ denote the world-space camera center, camera-to-world rotation, and intrinsics at frame r_i, respectively. The initial depth d_i^0 is taken from a reliable depth observation. For tracks without valid depth, it is initialized from a front-facing motion node whose projection is close to the reference pixel. The unnormalized ray in Eq. (1) preserves the camera-axis depth convention. Importantly, η_i adjusts reference geometry and it does not parameterize time-varying track motion.

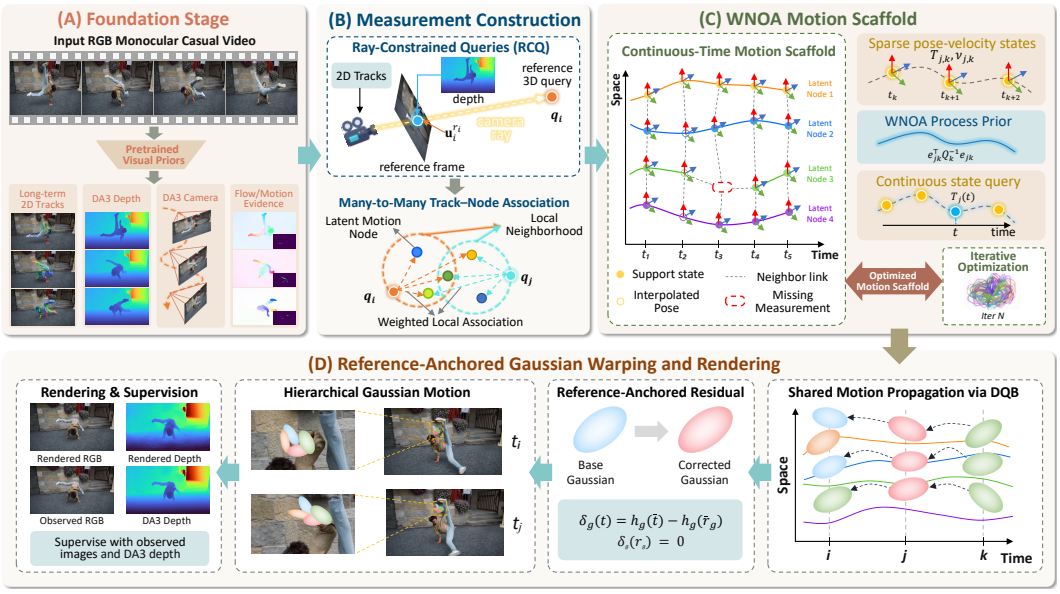


Figure 2: **Overview of RayCo4D.** Given a monocular video, RayCo4D constructs ray-conditioned 3D queries from 2D tracks and camera-depth estimates to constrain shared latent motion nodes. Sparse SE(3) pose–velocity states are coupled by a WNOA prior to model continuous motion. The inferred motion is propagated to Gaussians via dual-quaternion blending, while reference-anchored local residuals capture fine-grained non-rigid deformation for dynamic rendering and optimization.

Measurement–state decoupling (MSD). Let $\mathbf{T}_j(t) = [\mathbf{R}_j(t), \mathbf{p}_j(t)] \in \mathrm{SE}(3)$ denote the continuous pose of latent node j. For each query, we select its K nearest nodes at the reference time and compute normalized radial weights:

$$\mathcal{N}_i = \mathrm{KNN}_j\big(\mathbf{q}_i, \mathbf{p}_j(r_i)\big), \qquad a_{ij} = \frac{\exp\!\big(-\|\mathbf{q}_i - \mathbf{p}_j(r_i)\|_2^2/(2\sigma_j^2)\big)}{\sum_{\ell\in\mathcal{N}_i}\exp\!\big(-\|\mathbf{q}_i - \mathbf{p}_\ell(r_i)\|_2^2/(2\sigma_\ell^2)\big)}, \qquad j \in \mathcal{N}_i, \quad (2)$$

where σ_j controls the spatial influence of node j. Candidate neighborhoods are periodically refreshed as geometry changes, while the weights are recomputed from the current query and node positions. Queries in the same moving region consequently have overlapping node support, allowing their observations to jointly shape the local motion.

We combine the relative transformations of these nodes using dual-quaternion blending (DQB) (Kavan et al., 2007). The predicted world-space position of track i at time t is:

$$\widehat{\mathbf{x}}_i(t) = \mathrm{DQB}\Big(\big\{a_{ij}, \mathbf{T}_j(t)\mathbf{T}_j(r_i)^{-1}\big\}_{j\in\mathcal{N}_i}\Big)\cdot \mathbf{q}_i, \quad (3)$$

where $\cdot$ denotes the rigid action on a 3D point. Each observation constrains a local blend of node states, allowing shared nodes to aggregate evidence across tracks and time. This decouples track identity from latent motion-state identity.

Measurement factors. Let $\mathcal{O}_u$ denote visible track observations and $\mathcal{O}_d \subseteq \mathcal{O}_u$ those with valid depth. Keeping the observed coordinates, visibility, and confidence fixed, we constrain the predicted query positions with the confidence-weighted reprojection loss:

$$\mathcal{L}_{\mathrm{2D}} = \frac{1}{Z_u}\sum_{(i,f)\in\mathcal{O}_u} c_{if}\,\rho_2\!\left(\frac{\pi_f(\widehat{\mathbf{x}}_i(f)) - \mathbf{u}_{if}}{\sigma_u}\right), \qquad Z_u = \sum_{(i,f)\in\mathcal{O}_u} c_{if}. \quad (4)$$

Here c_{if} is track confidence, π_f is camera projection, σ_u is the image-space noise scale, and ρ_2 averages a componentwise smooth-ℓ_1 penalty. Observations in $\mathcal{O}_d$ also contribute a relative camera-axis depth loss weighted by track confidence and depth reliability, denoted $\mathcal{L}_{\mathrm{depth}}$; visible observations without depth still contribute to Eq. (4).

3.2 CONTINUOUS-TIME LATENT MOTION WITH WNOA

The measurement factors above reveal motion at a sparse set of observation times. The latent motion representation must ensure temporal coherence and provide node transformations at arbitrary times. RayCo4D therefore represents each node with sparse $\mathrm{SE}(3)$ pose–velocity states and connects adjacent supports using a white-noise-on-acceleration(WNOA) process. The same continuous model is used for measurement evaluation, Gaussian motion, and subframe rendering.

Sparse pose–velocity states. At each support time, we represent each node by its pose and body-frame velocity to capture both configuration and local motion. For support times $0 = \tau_0 < \cdots < \tau_{S-1} = F - 1$, we define:

$$\mathcal{X} = \{\mathbf{s}_{jk} = (\mathbf{T}_{jk}, \boldsymbol{\nu}_{jk})\}_{j,k}, \qquad \mathbf{T}_{jk} \in \mathrm{SE}(3), \quad \boldsymbol{\nu}_{jk} \in \mathbb{R}^6. \tag{5}$$

where $\mathbf{T}_{jk} = \mathbf{T}_j(\tau_k)$ and $\boldsymbol{\nu}_{jk}$ is the body-frame velocity. We order the translational components before the angular components and use the convention $\dot{\mathbf{T}}_j(t) = \mathbf{T}_j(t)\boldsymbol{\nu}_j(t)^\wedge$. The support times are sampled at a fixed frame stride and include both sequence endpoints. Both pose and velocity are jointly optimized using visual measurements.

WNOA process model. The sparse states require a temporal relation that determines how motion evolves between consecutive supports. We impose a locally linear WNOA prior on $\mathrm{SE}(3)$ (Anderson & Barfoot, 2015). For the interval $[\tau_k, \tau_{k+1}]$, let $\Delta_k = \tau_{k+1} - \tau_k, \boldsymbol{\xi}^+_{jk} = \mathrm{Log}\big(\mathbf{T}^{-1}_{jk}\mathbf{T}_{j,k+1}\big)^\vee$ denote the relative endpoint pose in the tangent space centered at $\mathbf{T}_{jk}$. Because the endpoint body velocities live in different local frames, we express the terminal velocity in the same tangent space using $\widetilde{\boldsymbol{\nu}}_{j,k+1} = \mathbf{J}_r(\boldsymbol{\xi}^+_{jk})^{-1}\boldsymbol{\nu}_{j,k+1}$, where $\mathbf{J}_r$ is the right Jacobian of $\mathrm{SE}(3)$.

WNOA models the local acceleration as zero-mean white noise. Its mean evolution follows constant velocity, while its covariance describes how deviations from that motion accumulate over the interval. The resulting neighboring-state factor is:

$$\mathbf{e}_{jk} = \begin{bmatrix} \boldsymbol{\xi}^+_{jk} - \Delta_k \boldsymbol{\nu}_{jk} \\ \widetilde{\boldsymbol{\nu}}_{j,k+1} - \boldsymbol{\nu}_{jk} \end{bmatrix}, \qquad \mathcal{L}_{\mathrm{WNOA}} = \frac{1}{3M(S-1)} \sum_{j=1}^{M} \sum_{k=0}^{S-2} \mathbf{e}^\top_{jk} \mathbf{Q}^{-1}_k \mathbf{e}_{jk}. \tag{6}$$

Here, M denotes the number of latent motion nodes.The two blocks of $\mathbf{e}_{jk}$ measure the endpoint pose discrepancy relative to the initial-velocity prediction and the velocity change in common local coordinates, respectively. The analytic covariance $\mathbf{Q}_k$ contains position, velocity, and cross-covariance blocks that scale respectively with $\Delta_k^3/3$, Δ_k, and $\Delta_k^2/2$. This time-dependent scaling permits larger motion variation across longer intervals while maintaining tighter coupling between nearby supports. We use separate spectral densities for translational and rotational acceleration and normalize translation using the spatial scale of the motion scaffold. The complete state-space derivation and the explicit form of $\mathbf{Q}_k$ are given in Appendix Sec. C.

Continuous state evaluation. Track measurements and Gaussian rendering generally occur between support times, so the model must provide a pose at every queried timestamp. For $t \in [\tau_k, \tau_{k+1}]$ and $s = (t - \tau_k)/\Delta_k$, the conditional mean of the WNOA process reduces to cubic Hermite interpolation in the local tangent space:

$$\overline{\boldsymbol{\xi}}_{jk}(t) = h_{10}(s)\Delta_k\boldsymbol{\nu}_{jk} + h_{01}(s)\boldsymbol{\xi}^+_{jk} + h_{11}(s)\Delta_k\widetilde{\boldsymbol{\nu}}_{j,k+1}, \qquad \mathbf{T}_j(t) = \mathbf{T}_{jk}\,\mathrm{Exp}\big(\overline{\boldsymbol{\xi}}_{jk}(t)^\wedge\big), \tag{7}$$

where h_{10}, h_{01}, and h_{11} are the standard cubic Hermite basis functions detailed in Appendix Sec. C.3. Eq. (7) matches the endpoint poses and body velocities in the local chart. Measurements from different tracks jointly refine the continuous node trajectories used for Gaussian motion.

3.3 HIERARCHICAL GAUSSIAN RECONSTRUCTION

RayCo4D propagates continuous node motion to Gaussian primitives and introduces reference-anchored local corrections for each primitive. Shared motion captures coherent structure-level dynamics, while local corrections model fine-scale non-rigid deformation. Both components are jointly optimized with Gaussian geometry and appearance under visual and geometric supervision.

Shared motion propagation. Each Gaussian g is stored at a reference time r_g with reference pose $\mathbf{P}_g^{\mathrm{ref}} = [\mathbf{R}_g^{\mathrm{ref}}, \boldsymbol{\mu}_g^{\mathrm{ref}}]$. We associate its reference center $\boldsymbol{\mu}_g^{\mathrm{ref}}$ with a local node set $\mathcal{N}_g$ using the geometric weighting rule in Eq. (2). These geometric weights provide the initial Gaussian-to-node binding. During photometric fitting, Gaussian-specific logit offsets b_{gj} refine these weights:

$$\overline{a}_{gj} = \mathrm{softmax}_{j\in\mathcal{N}_g}\left(\log a_{gj}^{\mathrm{geo}} + b_{gj}\right), \qquad \mathbf{A}_g(t) = \mathrm{DQB}\Big(\left\{\overline{a}_{gj}, \mathbf{T}_j(t)\mathbf{T}_j(r_g)^{-1}\right\}_{j\in\mathcal{N}_g}\Big), \quad (8)$$

where $\mathbf{A}_g(t) \in \mathrm{SE}(3)$ is the shared transformation induced by the continuous node trajectories. The offsets are initialized to zero and activated after the early shared-motion fitting stage. Candidate nodes are refreshed as the scene structure evolves, while the binding for each forward evaluation is computed from the Gaussian's reference geometry. Nearby Gaussians therefore retain overlapping node support and inherit a spatially coherent motion field.

Anchored residual motion (ARM). We equip each dynamic Gaussian with a lightweight local motion curve $\mathbf{h}_g : [0,1] \to \mathbb{R}^6$, represented by a small set of cubic Hermite controls. For $\bar{t} = t/\max(F-1,1)$ and $\bar{r}_g = r_g/\max(F-1,1)$, anchored residual motion (ARM) is defined as:

$$\boldsymbol{\delta}_g(t) = \mathbf{h}_g(\bar{t}) - \mathbf{h}_g(\bar{r}_g), \qquad \mathbf{E}_g(t) = \mathrm{Exp}\big([\mathbf{D}_\ell \boldsymbol{\delta}_g(t)]^\wedge\big), \qquad \mathbf{D}_\ell = \mathrm{diag}(\ell\mathbf{I}_3, \mathbf{I}_3), \quad (9)$$

where ℓ is the spatial unit of the motion scaffold. The translational components of the curve are expressed in scaffold units, and the rotational components are expressed in radians. By construction, $\boldsymbol{\delta}_g(r_g) = \mathbf{0}$ and $\mathbf{E}_g(r_g) = \mathbf{I}$ for any control values. The residual is expressed in the Gaussian's own reference coordinate frame and applied before shared motion transport:

$$\mathbf{P}_g(t) = \mathbf{A}_g(t)\,\mathbf{P}_g^{\mathrm{ref}}\,\mathbf{E}_g(t). \quad (10)$$

In Eq. (10), $\mathbf{E}_g(t)$ first adjusts the primitive in its local reference frame, $\mathbf{P}_g^{\mathrm{ref}}$ places it in the reference scene, and $\mathbf{A}_g(t)$ transports it through the shared continuous motion. The two identity conditions at r_g preserve the stored reference pose: $\mathbf{P}_g(r_g) = \mathbf{P}_g^{\mathrm{ref}}$. Ray-conditioned track factors supervise the shared node motion encoded by $\mathbf{A}_g(t)$, while image and surface losses act on the composed Gaussian pose $\mathbf{P}_g(t)$. Once the scaffold has stabilized, we enable the local controls and regularize their magnitude, curvature, and neighborhood consistency. The full Hermite parameterization and regularization terms are provided in Appendix Sec. D.2.

Dynamic rendering and optimization. For $\mathbf{P}_g(t) = [\mathbf{R}_g(t), \boldsymbol{\mu}_g(t)]$, the translation and rotation determine the Gaussian center and orientation at time t. Each primitive additionally maintains its learned scale, opacity, and spherical-harmonic appearance parameters. Dynamic Gaussians and a time-independent static Gaussian component are rendered jointly with differentiable Gaussian splatting (Kerbl et al., 2023). Since both $\mathbf{T}_j(t)$ and $\mathbf{E}_g(t)$ admit continuous-time evaluation, the reconstructed scene can be rendered at any real-valued time inside the observed interval. We optimize the scene using the grouped objective:

$$\mathcal{L} = \mathcal{L}_{\mathrm{scene}} + \lambda_u\mathcal{L}_{\mathrm{2D}} + \lambda_d\mathcal{L}_{\mathrm{depth}} + \lambda_W\mathcal{L}_{\mathrm{WNOA}} + \lambda_A\mathcal{L}_{\mathrm{ARAP}} + \mathcal{L}_{\mathrm{local}} + \mathcal{L}_{\mathrm{cal}} + \lambda_{\mathrm{warp}}\mathcal{L}_{\mathrm{warp}}. \quad (11)$$

Here, $\mathcal{L}_{\mathrm{scene}}$ collects the enabled image reconstruction, rendered-depth, surface-correspondence, and coverage/ownership terms applied to the final rendered scene. The measurement losses $\mathcal{L}_{\mathrm{2D}}$ and $\mathcal{L}_{\mathrm{depth}}$ connect the original track observations to the shared node states, and $\mathcal{L}_{\mathrm{WNOA}}$ couples those states across support times. The ARAP term preserves the spatial structure of the motion scaffold (Lei et al., 2025), while $\mathcal{L}_{\mathrm{local}}$ regularizes the per-Gaussian residual curves. When enabled, $\mathcal{L}_{\mathrm{cal}}$ regularizes camera-pose and depth corrections relative to the input estimates, while $\mathcal{L}_{\mathrm{warp}}$ evaluates adjacent-frame appearance using predicted Gaussian motion. Appendix D.5 provides the expanded definitions of the local and calibration terms. RayCo4D progressively optimizes the shared motion before introducing the more flexible Gaussian-level variables. Lifted tracks initialize the motion scaffold, with short gaps completed only when constructing candidate node paths.

4 Experiments

4.1 Experimental Setup

Implementation details. We use DA3 (Lin et al., 2025) for camera and depth initialization, BootsTAPIR (Doersch et al., 2024) for long-range tracks, and RAFT (Teed & Deng, 2020) for epipolar

Table 1: **Novel-view synthesis and component ablations.** DyCheck metrics are computed within the co-visible regions, while NVIDIA metrics follow the standard held-out-view protocol. The lower block reports component ablations on both datasets; underlines mark the best baseline.

	DyCheck			NVIDIA		
Method	mPSNR↑	mSSIM↑	mLPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MoSca (Lei et al., 2025)	19.86	0.702	0.298	25.89	0.839	0.079
Shape of Motion (Wang et al., 2025)	18.63	0.674	0.321	24.89	0.732	0.113
OriGS (Wu et al., 2025a)	19.97	0.707	0.307	26.53	0.850	0.068
RiGS (Wu et al., 2026)	18.89	0.689	0.354	25.24	0.793	0.098
SE(3) B-splineGS (Zhang et al., 2026)	20.53	0.713	0.276	26.78	0.848	0.063
USPLAT4D (Guo et al., 2026)	<u>20.64</u>	<u>0.721</u>	<u>0.263</u>	<u>26.99</u>	<u>0.853</u>	<u>0.059</u>
RayCo4D (w/o MSD)	19.62	0.691	0.313	26.61	0.845	0.062
RayCo4D (w/o RCQ)	19.78	0.699	0.299	26.72	0.851	0.057
RayCo4D (w/o WNOA factor)	19.94	0.703	0.283	26.77	0.854	0.055
RayCo4D (w/o ARM)	20.79	0.738	0.247	27.77	0.868	0.049
RayCo4D (Full)	**20.97**	**0.743**	**0.241**	**27.94**	**0.872**	**0.045**

and adjacent-frame flow-consistency cues. We optimize with Adam (Kingma & Ba, 2015) on a single RTX PRO 6000 Blackwell. Motion-scaffold and reference-query depth optimization takes 4,000 iterations, followed by 8,000 iterations of Gaussian warm-up, joint scene–motion refinement, and structure-frozen fitting. The motion scaffold remains fixed during Gaussian warm-up. Gaussian binding corrections, local residuals, and optional camera–depth calibration are activated by stage. Pose–velocity supports are placed every two frames, including both sequence endpoints. The training stages and parameter activation schedule are described in Appendix D.7.

Datasets and metrics. We evaluate novel-view synthesis on DyCheck (Gao et al., 2022) and NVIDIA (Yoon et al., 2020). DyCheck provides spatially separated iPhone views with limited overlap; NVIDIA provides held-out synchronized camera views from monocular training sequences. We report peak signal-to-noise ratio (PSNR), structural similarity (SSIM) (Wang et al., 2004), and learned perceptual image patch similarity (LPIPS) (Zhang et al., 2018). On DyCheck, we report mPSNR, mSSIM, and mLPIPS over co-visible regions; NVIDIA follows its standard held-out-view protocol. We also evaluate motion correspondence on DyCheck using PCK-T, comparing projected 3D motion queries with target-frame point observations. On NVIDIA, we assess reconstruction at intermediate timestamps withheld during training. DAVIS (Perazzi et al., 2016) supplies qualitative tests of rapid motion, occlusion, and non-rigid deformation, as synchronized novel-view ground truth is unavailable. Dataset and evaluation details are given in Appendix D.9.

Baselines. We compare with six methods: the shared-motion approaches MoSca (Lei et al., 2025) and Shape of Motion (Wang et al., 2025), the continuous-time baseline SE(3) B-splineGS (Zhang et al., 2026), and OriGS (Wu et al., 2025a), RiGS (Wu et al., 2026), and USPLAT4D (Guo et al., 2026). Additional baseline details are provided in Appendix D.8.

4.2 Comparison with the State of the Art

Quantitative comparison. Tab. 1 compares RayCo4D with six representative baselines on DyCheck and NVIDIA. RayCo4D achieves the best results across all reported metrics on both datasets. Compared with USPLAT4D, the strongest baseline, RayCo4D improves DyCheck mPSNR by 0.33 dB and mSSIM by 0.022, while reducing mLPIPS by 0.022 (8.4%). On NVIDIA, it achieves larger gains in PSNR (+0.95 dB) and perceptual quality, reducing LPIPS by 0.014 (23.7%), alongside a 0.019 improvement in SSIM. These consistent improvements across fidelity, structural similarity, and perceptual metrics demonstrate the effectiveness of RayCo4D under both evaluation protocols.

Qualitative comparison. Fig. 3 presents qualitative comparisons on DyCheck, NVIDIA, and DAVIS. On DyCheck and NVIDIA, RayCo4D produces sharper object boundaries and more faithful fine details, with reduced blurring and ghosting in dynamic regions. The magnified regions

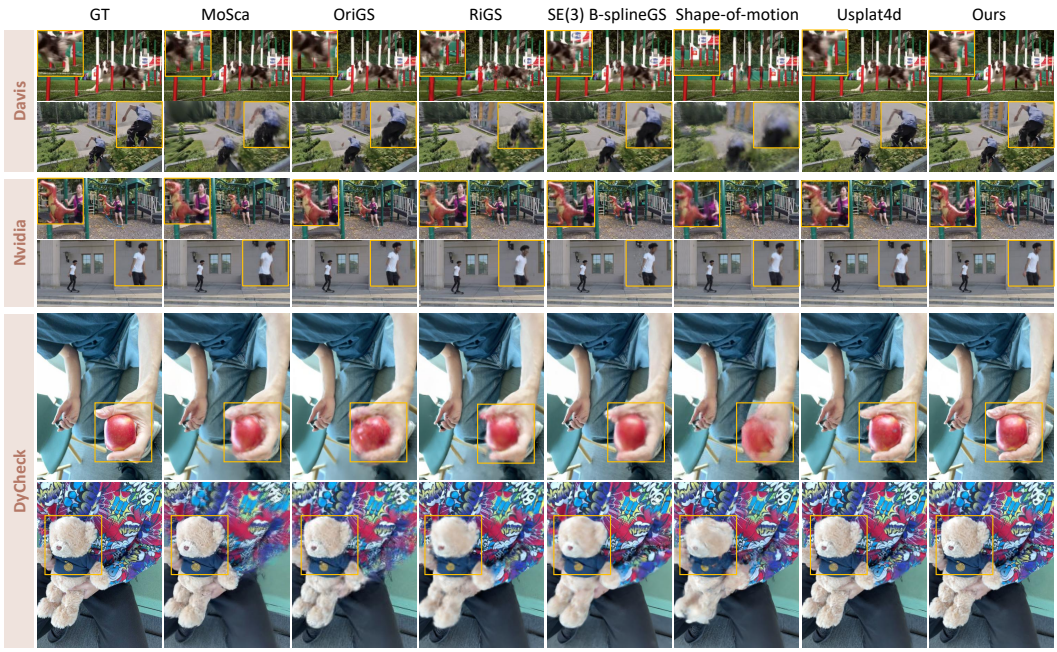


Figure 3: **Qualitative comparisons on DyCheck, NVIDIA, and DAVIS.** Ground-truth held-out views are shown for DyCheck and NVIDIA; DAVIS provides qualitative comparisons under rapid motion, occlusion, and non-rigid deformation.

highlight improvements around moving human silhouettes, object contours, and fine structures, where competing methods often exhibit motion artifacts or loss of detail. On DAVIS, RayCo4D further maintains coherent object shapes and clearer boundaries under rapid motion, partial occlusion, and non-rigid deformation. These results complement the quantitative evaluation and illustrate the method's performance on challenging real-world dynamic scenes.

Efficiency comparison. In terms of efficiency, RayCo4D requires 86 minutes on 200-frame DyCheck sequences, including 36 minutes for preprocessing and 50 minutes for training. Its overall runtime is comparable to the evaluated baselines. Appendix D.10 details the timing protocol and provides the complete per-method preprocessing and training breakdown in Tab. 7.

Failure cases. RayCo4D may struggle when motion changes abruptly during long observation gaps or when geometric estimates are unreliable. These challenges can lead to motion errors, geometric inconsistencies, and visible reconstruction artifacts, as shown in Appendix Sec. E.2.

4.3 Dissecting RayCo4D: Measurements, Dynamics, and Local Motion

We evaluate component contributions through quantitative ablations on DyCheck and NVIDIA. All ablations use the same input observations, initialization, training budget, and evaluation protocol as the full model, with only the indicated component modified.

What drives reconstruction quality? Tab. 1 evaluates four design choices. **w/o MSD** replaces the many-to-many measurement–state association with track-specific motion support; **w/o RCQ** optimizes reference queries freely in $\mathbb{R}^3$ instead of along their camera rays; **w/o WNOA factor** removes the process coupling between neighboring pose–velocity states; and **w/o ARM** disables the reference-anchored local residual. Removing MSD causes the largest mPSNR/PSNR drops: 1.35 dB on DyCheck and 1.33 dB on NVIDIA, with a 37.8% increase in NVIDIA LPIPS relative to the full model. The less distinct horse limb boundaries in Fig. 4 further illustrate the importance of shared motion support. Removing RCQ or the WNOA factor lowers mPSNR/PSNR by more than 1 dB on both datasets and increases NVIDIA LPIPS by 26.7% and 22.2%, respectively.

Their visual effects are complementary: removing RCQ blurs the dancer's extended limbs, whereas removing the WNOA factor blurs the umbrella's outline and colored panel boundaries. Together, these results support the roles of ray constraints and temporal coupling in recovering coherent shared geometry. Removing ARM has a smaller effect on whole-image fidelity, reducing mPSNR/PSNR by $0.18/0.17$ dB on DyCheck/NVIDIA, but still increases mLPIPS/LPIPS by $2.5\%/8.9\%$. The sharper limb contours in the jumping example with ARM illustrate its contribution to localized deformation beyond the shared motion scaffold.

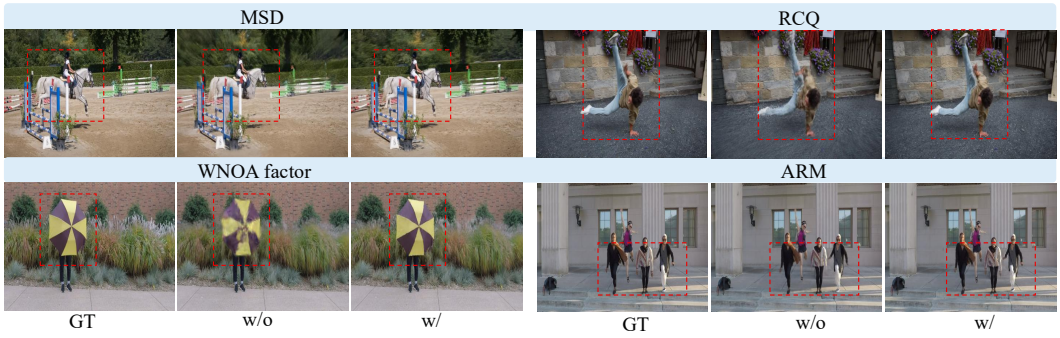


Figure 4: **Qualitative ablations on NVIDIA and DAVIS.** Comparisons with the full model illustrate the effects of individual components on structural coherence and fine-scale deformation.

Motion correspondence. Tab. 2 evaluates motion correspondence on DyCheck by projecting reconstructed 3D motion queries into target frames. RayCo4D achieves a PCK-T of 0.840, yielding relative improvements of 1.0% over SE(3) B-splineGS and 2.2% over MoSca. These gains suggest that the shared motion representation more accurately transports scene points across frames, complementing the rendering results with evidence of improved motion correspondence. This finding is consistent with the role of shared latent states in integrating track observations across different visibility intervals.

Table 2: **Motion correspondence on DyCheck (PCK-T↑).** Best result is highlighted in bold.

MoSca (Lei et al., 2025)	OriGS (Wu et al., 2025a)	SE(3) B-splineGS (Zhang et al., 2026)	**RayCo4D**
0.822	0.826	<u>0.832</u>	**0.840**

Intermediate-time reconstruction. Tab. 3 evaluates NVIDIA intermediate frames withheld from image, track, and depth supervision during training. Under the same temporal subsampling protocol, RayCo4D outperforms SE(3) B-splineGS by 1.11 dB in PSNR, increases SSIM by 1.9%, and reduces LPIPS by 25.9%. These gains indicate improved reconstruction at unobserved timestamps, particularly in perceptual quality.

Table 3: **Intermediate-time reconstruction on NVIDIA.** Best result is highlighted in bold.

Method	PSNR↑	SSIM↑	LPIPS↓
SE(3) B-splineGS (Zhang et al., 2026)	<u>25.76</u>	<u>0.835</u>	<u>0.081</u>
RayCo4D	**26.87**	**0.851**	**0.060**

5 Conclusion

We presented **RayCo4D**, a ray-conditioned continuous motion model for monocular 4D reconstruction. RayCo4D utilizes 2D tracks together with camera-depth geometry to constrain shared latent motion nodes through local many-to-many associations. Sparse $\mathrm{SE}(3)$ pose–velocity states, coupled by a white-noise-on-acceleration process prior, form continuous node trajectories that integrate evidence across different visibility intervals. The inferred motion is propagated to dynamic Gaussians through dual-quaternion blending, while reference-anchored local $\mathrm{SE}(3)$ corrections capture fine-grained non-rigid deformation. Experiments on DyCheck and NVIDIA scenes demonstrate competitive novel-view synthesis quality and improved novel-view synthesis, motion correspondence, and intermediate-time reconstruction. Our results validate that separating visual measurements from latent motion states provides a coherent and robust foundation for monocular 4D reconstruction.

AI USE STATEMENT

We used generative AI tools (GPT) for language polishing and grammar checking during manuscript preparation. Generative AI was not used to generate research ideas, conduct experiments, analyze results, or make scientific conclusions. All AI-assisted edits were reviewed and verified by the authors. We take full responsibility for the final content of this work, including all text, claims, results, and artifacts.

ETHICS STATEMENT

This work studies dynamic scene reconstruction from monocular videos and evaluates on existing research datasets. It does not involve human-subject studies or the collection of new personal or sensitive data. Videos depicting people may nevertheless raise privacy and consent concerns when used for reconstruction. Applications of this technology should respect dataset licenses, individual privacy, and consent, and avoid unauthorized surveillance or misleading visual content.

REPRODUCIBILITY STATEMENT

We describe the RayCo4D formulation, implementation details, optimization settings, datasets, and evaluation protocols in the main paper and appendix. We provide the core implementation as supplementary material, including ray-conditioned motion estimation, WNOA-based continuous-time modeling, and reference-anchored local refinement. The code package includes configuration files, dependency installation instructions, and commands for preprocessing, reconstruction, and rendering. Input datasets and third-party pretrained weights must be obtained separately, with weight download instructions provided in the accompanying README.

REFERENCES

Sean Anderson and Timothy D Barfoot. Full steam ahead: Exactly sparse gaussian process regression for batch continuous-time trajectory estimation on se (3). In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 157–164. IEEE, 2015.

Arnold Caleb Asiimwe and Carl Vondrick. EvoGS: 4D gaussian splatting as a learned dynamical system, 2025. URL https://arxiv.org/abs/2512.19648.

Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2 edition, 2024. doi: 10.1017/9781009299909.

Timothy D. Barfoot, Chi Hay Tong, and Simo Särkkä. Batch continuous-time trajectory estimation as exactly sparse gaussian process regression. In *Proceedings of Robotics: Science and Systems*, 2014. doi: 10.15607/RSS.2014.X.001.

Junli Deng, Ping Shi, Yihao Luo, and Qipei Li. Gaussians on their way: Wasserstein-constrained 4D gaussian splatting with state-space modeling. *Computer Graphics Forum*, 44(7):e70271, 2025. doi: 10.1111/cgf.70271.

Carl Doersch, Pauline Luc, Yi Yang, Dilara Gokay, Skanda Koppula, Ankush Gupta, Joseph Heyward, Ignacio Rocco, Ross Goroshin, João Carreira, and Andrew Zisserman. BootsTAP: Bootstrapped training for tracking-any-point. *arXiv preprint arXiv:2402.00847*, 2024.

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.

Fengzhi Guo, Chih-Chuan Hsu, Sihao Ding, and Cheng Zhang. Uncertainty matters in dynamic gaussian splatting for monocular 4D reconstruction. In *International Conference on Learning Representations*, 2026.

Yueru Jia, Jiaming Liu, Sixiang Chen, Chenyang Gu, Zhilve Wang, Longzan Luo, Xiaoqi Li, Pengwei Wang, Zhongyuan Wang, Renrui Zhang, et al. Lift3d policy: Lifting 2d foundation models for robust 3d robotic manipulation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17347–17358. IEEE, 2025.

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*, pp. 18–35. Springer, 2024.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, pp. 39–46, 2007.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, 2023. doi: 10.1145/3592433.

Mijeong Kim, Jungtaek Kim, and Bohyung Han. GP-4DGS: Probabilistic 4D gaussian splatting from monocular video via variational gaussian processes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 33394–33403, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

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*, pp. 6165–6177, 2025.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 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*, pp. 26866–26875, 2025.

Jongmin Park, Jeonghwan Yun, Minh-Quan Viet Bui, and Munchurl Kim. WebSpline: Structure-informed splines for real-time 3D gaussians from monocular videos, 2026. URL `https://arxiv.org/abs/2606.02096`.

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*, pp. 724–732, 2016.

Chan Hee Song, Valts Blukis, Jonathan Tremblay, Stephen Tyree, Yu Su, and Stan Birchfield. Robospatial: Teaching spatial understanding to 2d and 3d vision-language models for robotics. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15768–15780. IEEE, 2025.

Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, 2020.

Daniel Wang, Patrick Rim, Tian Tian, Dong Lao, Alex Wong, and Ganesh Sundaramoorthi. ODE-GS: Latent ODEs for dynamic scene extrapolation with 3D gaussian splatting. In *International Conference on Learning Representations*, 2026a.

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*, pp. 9660–9672, 2025.

Xuezhen Wang, Li Ma, Yulin Shen, Zeyu Wang, and Pedro V. Sander. RetimeGS: Continuous-time reconstruction of 4D gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7340–7350, 2026b.

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.

Chenyu Wu, Wanhua Li, Zhu-Tian Chen, and Hanspeter Pfister. RiGS: Rigid-aware 4D gaussian splatting from a single monocular video, 2026. URL https://arxiv.org/abs/2605.23672.

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*, pp. 20310–20320, 2024.

Junyi Wu, Jiachen Tao, Haoxuan Wang, Gaowen Liu, Ramana Rao Kompella, and Yan Yan. Orientation-anchored hyper-gaussian for 4D reconstruction from casual videos. In *Advances in Neural Information Processing Systems*, volume 38, pp. 159962–159985, 2025a. doi: 10.52202/085713-4826.

Rundi Wu, Ruiqi Gao, Ben Poole, Alex Trevithick, Changxi Zheng, Jonathan T Barron, and Aleksander Holynski. Cat4d: Create anything in 4d with multi-view video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26057–26068, 2025b.

Jinbo Yan, Rui Peng, Zhiyan Wang, Luyang Tang, Jiayu Yang, Jie Liang, Jiahao Wu, and Ronggang Wang. Instant gaussian stream: Fast and generalizable streaming of dynamic scene reconstruction via gaussian splatting. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16520–16531. IEEE, 2025.

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*, pp. 20331–20341, 2024.

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*, pp. 5336–5345, 2020.

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*, 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*, pp. 33291–33300, 2026.

SUPPLEMENTARY MATERIAL

This supplementary material provides additional details and results for RayCo4D. Sections A and B present additional related work and notation. Section C derives the WNOA formulation and its continuous-time interpolation. Section D describes hierarchical Gaussian reconstruction, training, evaluation protocols, and runtime. Sections E and F present additional experimental results, challenging cases, and scene-wise qualitative comparisons.

A ADDITIONAL RELATED WORK

Spatial structure and motion sharing. Recent structured representations address different aspects of dynamic scene complexity. OriGS (Wu et al., 2025a) uses a Global Orientation Field to organize region-specific deformations. RiGS (Wu et al., 2026) separates static, rigid, and transient Gaussians to accommodate motion at different temporal scales. USPLAT4D (Guo et al., 2026) constructs a spatio-temporal motion graph using per-Gaussian uncertainty, propagating reliable motion cues toward weakly observed regions. RayCo4D organizes spatial sharing through local track–node associations: reference queries retain their geometric parameters, while shared node states encode time-varying motion.

Spline trajectories and local refinement. Beyond SplineGS (Park et al., 2025) and SE(3) B-splineGS (Zhang et al., 2026), WebSpline (Park et al., 2026) initializes a Structural Proxy Graph from 2D tracks to regularize per-Gaussian Hermite trajectories. RetimeGS (Wang et al., 2026b) defines continuous-time Gaussian behavior for temporal retiming and intermediate-time rendering. RayCo4D also uses cubic Hermite expressions, at two distinct levels: the WNOA conditional mean interpolates shared pose–velocity states in a local $\mathrm{SE}(3)$ chart, and per-Gaussian Hermite curves parameterize reference-anchored local corrections. The shared process factors and local residual constraints determine how these temporal representations are estimated and combined.

Probabilistic and dynamical formulations. GP-4DGS (Kim et al., 2026) models spatio-temporal deformation correlations through variational Gaussian processes, supporting uncertainty estimation and temporal extrapolation. Gaussians on their Way (Deng et al., 2025) treats Gaussian means and covariances as dynamic states and fuses motion predictions with deformation observations. EvoGS (Asiimwe & Vondrick, 2025) integrates a learned velocity field, while ODE-GS (Wang et al., 2026a) evolves latent states derived from reconstructed Gaussian trajectories through a Neural ODE. RayCo4D places its WNOA prior on sparse shared $\mathrm{SE}(3)$ pose–velocity states. Ray-conditioned measurements constrain these states through local associations, and their continuous transformations drive the dense Gaussian representation.

B NOTATION

Tab. 4 summarizes the main symbols. Time $t \in [0, F-1]$ is measured in frame units, with noninteger values denoting subframe times. All rigid transformations act on column vectors: $\mathbf{T} \cdot \mathbf{x} = \mathbf{R}\mathbf{x} + \mathbf{p}$ for $\mathbf{T} = [\mathbf{R}, \mathbf{p}] \in \mathrm{SE}(3)$, and transformation products act from right to left. Twists are ordered as translation followed by rotation, with body velocity defined by $\dot{\mathbf{T}}(t) = \mathbf{T}(t)\boldsymbol{\nu}(t)^{\wedge}$. The operators $(\cdot)^{\wedge}$ and $(\cdot)^{\vee}$ convert between six-dimensional vectors and $\mathfrak{se}(3)$ matrices; Exp and Log denote the group exponential and local logarithm.

For camera-to-world pose $\mathbf{C}_f = [\mathbf{R}_f^c, \mathbf{c}_f]$ and intrinsics $\mathbf{K}_f$, the image projection and camera-axis depth of a world point $\mathbf{x}$ are

$$\begin{aligned} \pi_f(\mathbf{x}) &= \pi\big(\mathbf{K}_f(\mathbf{R}_f^c)^{\top}(\mathbf{x} - \mathbf{c}_f)\big)\,, \\ z_f(\mathbf{x}) &= \mathbf{e}_3^{\top}(\mathbf{R}_f^c)^{\top}(\mathbf{x} - \mathbf{c}_f), \end{aligned} \tag{12}$$

where $\pi([x, y, z]^{\top}) = [x/z, y/z]^{\top}$ and $\mathbf{e}_3 = [0, 0, 1]^{\top}$. Depth denotes the camera-axis coordinate rather than Euclidean distance to the camera center.

Table 4: **Main notation used in RayCo4D.** Symbols are grouped by their roles in the representation.

Symbol	Meaning
Indices and time	
f, i, j, g, k	Frame, track, motion-node, Gaussian, and support-time indices.
F, M, S	Numbers of video frames, motion nodes, and temporal supports.
t, τ_k, Δ_k	Query time, support time, and interval $\Delta_k = \tau_{k+1} - \tau_k$.
r_i, r_g	Reference times of track i and Gaussian g.
$\bar{t}$	Normalized time $t/\max(F-1, 1)$ for local motion curves.
Measurements and associations	
$\mathbf{C}_f, \mathbf{K}_f$	Camera-to-world pose and camera intrinsics.
$\mathbf{u}_{if}, c_{if}$	Observed track pixel and its confidence.
$\widetilde{\mathbf{u}}_{if}$	Homogeneous pixel coordinate $[\mathbf{u}_{if}^\top, 1]^\top$.
$\mathcal{O}_u, \mathcal{O}_d$	Visible observations and the subset with valid depth.
$\mathbf{q}_i, \widehat{\mathbf{x}}_i(t)$	Reference query in world space and its predicted position at t.
d_i^0, z_i, η_i	Initial query depth, optimized camera-axis depth, and scalar depth parameter.
$\mathcal{N}_i, a_{ij}$	Track-to-node neighborhood and normalized association weight.
K, σ_j	Number of neighboring nodes and node influence radius.
Continuous shared motion	
$\mathbf{T}_j(t)$	Continuous rigid pose of motion node j.
$\mathbf{T}_{jk}, \boldsymbol{\nu}_{jk}$	Node pose and body-frame velocity at τ_k.
$\boldsymbol{\xi}_{jk}(t)$	Local pose coordinate $\mathrm{Log}(\mathbf{T}_{jk}^{-1}\mathbf{T}_j(t))^\vee$.
$\mathbf{J}_r$	Right Jacobian of $\mathrm{SE}(3)$.
$\mathbf{e}_{jk}$	Neighboring-state WNOA residual in $\mathbb{R}^{12}$.
$\mathbf{Q}_c, \mathbf{Q}_k$	Acceleration spectral-density matrix and integrated process covariance over Δ_k.
ℓ	Characteristic spatial scale of the motion scaffold.
Hierarchical Gaussian motion	
$\mathbf{P}_g^{\mathrm{ref}}, \mathbf{P}_g(t)$	Reference Gaussian pose and composed pose at time t.
$\mathcal{N}_g, \overline{a}_{gj}$	Gaussian-to-node neighborhood and normalized binding weight.
$a_{gj}^{\mathrm{geo}}, b_{gj}$	Geometric binding weight and learnable logit offset.
DQB	Dual-quaternion blending of rigid transformations.
$\mathbf{A}_g(t), \mathbf{E}_g(t)$	Shared motion transform and reference-anchored local residual.
$\mathbf{h}_g, \boldsymbol{\delta}_g(t)$	Local motion curve and anchored increment $\mathbf{h}_g(\bar{t}) - \mathbf{h}_g(\bar{r}_g)$.
$\mathbf{D}_\ell$	Translation-scaling matrix $\mathrm{diag}(\ell\mathbf{I}_3, \mathbf{I}_3)$.

C Full WNOA Formulation

This section provides the complete formulation of the continuous-time white-noise-on-acceleration process used in RayCo4D. Twists are ordered as translation followed by rotation, body velocities satisfy $\dot{\mathbf{T}}(t) = \mathbf{T}(t)\boldsymbol{\nu}(t)^\wedge$, and time is measured in video-frame units.

C.1 Local Coordinates on $\mathrm{SE}(3)$

Consider node j over the support interval $[\tau_k, \tau_{k+1}]$. We define a local tangent-space trajectory centered at the left endpoint:

$$\boldsymbol{\xi}_{jk}(t) = \mathrm{Log}\Big(\mathbf{T}_{jk}^{-1}\mathbf{T}_j(t)\Big)^\vee. \tag{13}$$

The derivative of this local coordinate is related to the body-frame velocity through the right Jacobian:

$$\dot{\boldsymbol{\xi}}_{jk}(t) = \mathbf{J}_r(\boldsymbol{\xi}_{jk}(t))^{-1}\,\boldsymbol{\nu}_j(t). \tag{14}$$

At the left endpoint, $\boldsymbol{\xi}_{jk}(\tau_k) = \mathbf{0}$, and hence

$$\dot{\boldsymbol{\xi}}_{jk}(\tau_k) = \boldsymbol{\nu}_{jk}. \tag{15}$$

At the right endpoint, define

$$\begin{aligned}\boldsymbol{\xi}_{jk}^{+} &= \mathrm{Log}\Big(\mathbf{T}_{jk}^{-1}\mathbf{T}_{j,k+1}\Big)^{\vee}, \\ \widetilde{\boldsymbol{\nu}}_{j,k+1} &= \mathbf{J}_r(\boldsymbol{\xi}_{jk}^{+})^{-1}\boldsymbol{\nu}_{j,k+1}.\end{aligned} \tag{16}$$

The transformed velocity $\widetilde{\boldsymbol{\nu}}_{j,k+1}$ is the derivative of the local coordinate at the right endpoint. It allows both endpoint velocities to be compared in the same tangent space.

C.2 Linear WNOA Process in the Local Chart

We concatenate local pose and velocity into the state

$$\boldsymbol{\gamma}_{jk}(t) = \begin{bmatrix}\boldsymbol{\xi}_{jk}(t) \\ \dot{\boldsymbol{\xi}}_{jk}(t)\end{bmatrix} \in \mathbb{R}^{12}. \tag{17}$$

The WNOA process assumes

$$\ddot{\boldsymbol{\xi}}_{jk}(t) = \mathbf{w}(t), \qquad \mathbb{E}[\mathbf{w}(t)\mathbf{w}(t')^{\top}] = \mathbf{Q}_c\delta(t-t'), \tag{18}$$

which gives the linear time-invariant stochastic differential equation

$$\dot{\boldsymbol{\gamma}}_{jk}(t) = \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{I}_6 \\ \mathbf{0} & \mathbf{0}\end{bmatrix}}_{\mathbf{A}}\boldsymbol{\gamma}_{jk}(t) + \underbrace{\begin{bmatrix}\mathbf{0} \\ \mathbf{I}_6\end{bmatrix}}_{\mathbf{L}}\mathbf{w}(t). \tag{19}$$

For a time interval of length δ, the state-transition matrix is

$$\boldsymbol{\Phi}(\delta) = \exp(\mathbf{A}\delta) = \begin{bmatrix}\mathbf{I}_6 & \delta\mathbf{I}_6 \\ \mathbf{0} & \mathbf{I}_6\end{bmatrix}. \tag{20}$$

Integrating the white acceleration noise gives the process covariance

$$\begin{aligned}\mathbf{Q}(\delta) &= \int_0^{\delta}\boldsymbol{\Phi}(\delta-s)\mathbf{L}\mathbf{Q}_c\mathbf{L}^{\top}\boldsymbol{\Phi}(\delta-s)^{\top}\,\mathrm{d}s \\ &= \begin{bmatrix}\frac{\delta^3}{3}\mathbf{Q}_c & \frac{\delta^2}{2}\mathbf{Q}_c \\ \frac{\delta^2}{2}\mathbf{Q}_c & \delta\mathbf{Q}_c\end{bmatrix}.\end{aligned} \tag{21}$$

We use separate acceleration spectral densities for translation and rotation:

$$\mathbf{Q}_c = \mathrm{diag}\left(q_p\ell^2\mathbf{I}_3, q_R\mathbf{I}_3\right), \tag{22}$$

where ℓ is the characteristic spatial scale of the motion scaffold. Equivalently, translational pose and velocity residuals can be normalized by ℓ before applying the density q_p.

The endpoint states in the local chart are

$$\boldsymbol{\gamma}_{jk}^{-} = \begin{bmatrix}\mathbf{0} \\ \boldsymbol{\nu}_{jk}\end{bmatrix}, \qquad \boldsymbol{\gamma}_{jk}^{+} = \begin{bmatrix}\boldsymbol{\xi}_{jk}^{+} \\ \widetilde{\boldsymbol{\nu}}_{j,k+1}\end{bmatrix}. \tag{23}$$

For $\Delta_k = \tau_{k+1} - \tau_k$, the transition residual becomes

$$\begin{aligned}\mathbf{e}_{jk} &= \boldsymbol{\gamma}_{jk}^{+} - \boldsymbol{\Phi}(\Delta_k)\boldsymbol{\gamma}_{jk}^{-} \\ &= \begin{bmatrix}\boldsymbol{\xi}_{jk}^{+} - \Delta_k\boldsymbol{\nu}_{jk} \\ \widetilde{\boldsymbol{\nu}}_{j,k+1} - \boldsymbol{\nu}_{jk}\end{bmatrix}.\end{aligned} \tag{24}$$

The corresponding process objective is

$$\mathcal{L}_{\mathrm{WNOA}} = \frac{1}{3M(S-1)}\sum_{j=1}^{M}\sum_{k=0}^{S-2}\mathbf{e}_{jk}^{\top}\mathbf{Q}(\Delta_k)^{-1}\mathbf{e}_{jk}. \tag{25}$$

Equivalent implementation form. Let

$$\mathbf{e}_{jk}^{p} = \boldsymbol{\xi}_{jk}^{+} - \Delta_k \boldsymbol{\nu}_{jk}, \qquad \mathbf{e}_{jk}^{v} = \widetilde{\boldsymbol{\nu}}_{j,k+1} - \boldsymbol{\nu}_{jk}. \tag{26}$$

The full covariance-weighted quadratic can be evaluated without explicitly forming $\mathbf{Q}(\Delta_k)^{-1}$. For translation, the equivalent energy is

$$\mathcal{E}_{jk}^{\text{trans}} = \frac{12}{q_p \Delta_k^3} \left\| \frac{\mathbf{e}_{jk,1:3}^{p} - \frac{\Delta_k}{2}\mathbf{e}_{jk,1:3}^{v}}{\ell} \right\|_2^2 + \frac{1}{q_p \Delta_k} \left\| \frac{\mathbf{e}_{jk,1:3}^{v}}{\ell} \right\|_2^2, \tag{27}$$

and the rotational energy is

$$\mathcal{E}_{jk}^{\text{rot}} = \frac{12}{q_R \Delta_k^3} \left\| \mathbf{e}_{jk,4:6}^{p} - \frac{\Delta_k}{2}\mathbf{e}_{jk,4:6}^{v} \right\|_2^2 + \frac{1}{q_R \Delta_k} \left\| \mathbf{e}_{jk,4:6}^{v} \right\|_2^2. \tag{28}$$

Equations (27) and (28) are algebraically equivalent to the quadratic form in Eq. (25).

C.3 Continuous-Time Conditional Mean

For a query time $t \in [\tau_k, \tau_{k+1}]$, let

$$\delta = t - \tau_k, \qquad \Delta_k = \tau_{k+1} - \tau_k, \qquad s = \frac{\delta}{\Delta_k}. \tag{29}$$

The conditional mean of the linear WNOA process can be written as

$$\begin{aligned} \overline{\boldsymbol{\gamma}}_{jk}(t) &= \boldsymbol{\Phi}(\delta)\boldsymbol{\gamma}_{jk}^{-} \\ &\quad + \mathbf{Q}(\delta)\boldsymbol{\Phi}(\Delta_k - \delta)^{\top}\mathbf{Q}(\Delta_k)^{-1}\left(\boldsymbol{\gamma}_{jk}^{+} - \boldsymbol{\Phi}(\Delta_k)\boldsymbol{\gamma}_{jk}^{-}\right). \end{aligned} \tag{30}$$

For WNOA, the pose component of Eq. (30) reduces to cubic Hermite interpolation. The basis functions are

$$\begin{aligned} h_{00}(s) &= 2s^3 - 3s^2 + 1, \quad h_{10}(s) = s^3 - 2s^2 + s, \\ h_{01}(s) &= -2s^3 + 3s^2, \qquad h_{11}(s) = s^3 - s^2. \end{aligned} \tag{31}$$

Because the local pose at the left endpoint is zero, the h_{00} term vanishes, giving

$$\overline{\boldsymbol{\xi}}_{jk}(t) = h_{10}(s)\Delta_k \boldsymbol{\nu}_{jk} + h_{01}(s)\boldsymbol{\xi}_{jk}^{+} + h_{11}(s)\Delta_k \widetilde{\boldsymbol{\nu}}_{j,k+1}. \tag{32}$$

The continuous pose is then recovered on the group:

$$\mathbf{T}_j(t) = \mathbf{T}_{jk} \operatorname{Exp}\left(\overline{\boldsymbol{\xi}}_{jk}(t)^{\wedge}\right). \tag{33}$$

For completeness, differentiating the Hermite curve gives

$$\begin{aligned} \dot{\overline{\boldsymbol{\xi}}}_{jk}(t) = \frac{1}{\Delta_k}\Big[& h'_{10}(s)\Delta_k \boldsymbol{\nu}_{jk} + h'_{01}(s)\boldsymbol{\xi}_{jk}^{+} \\ & + h'_{11}(s)\Delta_k \widetilde{\boldsymbol{\nu}}_{j,k+1}\Big], \end{aligned} \tag{34}$$

and the corresponding body velocity is

$$\boldsymbol{\nu}_j(t) = \mathbf{J}_r\left(\overline{\boldsymbol{\xi}}_{jk}(t)\right)\dot{\overline{\boldsymbol{\xi}}}_{jk}(t). \tag{35}$$

This construction satisfies

$$\begin{aligned} \mathbf{T}_j(\tau_k) &= \mathbf{T}_{jk}, \quad \mathbf{T}_j(\tau_{k+1}) = \mathbf{T}_{j,k+1}, \\ \boldsymbol{\nu}_j(\tau_k) &= \boldsymbol{\nu}_{jk}, \quad \boldsymbol{\nu}_j(\tau_{k+1}) = \boldsymbol{\nu}_{j,k+1}. \end{aligned} \tag{36}$$

Thus, the process factor and the continuous pose query are derived from the same pose–velocity state model.

D Additional Details of Hierarchical Gaussian Reconstruction

D.1 Gaussian-to-Node Binding

For Gaussian g, let $\boldsymbol{\mu}_g^{\mathrm{ref}}$ be its reference center and $\mathcal{N}_g$ its K nearest motion nodes at reference time r_g. The geometric binding weights are

$$a_{gj}^{\mathrm{geo}} = \frac{\exp\big(-\|\boldsymbol{\mu}_g^{\mathrm{ref}} - \mathbf{p}_j(r_g)\|_2^2/(2\sigma_j^2)\big)}{\sum_{\ell\in\mathcal{N}_g}\exp\big(-\|\boldsymbol{\mu}_g^{\mathrm{ref}} - \mathbf{p}_\ell(r_g)\|_2^2/(2\sigma_\ell^2)\big)}. \tag{37}$$

Photometric fitting introduces the Gaussian-specific offsets b_{gj}:

$$\overline{a}_{gj} = \frac{a_{gj}^{\mathrm{geo}}\exp(b_{gj})}{\sum_{\ell\in\mathcal{N}_g} a_{g\ell}^{\mathrm{geo}}\exp(b_{g\ell})}. \tag{38}$$

The offsets are initialized to zero, so $\overline{a}_{gj} = a_{gj}^{\mathrm{geo}}$ at the beginning of optimization.

For each node, define the relative transformation from the Gaussian reference time to query time t:

$$\Delta\mathbf{T}_{gj}(t) = \mathbf{T}_j(t)\mathbf{T}_j(r_g)^{-1}. \tag{39}$$

The shared Gaussian transform is obtained by dual-quaternion blending (Kavan et al., 2007):

$$\mathbf{A}_g(t) = \mathrm{DQB}\Big(\{\overline{a}_{gj}, \Delta\mathbf{T}_{gj}(t)\}_{j\in\mathcal{N}_g}\Big). \tag{40}$$

Because $\Delta\mathbf{T}_{gj}(r_g) = \mathbf{I}$ for every j, the blended transformation also satisfies $\mathbf{A}_g(r_g) = \mathbf{I}$.

D.2 Local Hermite Motion Parameterization

Each Gaussian stores $L-1$ learnable six-dimensional controls. We prepend a fixed zero control:

$$\mathbf{c}_{g,0} = \mathbf{0}, \qquad \mathbf{c}_{g,1}, \ldots, \mathbf{c}_{g,L-1} \in \mathbb{R}^6. \tag{41}$$

For normalized time $u \in [0,1]$, define

$$k = \min(\lfloor (L-1)u \rfloor, L-2), \qquad s = (L-1)u - k. \tag{42}$$

Let

$$k^- = \max(k-1, 0), \qquad k^+ = \min(k+2, L-1).$$

The endpoint tangents for segment k are computed from adjacent control differences:

$$\begin{aligned} \mathbf{m}_{g,k} &= \frac{\mathbf{c}_{g,k+1} - \mathbf{c}_{g,k^-}}{k+1-k^-}, \\ \mathbf{m}_{g,k+1} &= \frac{\mathbf{c}_{g,k^+} - \mathbf{c}_{g,k}}{k^+ - k}. \end{aligned} \tag{43}$$

Using the cubic Hermite bases

$$\begin{aligned} h_{00}(s) &= 2s^3 - 3s^2 + 1, \quad h_{10}(s) = s^3 - 2s^2 + s, \\ h_{01}(s) &= -2s^3 + 3s^2, \qquad h_{11}(s) = s^3 - s^2, \end{aligned} \tag{44}$$

the local curve is

$$\begin{aligned} \mathbf{h}_g(u) = {} & h_{00}(s)\mathbf{c}_{g,k} + h_{10}(s)\mathbf{m}_{g,k} \\ & + h_{01}(s)\mathbf{c}_{g,k+1} + h_{11}(s)\mathbf{m}_{g,k+1}. \end{aligned} \tag{45}$$

For video time t, RayCo4D evaluates the difference

$$\boldsymbol{\delta}_g(t) = \mathbf{h}_g\left(\frac{t}{\max(F-1,1)}\right) - \mathbf{h}_g\left(\frac{r_g}{\max(F-1,1)}\right). \tag{46}$$

The subtraction makes the residual exactly zero at the Gaussian's own reference time for every possible set of control values. After scaling the translational part by the scaffold spatial unit ℓ, the local transform is

$$\mathbf{E}_g(t) = \mathrm{Exp}\big([\mathrm{diag}(\ell\mathbf{I}_3, \mathbf{I}_3)\boldsymbol{\delta}_g(t)]^\wedge\big). \tag{47}$$

D.3 Local Motion Regularization

The weighted local-motion regularizer is

$$\mathcal{L}_{\text{local}} = \lambda_m \mathcal{L}_{\text{mag}} + \lambda_c \mathcal{L}_{\text{curv}} + \lambda_n \mathcal{L}_{\text{nbr}}. \tag{48}$$

Control magnitude. The magnitude term limits the overall capacity of the local correction:

$$\mathcal{L}_{\text{mag}} = \frac{1}{6G(L-1)} \sum_{g=1}^{G} \sum_{k=1}^{L-1} \|\mathbf{c}_{g,k}\|_2^2. \tag{49}$$

Control curvature. Temporal smoothness is encouraged through second differences of the control sequence:

$$\mathcal{L}_{\text{curv}} = \frac{1}{6G(L-2)} \sum_{g=1}^{G} \sum_{k=1}^{L-2} \|\mathbf{c}_{g,k+1} - 2\mathbf{c}_{g,k} + \mathbf{c}_{g,k-1}\|_2^2. \tag{50}$$

For $L = 2$, this term is set to zero. It acts as a smoothness prior on the control sequence.

Neighborhood consistency. Local-motion neighborhoods are constructed in a common scene configuration. We first transport every Gaussian to

$$t_\star = \lfloor F/2 \rfloor$$

using the shared motion only:

$$\mathbf{P}_g^0(t) = \mathbf{A}_g(t)\mathbf{P}_g^{\text{ref}}. \tag{51}$$

For each Gaussian, we retain up to five nearest neighbors in this common configuration. The edge weight between Gaussians g and h is

$$\beta_{gh} = \exp\left(-\frac{\|\boldsymbol{\mu}_g^0(t_\star) - \boldsymbol{\mu}_h^0(t_\star)\|_2^2}{2\ell^2}\right) \frac{|\mathcal{N}_g \cap \mathcal{N}_h|}{K}. \tag{52}$$

The first factor favors spatially close Gaussians, and the second factor measures the overlap of their shared-node support.

For an adjacent-frame interval $[a, b]$, define the local control increment

$$\Delta\mathbf{h}_g^{a,b} = \mathbf{h}_g\left(\frac{b}{\max(F-1,1)}\right) - \mathbf{h}_g\left(\frac{a}{\max(F-1,1)}\right). \tag{53}$$

Let $\mathbf{R}_g^0(t)$ be the Gaussian orientation produced by shared motion at the sampled scene time t. We express the increment in a common world orientation as

$$\mathbf{d}_g(t) = \operatorname{diag}\left(\mathbf{R}_g^0(t), \mathbf{R}_g^0(t)\right) \Delta\mathbf{h}_g^{a,b}. \tag{54}$$

The neighborhood term is

$$\mathcal{L}_{\text{nbr}} = \frac{\sum_{(g,h)} \beta_{gh} \|\mathbf{d}_g(t) - \mathbf{d}_h(t)\|_2^2}{6 \sum_{(g,h)} \beta_{gh}}. \tag{55}$$

In implementation, this objective is estimated from sampled center Gaussians and their cached neighbors.

D.4 Dynamic Gaussian Parameters

Write the composed Gaussian pose as

$$\mathbf{P}_g(t) = [\mathbf{R}_g(t), \boldsymbol{\mu}_g(t)].$$

For a learned time-independent scale vector $\mathbf{s}_g \in \mathbb{R}^3_{>0}$, the world-space covariance is

$$\boldsymbol{\Sigma}_g(t) = \mathbf{R}_g(t) \operatorname{diag}(\mathbf{s}_g^2) \mathbf{R}_g(t)^\top. \tag{56}$$

The center and covariance vary through the hierarchical motion, while scale, opacity, and spherical-harmonic coefficients are learned as Gaussian attributes shared across scene times. The dynamic and static Gaussian sets are concatenated before differentiable rasterization.

D.5 Training Objective and Auxiliary Terms

The main objective is reproduced here for completeness:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{\text{scene}} + \lambda_u \mathcal{L}_{\text{2D}} + \lambda_d \mathcal{L}_{\text{depth}} + \lambda_W \mathcal{L}_{\text{WNOA}} + \lambda_A \mathcal{L}_{\text{ARAP}} \\ + \mathcal{L}_{\text{local}} + \mathcal{L}_{\text{cal}} + \lambda_{\text{warp}} \mathcal{L}_{\text{warp}}.\end{aligned} \tag{57}$$

The scene objective combines RGB reconstruction, rendered depth, rendered track correspondence, coverage, ownership, and binding-offset regularization. RGB supervision uses pixel-averaged ℓ_1 error plus $0.1(1-\text{SSIM})$. The RGB, rendered-depth, correspondence, and binding-offset terms have weights 1, 0.01, 0.005, and 0.01, respectively; track coverage, ownership coverage, and ownership classification each use 0.1. Ownership classification is linearly annealed to zero during joint fitting.

Adjacent-frame appearance transport. For adjacent frames f and f', the predicted Gaussian poses define a local image transport $\mathcal{W}_{f\to f'}$. The target image is sampled at the transported coordinates, yielding the appearance consistency term

$$\mathcal{L}_{\text{warp}} = \text{LBMean}\left(\left\|I_{f'}\left(\mathcal{W}_{f\to f'}(\mathbf{p})\right) - I_f(\mathbf{p})\right\|_1, \omega(\mathbf{p})\right), \tag{58}$$

where LBMean denotes layer-balanced averaging and ω combines static/dynamic ownership with externally estimated correspondence-validity support. Predicted alpha and depth determine image coverage and occlusion at the transported location. A companion coverage term encourages sufficient Gaussian support at valid transported pixels.

Camera calibration. When camera refinement is enabled, the camera-to-world pose is written as

$$\mathbf{C}_f = \text{Exp}(\boldsymbol{\epsilon}_f^{\wedge})\mathbf{C}_f^0, \qquad \boldsymbol{\epsilon}_0 = \mathbf{0}, \tag{59}$$

where $\mathbf{C}_f^0$ is the input pose. Intrinsics remain fixed. The camera regularizers are

$$\begin{aligned}\mathcal{L}_{\text{cam}} &= \frac{\sum_{f=1}^{F-1} \|\boldsymbol{\epsilon}_f\|_2^2}{6(F-1)}, \\ \mathcal{L}_{\text{cam-smooth}} &= \frac{\sum_{f=0}^{F-2} \|\boldsymbol{\epsilon}_{f+1} - \boldsymbol{\epsilon}_f\|_2^2}{6(F-1)}.\end{aligned} \tag{60}$$

Depth calibration. We optimize a scalar log-depth correction s_f per frame, centered across the sequence:

$$\begin{aligned}\kappa_f &= s_f - \frac{1}{F}\sum_{h=0}^{F-1} s_h, \\ D_f^{\text{corr}}(\mathbf{u}) &= D_f(\mathbf{u})\exp(\kappa_f).\end{aligned} \tag{61}$$

The optional spatial correction grid is fixed to zero. The calibration objective combines penalties on these corrections with the camera regularizers and static-track constraints:

$$\mathcal{L}_{\text{cal}} = \lambda_{\text{cam}}\mathcal{L}_{\text{cam}} + \lambda_{\text{cam-smooth}}\mathcal{L}_{\text{cam-smooth}} + \lambda_{\text{dep-cal}}\mathcal{L}_{\text{dep-cal}} + \lambda_{\text{static}}\mathcal{L}_{\text{static}}. \tag{62}$$

Here, $\mathcal{L}_{\text{dep-cal}}$ penalizes squared log-depth corrections at valid depth observations, weighted by depth reliability; $\mathcal{L}_{\text{static}}$ is the confidence-weighted smooth-ℓ_1 reprojection error of fixed static landmarks and updates only the camera corrections.

Loss weights. We set $(\lambda_u, \lambda_d, \lambda_W, \lambda_A) = (1, 1, 0.1, 0.1)$ and $(\lambda_m, \lambda_c, \lambda_n) = (0.001, 0.01, 0.01)$. The camera prior, camera smoothness, depth-calibration prior, and static landmark weights are 0.001, 0.0001, 0.1, and 0.01, respectively. The depth-calibration prior is evaluated on both sampled ray observations and the rendered source frame. Appearance transport and its companion coverage term use weights 0.05 and 0.1.

D.6 Optimization Schedule

Tab. 5 summarizes the four stages. After motion warm-up, step indices below refer to the separate 8,000-step scene-fitting stage, starting at zero. Gaussian binding offsets, camera–depth calibration, local Hermite controls, and appearance transport are enabled at steps 1,000, 2,000, 2,500, and 3,000, respectively. Gaussian parameters are updated every step; node states, query depths, and active calibration variables are updated every five steps after Gaussian warm-up, including the final stage. Gaussian structure control is active at steps 1,000–5,999; node adaptation is restricted to joint fitting. Ownership is initialized before Gaussian fitting, corrected once at step 3,000, and then fixed.

Table 5: Training stages. Iterations are counted separately for each row.

Stage	Iterations	Optimized variables
Motion warm-up	4,000	Node states and query depths
Gaussian warm-up	1,000	Gaussian geometry and appearance
Joint fitting	6,000	Scene, motion, and calibration
Final fitting	1,000	Continuous variables; fixed structure

D.7 Additional Implementation Details

Initialization and representation. Lifted tracks initialize motion nodes through spatial coverage sampling; the node count depends on the scene. Missing positions are interpolated linearly between observations and extended from the nearest endpoint outside the observed span. These filled positions are used only for initialization and are not marked as measured support. Pose–velocity supports are placed every two frames, including both endpoints; initial rotations are identities and body velocities are estimated from neighboring support poses. We use $K = 16$ candidate nodes, scaffold scale $\ell = 0.02$ in normalized scene units, $q_p = q_R = 1$, and $L = 8$ local Hermite controls. Initialization uses at most 80,000 static depth samples, 80,000 dynamic track seeds, and 40,000 additional dynamic depth samples.

Sampling and updates. Each motion update samples up to 16,384 visible observations with replacement. Motion regularization uses a contiguous window of up to 16 support states and up to 1,024 sampled nodes; this is a minibatch size, not the total node count. Track candidates are refreshed every 100 motion-warm-up steps. During scene fitting, binding refreshes occur at initialization, step 3,000, and as required by node-structure changes. Layer supervision is evaluated every two steps and appearance transport every four steps, with their losses scaled by the respective intervals.

Optimization. We use Adam with the learning rates in Tab. 6. During motion warm-up, node and query-depth rates follow a cosine decay to 5% of their initial values; scene fitting starts these groups at 10% of their initial warm-up rates. From scene-fitting step 2,000, Gaussian position rates decay exponentially to 1.6×10^{-6}, and camera and depth-scale rates to one tenth of their initial values. The other rates remain constant.

D.8 Baseline Details

We compare against methods covering the main motion representations used in recent monocular dynamic reconstruction. MoSca (Lei et al., 2025) constructs a sparse motion scaffold from lifted tracks and propagates node transformations to dynamic Gaussians. Shape of Motion (Wang et al., 2025) represents scene dynamics using shared $\mathrm{SE}(3)$ motion bases constrained by long-range tracks and monocular geometry. SE(3) B-splineGS (Zhang et al., 2026) parameterizes continuous motion using $\mathrm{SE}(3)$ B-spline control trajectories, providing the most direct comparison to our WNOA pose–velocity formulation. OriGS (Wu et al., 2025a) and RiGS (Wu et al., 2026) are included for their treatment of complex local deformation and multi-timescale motion, respectively. USPLAT4D (Guo et al., 2026) evaluates reconstruction under incomplete and unreliable observations, including occlusion.

Table 6: Initial learning rates. Node and query-depth entries refer to motion warm-up; their scene-fitting rates are scaled by 0.1.

Parameter group	Learning rate
Reference-query depths	10^{-2}
Node translation, rotation, and body velocity	3×10^{-3}
Gaussian position	1.6×10^{-4}
Gaussian rotation / scale / opacity	10^{-3} / 5×10^{-3} / 5×10^{-2}
SH coefficients: constant / higher-order	2.5×10^{-3} / 1.25×10^{-4}
Gaussian binding offsets	10^{-2}
Local Hermite controls	10^{-3}
Camera translation and rotation corrections	3×10^{-5}
Per-frame log-depth scales	10^{-3}

D.9 Detail Datasets and Evaluation Protocols

DyCheck Dataset. The DyCheck Dataset (Gao et al., 2022) contains casually captured monocular videos of dynamic scenes together with spatially separated evaluation views. It is particularly challenging because the training and evaluation cameras often observe only partially overlapping scene regions. We follow the standard seven-scene protocol at $2\times$ resolution. Novel-view synthesis metrics are computed within the provided co-visibility masks, yielding mPSNR, mSSIM, and mLPIPS.

We evaluate motion correspondence on DyCheck using the percentage of correctly transferred keypoints (PCK-T), following the DyCheck protocol (Gao et al., 2022) as in OriGS (Wu et al., 2025a) and SE(3) B-splineGS (Zhang et al., 2026). All methods use the same query points, target frames, visibility masks, and distance threshold. RayCo4D propagates reference queries through its shared motion field and projects them into the target camera; only observations visible in both frames are evaluated.

NVIDIA Dynamic Scene Dataset. The NVIDIA Dynamic Scene Dataset (Yoon et al., 2020) consists of synchronized multi-camera recordings of dynamic scenes. Following prior monocular dynamic reconstruction methods, we construct each training sequence from a single moving-camera video and reserve the remaining synchronized views for evaluation. We report PSNR, SSIM, and LPIPS between the rendered and ground-truth test images using the standard image resolution and evaluation split.

DAVIS. DAVIS (Perazzi et al., 2016) contains in-the-wild monocular videos with challenging non-rigid motion, occlusion, motion blur, and complex object interactions. We use DAVIS to qualitatively assess reconstruction stability on scenes outside controlled multi-view capture settings. Because DAVIS does not provide synchronized images from held-out cameras, we do not report quantitative novel-view synthesis results on this dataset.

Continuous-time evaluation. Selected intermediate frames are excluded from image, track, and depth supervision. All compared methods use the same observed frames and held-out timestamps. We render the corresponding camera views and report PSNR, SSIM, and LPIPS on NVIDIA (Tab. 3); Fig. 5 additionally shows qualitative results on DAVIS and DyCheck.

D.10 Runtime Comparison

Tab. 7 reports average wall-clock times for 200-frame sequences from DyCheck at the $2\times$ resolution setting, measured on an NVIDIA RTX PRO 6000 GPU. Pre. denotes preprocessing, and Train. denotes reconstruction training. For RayCo4D, training includes motion-scaffold optimization and the three scene-fitting stages in Tab. 5.

USPLAT4D (Guo et al., 2026) uses MoSca-generated Gaussian priors (Lei et al., 2025); its 60-minute Pre. entry includes the cost of generating this initialization.

Table 7: **Average runtime on DyCheck (min).**

Method	Pre.	Train.
MoSca (Lei et al., 2025)	20	40
Shape of Motion (Wang et al., 2025)	35	63
OriGS (Wu et al., 2025a)	30	77
RiGS (Wu et al., 2026)	37	56
SE(3) B-splineGS (Zhang et al., 2026)	27	61
USPLAT4D (Guo et al., 2026)	60	20
RayCo4D	36	50

RayCo4D takes 86 minutes in total, compared with 88–107 minutes for Shape of Motion, OriGS, RiGS, and SE(3) B-splineGS. Its 50-minute training stage is also shorter than the 56–77 minutes required by these four methods. MoSca has a lower total runtime of 60 minutes. USPLAT4D takes 80 minutes in total, including 60 minutes for preprocessing and 20 minutes for training.Together with the DyCheck results in Tab. 1, these timings characterize the trade-off between reconstruction quality and computational cost.

E Additional Experimental Results

E.1 Continuous-Time Reconstruction

Fig. 5 complements the NVIDIA results in Tab. 3 with qualitative comparisons on DAVIS, NVIDIA, and DyCheck, following Sec. D.9.

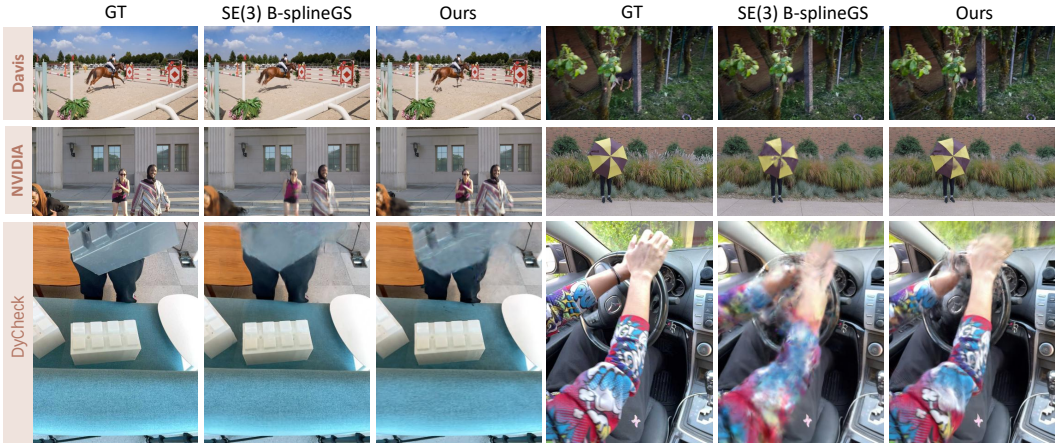


Figure 5: Continuous-time reconstruction on DAVIS, NVIDIA, and DyCheck. GT denotes intermediate frames excluded from image, track, and depth supervision. All renderings match the GT timestamps and camera views.

E.2 Challenging Cases and Limitations

Long intervals without reliable observations leave the intervening motion ambiguous. If motion changes abruptly within such an interval, the locally smooth WNOA prior may yield inaccurate intermediate motion.

The measurement–state formulation separates motion-node identity from individual tracks, but the latent states still rely on geometric observations. Large errors in tracks, estimated camera poses, or monocular depths can therefore distort the recovered motion and geometry. Future work could model measurement uncertainty, adapt temporal support to motion changes, and use stronger scene-level priors for poorly observed intervals. Fig. 6 shows examples of remaining reconstruction errors.



Figure 6: Challenging reconstruction examples on DyCheck, DAVIS, and NVIDIA. GT denotes held-out real frames excluded from training; RayCo4D renderings use the corresponding timestamps and camera views.

F Additional Qualitative Comparisons

Figs. 7–12 provide scene-wise qualitative comparisons of the full RayCo4D model and the baselines on DAVIS, NVIDIA, and DyCheck, supplementing Fig. 3 in the main paper.

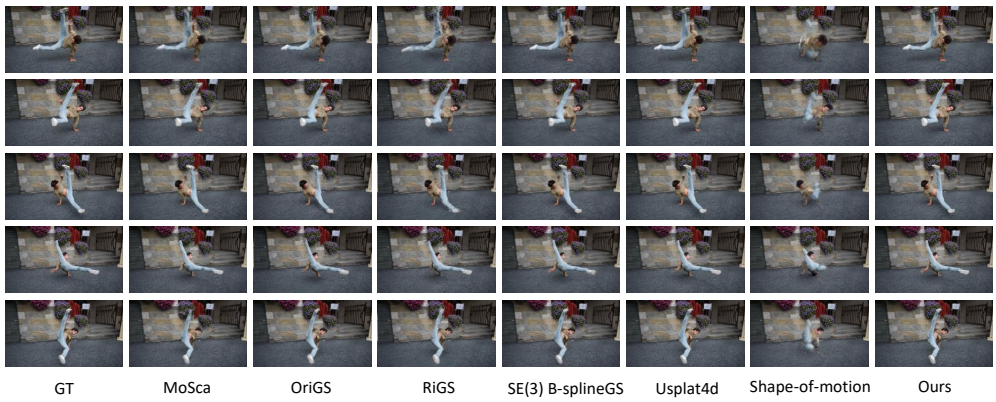


Figure 7: Qualitative reconstruction comparison on the "Breakdance-flare" scene of DAVIS (Perazzi et al., 2016).



Figure 8: Qualitative reconstruction comparison on the "Parkour" scene of DAVIS (Perazzi et al., 2016).

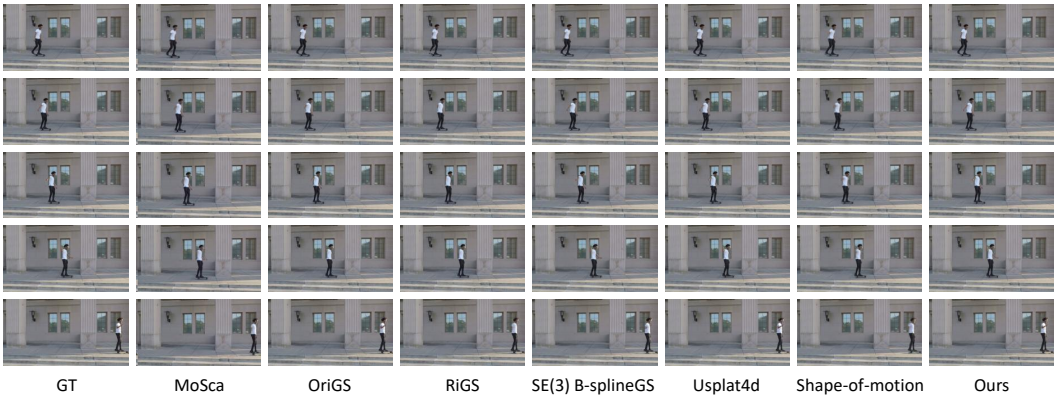


Figure 9: Visual comparison of novel-view synthesis on the "Skating" scene of the NVIDIA dataset (Yoon et al., 2020).

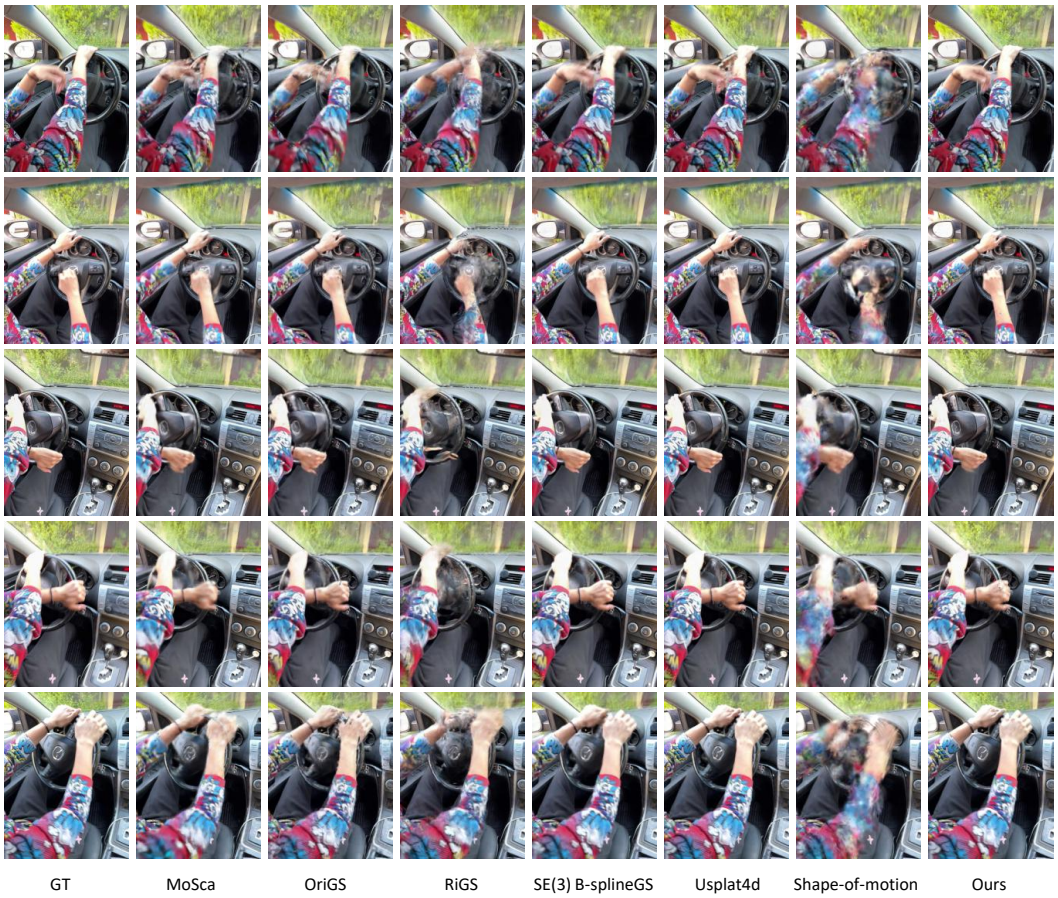


Figure 10: Visual comparison of novel-view synthesis on the "Wheel" scene of the DyCheck dataset (Gao et al., 2022).

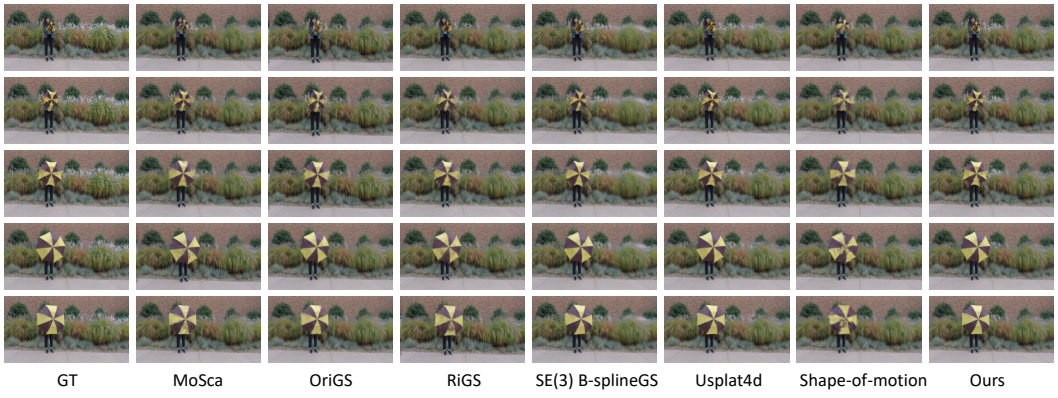


Figure 11: Visual comparison of novel-view synthesis on the "Umbrella" scene of the NVIDIA dataset (Yoon et al., 2020).

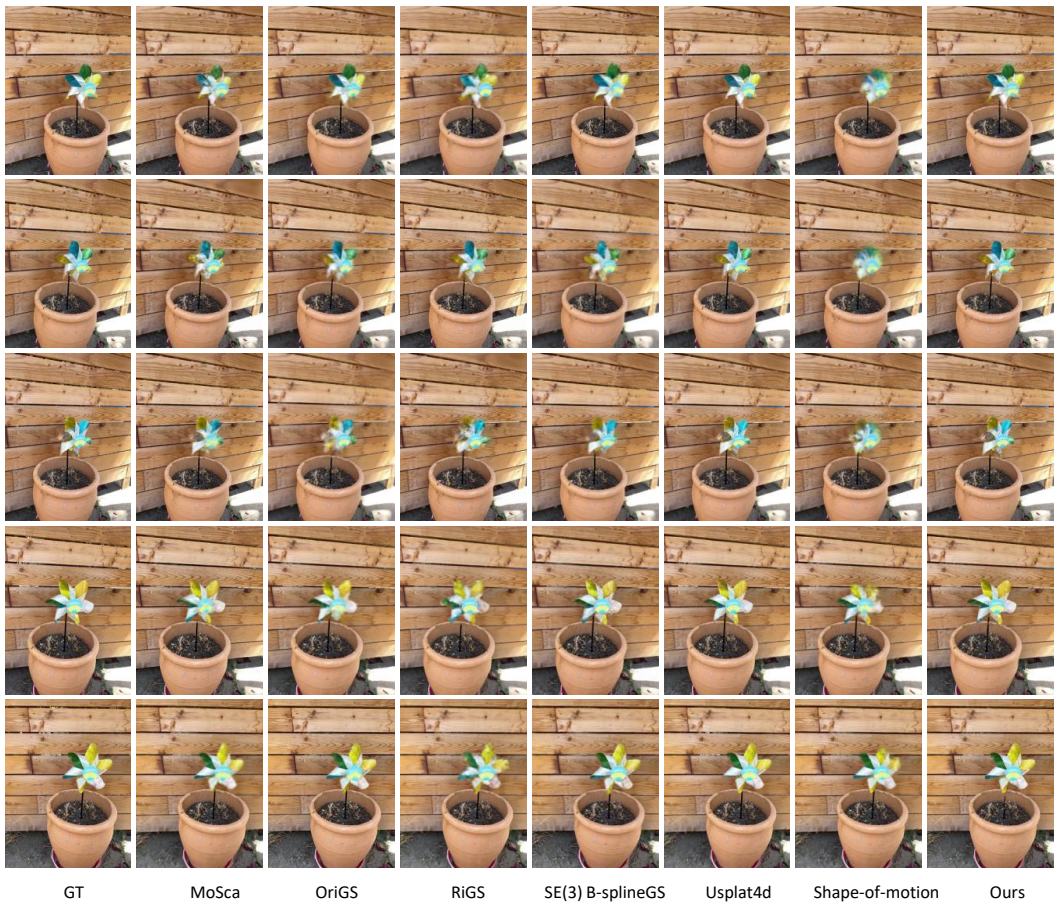


Figure 12: Visual comparison of novel-view synthesis on the "Paper-windmill" scene of the DyCheck dataset (Gao et al., 2022).