INSTANCEFLOW-SLAM: INSTANCE-GUIDED CONTINUOUS MOTION RECONSTRUCTION FOR DYNAMIC GAUSSIAN SLAM

Anonymous authors
Paper under double-blind review

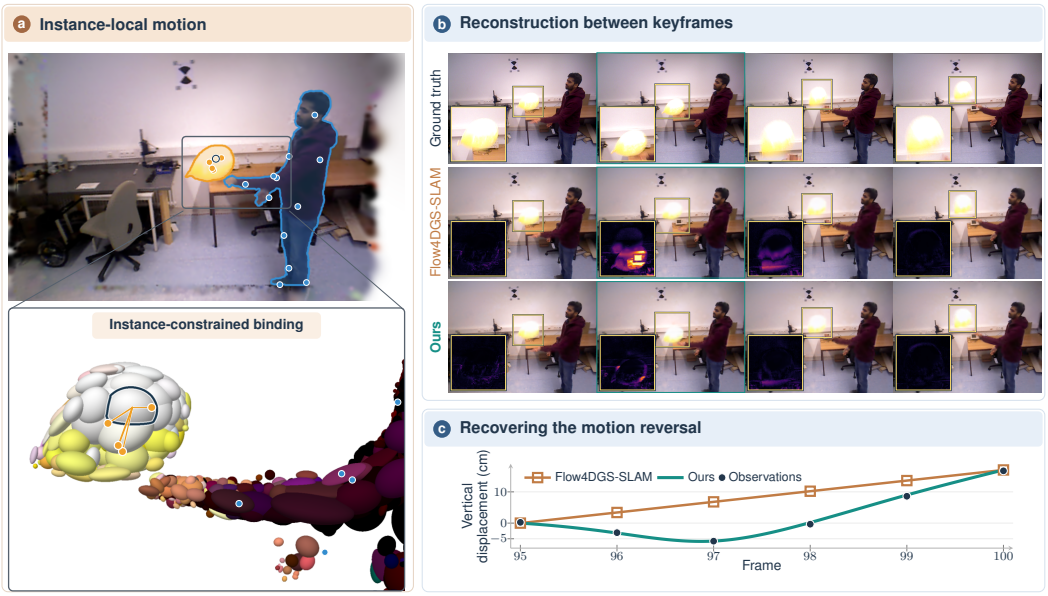

Figure 1: Instance-guided continuous motion reconstruction. (a) Instance visualization, and the highlighted Gaussian is driven by four same-instance nodes. (b) Reconstructions at frames 95, 97, 99, and 100, with balloon crops and RGB photometric error maps. (c) Balloon Gaussian displacement along the fixed upward axis of camera 95, relative to its frame-95 position. Our continuous trajectory follows tracking-and-depth observations and recovers the motion missed by linear propagation.

ABSTRACT

Keyframe-based Gaussian SLAM leaves dynamic motion between map updates underconstrained, while increasing the mapping frequency raises computational cost. We introduce InstanceFlow-SLAM, which reconstructs continuous motion from intermediate RGB-D observations without additional Gaussian mapping updates. Instance identity restricts motion sharing among Gaussians, observations, and nodes, while adaptive node allocation captures local motion variation within each object. Continuous SE(3) B-splines represent node motion and are fitted to image tracks and depth-supported 3D observations. Reliable observations determine each trajectory's temporal support, preventing unsupported endpoints from distorting the fitted motion. After keyframe mapping, boundary-consistent motion transfer preserves the recovered interior variation while aligning it with updated Gaussian states. Experiments on TUM RGB-D and Bonn demonstrate improved dynamic reconstruction with competitive camera localization. On Bonn, our method achieves the best average full-image reconstruction scores and the lowest average trajectory error among the compared methods, improving average dynamic-region PSNR by 1.44 dB over Flow4DGS-SLAM.

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

Gaussian Splatting (GS) SLAM has achieved high-quality, renderable reconstruction by tracking the camera and updating a scene map at selected keyframes (Keetha et al., 2024; Matsuki et al., 2024). This strategy works well for static environments, whose geometry remains largely consistent between map updates. Recent methods extend GS SLAM to dynamic scenes, jointly estimating camera motion and reconstructing static backgrounds and moving objects (Li et al., 2025; Wang & Lee, 2026; Wu et al., 2025; Zhu et al., 2025; Li et al., 2026). Yet keyframe updates alone may leave the motion of dynamic objects between them underconstrained. Methods relying primarily on discrete keyframe states can miss changes in direction, speed, or local shape, as illustrated in Figure 1. Increasing keyframe frequency provides more dynamic updates but also raises the cost of mapping the entire scene. Improving dynamic reconstruction without increasing the keyframe mapping workload therefore remains an open problem.

Early Gaussian SLAM methods for dynamic environments primarily addressed moving content as a source of interference. By identifying dynamic regions or estimating their uncertainty, they reduce the effect of moving objects on camera tracking and static-background reconstruction (Xu et al., 2024; Zheng et al., 2025). This strategy improves robustness, but discarding dynamic observations cannot produce a complete 4D map of the scene. A different line of work therefore reconstructs both the static background and moving objects, extending SLAM from a static 3D map to a time-varying scene representation (**?**Li et al., 2025; Wu et al., 2025; Zhu et al., 2025). Within this line of work, continuously parameterized motion models provide flexible trajectories, but fitting motion jointly with the Gaussian scene (Huang et al., 2024) can be computationally demanding (Li et al., 2025). More explicit propagation schemes reduce the cost of updating dynamic states (Wang & Lee, 2026), but motion inferred mainly from sparse keyframe states may miss changes within the interval. Thus, explicitly reconstructing moving objects has not resolved the tension between temporal detail and mapping cost.

A natural way to improve dynamic reconstruction is to use the RGB-D frames between keyframes. Treating every such frame as a keyframe, however, would substantially increase the Gaussian mapping workload. The challenge is to use their observations to *constrain dynamic motion without performing additional map optimization*. This entails three related difficulties. First, observations near contacts and occlusions may belong to different moving objects or to the static background, making motion association ambiguous. Second, motion may vary within an object: a single rigid trajectory can be insufficient, while independent trajectories for every Gaussian would lose local motion coherence. Third, tracks and depth measurements may become unreliable as visibility changes, leaving parts of a trajectory without observational support.

To address these challenges, we introduce **InstanceFlow-SLAM**, a novel 4D Gaussian SLAM system that jointly estimates camera motion and reconstructs dynamic scenes, which retains keyframe-based Gaussian mapping while proposing non-keyframe RGB-D observations to recover continuous motion of dynamic objects. To be specific, we first use instance identity to establish object-specific motion groups, which determine the associations among Gaussians, observations, and motion nodes. Given these groups, motion-adaptive nodes are allocated within each instance, and their continuous $\mathrm{SE}(3)$ B-spline trajectories are fitted to image tracks and depth-supported 3D observations, capturing local motion variation while maintaining intra-instance coherence. Since these observations may become unreliable as visibility changes, each Gaussian trajectory is constrained only over the temporal range supported by reliable evidence. After keyframe mapping updates the Gaussian states, boundary-consistent motion transfer then carries the fitted intermediate motion to the updated map, preserving continuous motion between keyframes without additional Gaussian mapping.

Our main contributions are as follows:

- We reveal that continuous dynamic reconstruction requires motion evidence to be shared within object instances and only over reliable temporal support; cross-instance coupling and unsupported terminal evidence can corrupt the recovered trajectories.
- We introduce InstanceFlow-SLAM, which labels dynamic Gaussians from instance masks at each keyframe and allocates motion nodes within each instance according to local motion variation, recovering continuous motion with observation-constrained SE(3) splines.

- We develop observation-supported trajectory boundaries and boundary-consistent motion transfer, which preserve interior nonlinear motion alongside the Gaussian states updated by keyframe mapping without per-frame dynamic mapping.
- InstanceFlow-SLAM achieves state-of-the-art dynamic scene reconstruction on TUM RGB-D and Bonn. On Bonn, it improves dynamic-region PSNR by 1.44 dB over Flow4DGS-SLAM and achieves the lowest average trajectory error.

2 Related Work

Static Gaussian Splatting SLAM. Feature-based systems maintain sparse maps for localization (Campos et al., 2021), while dense fusion methods reconstruct scene surfaces (Newcombe et al., 2011). Neural SLAM approaches use learned scene representations and differentiable rendering to optimize geometry, appearance, and camera poses (Sucar et al., 2021; Zhu et al., 2022). 3D Gaussian Splatting provides explicit primitives and fast rasterization for high-quality rendering (Kerbl et al., 2023). Subsequent Gaussian SLAM systems adapt this representation to incremental tracking and mapping, unifying scene reconstruction and image rendering (Keetha et al., 2024; Matsuki et al., 2024; Yan et al., 2024; Yugay et al., 2023; Sandström et al., 2025). Their static maps, however, do not explicitly describe object motion, motivating temporal representations for dynamic reconstruction.

Gaussian SLAM in Dynamic Scenes. Dynamic Gaussian SLAM methods either suppress moving content for static mapping or explicitly reconstruct it. DG-SLAM and WildGS-SLAM use motion masking or uncertainty estimation to reduce dynamic interference and recover static backgrounds (Xu et al., 2024; Zheng et al., 2025). CAD-SLAM and D^{2}GSLAM combine dynamic identification with separate static and dynamic representations to retain moving content (Wu et al., 2025; Zhu et al., 2025). DynaGSLAM tracks Gaussian motion and interpolates trajectories, while 4DGS-SLAM learns control-point-based neural deformation fields (Li et al., 2026; 2025). To reduce dynamic mapping overhead, Flow4DGS-SLAM uses optical flow to propagate explicit keyframe centers, alongside GMM-based temporal orientation and opacity (Wang & Lee, 2026). Its centers are linearly interpolated between keyframes, which can miss nonlinear motion even when the endpoints are accurate. Building on this framework, we use instance-local shared motion and within-window observations to recover continuous trajectories and transfer their interior variation to the map updated by keyframe mapping.

3 Method

3.1 Problem Setup and System Overview

Given an RGB-D stream $\{I_t, D_t\}_{t=0}^{N-1}$ and camera intrinsics K, InstanceFlow-SLAM estimates camera poses $\{T_t\}$ and maintains a Gaussian map $\mathcal{G} = \mathcal{G}^s \cup \mathcal{G}^d$ containing static and dynamic primitives. Each Gaussian has rendering attributes including a center, orientation, scale, opacity, and appearance. For each dynamic Gaussian, we additionally maintain an instance label ℓ_i that identifies the moving object to which it is assigned. This label governs its motion associations rather than its rendering parameters. The map is optimized at selected keyframes following the strategy of methods (Li et al., 2025; Wang & Lee, 2026), while dynamic Gaussians additionally require time-varying states between them.

Let a and b denote consecutive keyframe times. For the interval $[a, b]$, we obtain instance masks, 2D point tracks, and depth-supported 3D observations from the RGB-D stream. Once these observations are available, we fit continuous, instance-local motion to the dynamic Gaussians, perform the regular keyframe mapping update at b, and transfer the fitted interior motion to the updated map. Thus, intermediate frames constrain dynamic motion without triggering additional Gaussian mapping updates. Instance observations use a fixed two-frame look-ahead, so the system operates online with bounded delay. Camera tracking and keyframe mapping otherwise follow the underlying SLAM pipeline.

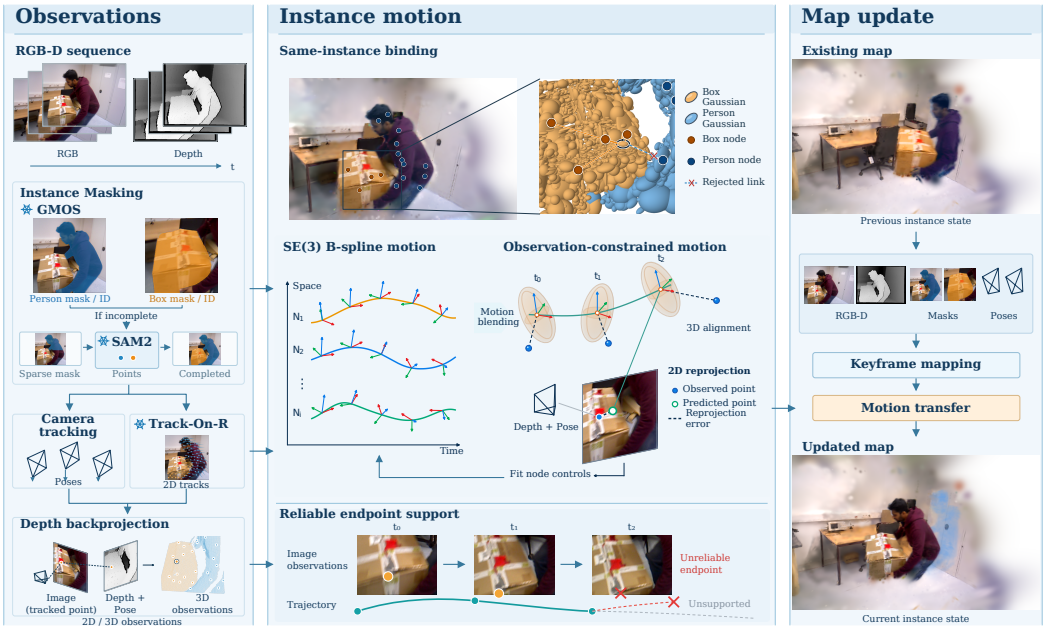

Figure 2: Overview of InstanceFlow-SLAM. RGB-D observations provide instance masks, camera poses, and 2D/3D correspondences. Dynamic Gaussians bind to same-instance nodes, whose continuous $\mathrm{SE}(3)$ B-spline motion is fitted within reliable trajectory support and integrated with keyframe mapping.

3.2 Instance-Constrained Motion Support

To fit dynamic Gaussian motion from the intermediate observations described above, we first associate each observation with the moving object it describes. This association is particularly uncertain where objects touch or overlap in the image: nearby observations may belong to different objects or to the static background. We therefore use the instance label ℓ_i to define the motion support of each dynamic Gaussian. Within a keyframe interval, Gaussian i is associated only with observations and motion nodes belonging to the same instance. This limits motion mixing across object boundaries while allowing Gaussians within an instance to share local motion.

We obtain dynamic instance masks using a moving-object segmentation model (Xie et al., 2026). When blur, occlusion, or contact leaves an instance's visible mask incomplete, we prompt SAM2 (Ravi et al., 2025) with reliable instance observations to refine it. At each keyframe, dynamic Gaussians projected into the resulting masks receive the corresponding instance labels. These labels remain fixed during motion fitting over the interval. The masks also distinguish dynamic from static image regions during camera tracking.

We track points on each dynamic instance through $[a, b]$ (Aydemir et al., 2026) and retain visible, in-image observations consistent with their source instance. Where depth is valid, we back-project tracked image points using the camera intrinsics and estimated poses to obtain world-space 3D positions. These instance-consistent 2D and 3D observations guide node allocation and continuous motion fitting. If a propagated Gaussian projects into another instance's region, we first correct it using nodes of its assigned instance; if the conflict persists, we suppress its rendering contribution during the optimization process.

3.3 Motion-Adaptive Continuous Representation within Instances

Instance labels prevent motion sharing across objects, but a single rigid transform may not explain all regions within an instance. In this section, we represent each instance with shared local motion nodes, allocating more nodes where the observations indicate distinct motion.

Motion-adaptive node allocation. Each instance initially forms one motion region $\mathcal{S}$. We test whether splitting it into two subregions better explains the instance-consistent 3D observations. At each observation time $t \in \mathcal{T}$, a rigid transform (R_t, d_t) maps the source positions x_i^a toward their observations y_{it}. For a candidate partition $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$, we compare the weighted fitting error of one shared transform with that of separate transforms for the two subregions. With valid-observation weights ω_{it} and $Z_{\mathcal{S}} = \sum_{i\in\mathcal{S},t\in\mathcal{T}} \omega_{it}$, the regional fitting error $E(\mathcal{S})$ and split gain ΔE are defined as:

$$E(\mathcal{S}) = \min_{\{R_t,d_t\}} \frac{1}{Z_{\mathcal{S}}} \sum_{i\in\mathcal{S}} \sum_{t\in\mathcal{T}} \omega_{it} \|R_t x_i^a + d_t - y_{it}\|_2^2, \tag{1}$$

$$\Delta E = E(\mathcal{S}) - \frac{Z_{\mathcal{S}_1} E(\mathcal{S}_1) + Z_{\mathcal{S}_2} E(\mathcal{S}_2)}{Z_{\mathcal{S}}}. \tag{2}$$

We generate candidate partitions by projecting the source Gaussian centers onto the principal spatial direction of $\mathcal{S}$ and splitting at the 33rd, 50th, and 67th percentiles. If the largest ΔE exceeds a threshold, we accept that split and recursively process its children. Each terminal region defines one node. Thus, node allocation depends on observed motion rather than spatial extent alone.

Continuous node motion. For each motion node n, we represent its rotation and translation over the keyframe interval with a cumulative cubic SE(3) B-spline (Sommer et al., 2020). Temporal control transforms $P_{n,k} \in \mathrm{SE}(3)$ define:

$$H_n(t) = P_{n,k-1} \prod_{j=1}^{3} \mathrm{Exp}(\beta_j(v)\xi_{n,k+j-2}), \qquad t \in [\tau_k, \tau_{k+1}]. \tag{3}$$

Here, $\xi_{n,k} = \mathrm{Log}(P_{n,k}^{-1} P_{n,k+1})$ is the relative increment between adjacent controls, $v = (t - \tau_k)/(\tau_{k+1} - \tau_k)$ is normalized segment time, β_j are cumulative cubic B-spline basis functions. Weighted rigid registration of the 3D trajectory observations of Gaussians within each node's supported region initializes its controls. Spline details appear in Appendices A.2 and A.3.

Instance-constrained motion blending. Each dynamic Gaussian i binds to up to four nearby nodes within its instance. We assign normalized weights w_{in} based on the distance between its source position and each node; the node set $\mathcal{N}_i$ and weights remain fixed throughout the interval. To make the source position the trajectory's starting point, we express node motion relative to time a as $\widetilde{H}_n(t) = H_n(t)H_n(a)^{-1}$. Dual-quaternion blending(DQB) (Kavan et al., 2007) then gives the Gaussian transform and center trajectory:

$$H_i(t) = \mathrm{DQB}\Big(\{w_{in}, \widetilde{H}_n(t)\}_{n\in\mathcal{N}_i}\Big), \qquad g_i(t) = R_i(t)x_i^a + d_i(t). \tag{4}$$

Here, $(R_i(t), d_i(t))$ are the rotation and translation components of $H_i(t)$. The resulting $g_i(t)$ is a node-driven trajectory that will be fitted to observations and adjusted at its supported boundaries in Sec. 3.4. Binding weights are detailed in Appendix A.2.

3.4 Observation-Constrained Motion Fitting

The node-driven trajectory $g_i(t)$ provides continuous motion, but its spline controls must be fitted to observations within the keyframe interval. We use the instance-consistent 2D tracks and depth-supported 3D positions from Sec. 3.2, while keeping node allocation and Gaussian-to-node bindings fixed.

Observation-supported trajectory boundaries. A track may lose reliable support before the next keyframe. For each Gaussian i, we define an effective terminal time e_i and position anchor z_i. If its terminal observation is valid, these correspond to the keyframe endpoint; otherwise, we use the last reliable depth-supported observation. We correct the node-driven trajectory by interpolating its discrepancies from the source position and terminal anchor:

$$\bar{x}_i(t) = g_i(t) + (1 - \alpha_i(t))\big[x_i^a - g_i(a)\big] + \alpha_i(t)\big[z_i - g_i(e_i)\big], \qquad \alpha_i(t) = \frac{t-a}{e_i - a}. \tag{5}$$

Thus, $\bar{x}_i(a) = x_i^a$ and $\bar{x}_i(e_i) = z_i$, while the nonlinear motion inside the supported interval is retained. If $e_i < b$, the Gaussian does not contribute to rendering after e_i in this interval.

Joint motion optimization. We optimize the spline controls by aligning the corrected Gaussian trajectories with their instance-consistent observations. Robust 2D reprojection and 3D position terms provide image-space and depth-supported world-space constraints. Where observations are sparse or noisy, a local structure term limits changes in distances between neighboring Gaussians, a temporal term discourages abrupt motion changes, and a control term limits departure from initialization:

$$\mathcal{L}_{\text{fit}} = \mathcal{L}_{\text{2D}} + \lambda_{\text{3D}}\mathcal{L}_{\text{3D}} + \lambda_{\text{local}}\mathcal{L}_{\text{local}} + \lambda_{\text{acc}}\mathcal{L}_{\text{acc}} + \lambda_{\text{ctrl}}\mathcal{L}_{\text{ctrl}}. \tag{6}$$

Detailed loss definitions are provided in Appendix A.4.

3.5 Boundary-Consistent Motion Transfer

The trajectories fitted in Sec. 3.4 precede keyframe mapping, which may change their boundary Gaussian states. To retain the observed interior motion while matching the updated map, we transfer motion relative to an interpolation of the trajectory boundaries.

Position transfer. For a trajectory supported over $[a, b]$, let $x_i^{\text{fit}}(t) = \bar{x}_i(t)$ and let $x_i^{a,\text{pre}}$ and $x_i^{b,\text{pre}}$ denote its positions before mapping. We extract the interior residual

$$r_i(t) = x_i^{\text{fit}}(t) - \left[(1-s)x_i^{a,\text{pre}} + s x_i^{b,\text{pre}}\right], \qquad s = \frac{t-a}{b-a}. \tag{7}$$

After mapping, the output trajectory is

$$x_i^{\text{out}}(t) = (1-s)x_i^{a,\text{post}} + s x_i^{b,\text{post}} + r_i(t). \tag{8}$$

Since $r_i(a) = r_i(b) = 0$, the trajectory matches the updated keyframe positions while preserving its fitted interior variation. For trajectories ending at $e_i < b$, we apply the same decomposition only over their supported interval $[a, e_i]$ and stop rendering them after e_i.

Orientation and map updates. We similarly extract the node-blended rotational variation relative to interpolation between its supported boundaries and compose that residual with the mapped Gaussian orientation. The residual is the identity at the boundaries, preserving the updated orientations. Gaussians created by densification inherit the residual of their source Gaussian; Gaussians without fitted motion retain the underlying pipeline's motion representation. The exact rotational composition and covariance update are given in Appendix A.5.

4 Experiments

4.1 Experimental Setup

Setup. We evaluate full-image and dynamic-region reconstruction, camera trajectories, and runtime to assess overall rendering quality, continuous dynamic reconstruction, localization accuracy, and practical efficiency, respectively.

Baselines. We compare reconstruction performance with MonoGS (Matsuki et al., 2024), SplaTAM (Keetha et al., 2024), SC-GS (Huang et al., 2024), 4DGS-SLAM (Li et al., 2025), and Flow4DGS-SLAM (Wang & Lee, 2026), and additionally include RoDyn-SLAM (Jiang et al., 2024) for camera-pose evaluation. Flow4DGS-SLAM is our direct baseline: we retain its keyframe mapping pipeline and introduce instance-guided continuous motion reconstruction and boundary-consistent motion transfer. For 4DGS-SLAM and Flow4DGS-SLAM, we report reproduced results without offline color refinement.

Datasets. We evaluate on two real-world RGB-D datasets, TUM RGB-D (Sturm et al., 2012) and Bonn (Palazzolo et al., 2019), using the same dynamic sequences as Flow4DGS-SLAM to facilitate direct comparison. The sequences cover varying camera motions, human activities, and object interactions, enabling evaluation of reconstruction quality and camera localization in dynamic scenes.

Metrics. We evaluate rendering with PSNR, SSIM, and LPIPS, and report dynamic-region PSNR for Flow4DGS-SLAM and our method on common frames using the same pseudo-ground-truth masks. To create these masks, we identify moving objects in each sequence, obtain candidate regions with Grounding DINO (Liu et al., 2024), and refine them with SAM2 (Ravi et al., 2025) when needed. Camera localization is measured by ATE RMSE after SE(3) alignment.



Figure 3: Dynamic segmentation comparison.

Implementation details. We process RGB-D sequences at 640×480 resolution. We use an online adaptation of GMOS to obtain instance masks and obtain trajectories using pretrained Track-On-R. We initialize node motion through weighted rigid registration and optimize it using Adam with a learning rate of 0.003 for up to 100 iterations, with early stopping upon convergence. We divide 2D and 3D residuals by 3 pixels and 0.05 m, respectively. The weights for the 3D, local structure, temporal smoothness, and control increment terms are 0.1, 0.1, 0.01, and 0.0001, respectively. Keyframe mapping uses 50 iterations.

4.2 Quality of Reconstructed Map

Tables 1–5 show competitive reconstruction quality on both datasets, with the best average full-image and dynamic-region scores on Bonn. Compared with Flow4DGS-SLAM, our full-image and dynamic-region PSNR improve by 0.30 dB and 1.44 dB on Bonn, respectively; both averages also improve slightly on TUM. Figure 5 and Appendix A.7 show more complete moving objects with fewer ghosts and less local blur.

Table 1: Quantitative results on the TUM RGB-D sequences. Pink and orange denote the top two results per column and metric.

Method	Metric	sit_st	sit_xyz	sit_rpy	walk_st	walk_xyz	walk_rpy	Avg.
MonoGS	PSNR (dB) ↑	19.95	23.92	16.99	16.47	14.02	15.12	17.74
	SSIM ↑	0.739	0.803	0.572	0.604	0.436	0.497	0.608
	LPIPS ↓	0.213	0.182	0.405	0.355	0.581	0.560	0.382
SplaTAM	PSNR (dB) ↑	24.12	22.07	19.97	16.70	17.03	16.54	19.40
	SSIM ↑	0.915	0.879	0.799	0.688	0.650	0.635	0.757
	LPIPS ↓	0.101	0.163	0.205	0.287	0.339	0.353	0.241
SC-GS	PSNR (dB) ↑	27.01	21.45	18.93	20.99	19.89	16.44	20.78
	SSIM ↑	0.900	0.686	0.529	0.762	0.590	0.475	0.657
	LPIPS ↓	0.182	0.369	0.512	0.291	0.470	0.554	0.396
4DGS-SLAM	PSNR (dB) ↑	26.49	23.48	19.95	23.01	18.94	18.87	21.79
	SSIM ↑	0.864	0.781	0.710	0.804	0.667	0.665	0.749
	LPIPS ↓	0.127	0.202	0.292	0.198	0.327	0.373	0.253
Flow4DGS-SLAM	PSNR (dB) ↑	27.22	25.45	23.20	25.25	22.00	22.66	24.30
	SSIM ↑	0.904	0.858	0.800	0.858	0.768	0.777	0.828
	LPIPS ↓	0.099	0.141	0.230	0.134	0.223	0.252	0.180
Ours	PSNR (dB) ↑	27.15	25.10	23.62	25.32	22.35	22.68	24.37
	SSIM ↑	0.902	0.853	0.808	0.859	0.778	0.782	0.830
	LPIPS ↓	0.100	0.151	0.232	0.135	0.223	0.251	0.182

Table 2: Dynamic-region PSNR (dB)↑ on the six TUM RGB-D sequences.

Method	sit_st	sit_xyz	sit_rpy	walk_st	walk_xyz	walk_rpy	Avg.
Flow4DGS-SLAM	**29.02**	26.94	24.60	24.68	22.20	21.81	24.87
Ours	29.00	**26.96**	**25.16**	**24.85**	**22.91**	**22.45**	**25.22**

4.3 Pose Estimation

Tables 4 and 6 show that our method maintains competitive localization accuracy, matching the average ATE of Flow4DGS-SLAM on TUM RGB-D and achieving the lowest average error on Bonn. These results indicate that improved dynamic reconstruction is compatible with accurate camera tracking.

Table 3: Quantitative results on the BONN sequences. "-" indicates reconstruction failure.

Method	Metric	balloon	balloon2	pt	pt2	sync	sync2	box	box2	box3	Avg.
MonoGS	PSNR (dB) ↑	21.35	20.22	20.53	20.09	22.03	20.55	20.76	19.38	24.81	21.06
	SSIM ↑	0.803	0.758	0.779	0.718	0.766	0.841	0.748	0.753	0.857	0.780
	LPIPS ↓	0.316	0.354	0.408	0.426	0.328	0.521	0.428	0.372	0.243	0.342
SplaTAM	PSNR (dB) ↑	19.65	17.67	18.30	15.57	19.33	19.67	20.81	21.69	21.41	19.34
	SSIM ↑	0.781	0.702	0.670	0.606	0.776	0.730	0.824	0.852	0.873	0.757
	LPIPS ↓	0.211	0.280	0.283	0.331	0.227	0.258	0.191	0.165	0.152	0.233
SC-GS	PSNR (dB) ↑	22.30	21.38	-	-	23.62	22.74	20.60	21.55	19.24	21.63
	SSIM ↑	0.737	0.708	-	-	0.788	0.801	0.688	0.722	0.628	0.724
	LPIPS ↓	0.448	0.450	-	-	0.427	0.359	0.515	0.491	0.539	0.461
4DGS-SLAM	PSNR (dB) ↑	25.38	23.27	20.50	18.73	24.37	24.41	20.84	23.49	24.93	22.88
	SSIM ↑	0.853	0.823	0.797	0.767	0.820	0.869	0.803	0.850	0.861	0.827
	LPIPS ↓	0.262	0.279	0.359	0.387	0.241	0.198	0.342	0.279	0.257	0.289
Flow4DGS-SLAM	PSNR (dB) ↑	26.19	25.01	25.53	25.82	27.69	27.33	25.80	26.72	26.90	26.33
	SSIM ↑	0.884	0.856	0.875	0.875	0.871	0.918	0.880	0.899	0.899	0.884
	LPIPS ↓	0.216	0.222	0.237	0.232	0.187	0.146	0.216	0.202	0.201	0.206
Ours	PSNR (dB) ↑	26.50	25.71	25.92	25.90	27.96	27.42	26.51	26.83	26.94	26.63
	SSIM ↑	0.892	0.870	0.879	0.877	0.875	0.920	0.882	0.895	0.896	0.887
	LPIPS ↓	0.207	0.216	0.235	0.231	0.185	0.144	0.208	0.208	0.212	0.205

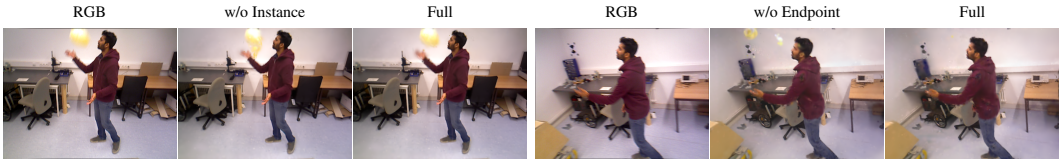

Figure 4: Qualitative ablation comparisons on BONN *balloon*



Figure 5: Qualitative comparisons on TUM RGB-D and Bonn.

4.4 Quality of Dynamic Masks

Figure 3 compares dynamic masks on Bonn. Our masks cover moving balloons and boxes beyond predefined semantic categories while separating adjacent instances. This coverage helps exclude moving content from camera tracking, while instance separation prevents observations from different objects from being pooled during motion fitting.

Table 4: Trajectory ATE RMSE [cm]↓ on the TUM RGB-D sequences.

Method	sit_st	sit_xyz	sit_rpy	walk_st	walk_xyz	walk_rpy	Avg.
RoDyn-SLAM	1.5	5.6	5.7	1.7	8.3	8.1	5.1
MonoGS	0.48	1.7	6.1	21.9	30.7	34.2	15.8
SplaTAM	0.52	1.5	11.8	83.2	134.2	142.3	62.2
4DGS-SLAM	0.72	2.3	2.9	0.60	4.4	3.2	2.3
Flow4DGS-SLAM	0.75	2.3	2.8	0.59	2.8	3.7	2.2
Ours	0.76	3.0	2.7	0.58	2.7	3.6	2.2

4.5 Ablation Studies

For the quantitative ablation, we individually remove instance constraints, nonlinear propagation, and unreliable endpoint handling while keeping the observation inputs and mapping budget fixed. Table 8 shows that all three components contribute to improved dynamic reconstruction quality. Figures 1 and 4 illustrate the corresponding failure modes: cross-instance binding mixes local motions, linear propagation misses curved trajectories, and unreliable endpoints leave local artifacts.

Table 5: Dynamic-region PSNR (dB)↑ on the BONN sequences.

Method	balloon	balloon2	pt	pt2	sync	sync2	box	box2	box3	Avg.
Flow4DGS-SLAM	24.43	22.93	23.12	24.35	26.39	24.41	21.99	25.75	25.73	24.34
Ours	**26.65**	**25.48**	**25.13**	**25.78**	**27.10**	**25.42**	**24.62**	**25.99**	**25.82**	**25.78**

Table 6: Trajectory ATE RMSE [cm]↓ on the BONN sequences.

Method	balloon	balloon2	pt	pt2	sync	sync2	box	box2	box3	Avg.
RoDyn-SLAM	7.9	11.5	14.5	13.8	1.3	1.4	4.9	6.2	10.2	7.9
MonoGS	29.6	22.1	54.5	36.9	68.5	0.56	71.5	10.7	3.6	33.1
SplaTAM	32.9	30.4	77.8	116.7	59.5	66.7	91.9	18.5	17.1	56.8
4DGS-SLAM	2.7	5.1	9.5	11.3	1.7	0.65	3.3	1.7	2.6	4.3
Flow4DGS-SLAM	3.1	3.8	7.1	11.7	0.85	0.67	3.6	2.0	2.4	3.9
Ours	3.1	4.2	7.3	10.9	0.57	0.65	2.3	1.6	2.3	3.7

Table 7: Runtime (ms) comparison.

Method	Seg. ↓	Track. ↓	Map. ↓	Motion Fit. ↓	FPS ↑
4DGS-SLAM	**16**	445	110562	—	0.04
Flow4DGS-SLAM	67	**407**	**6216**	—	**0.51**
Ours	395	437	6274	**551**	0.34

Table 8: Ablation on BONN *balloon*.

Variant	PSNR ↑	SSIM ↑	LPIPS ↓
Full	**26.844**	**0.8032**	**0.0806**
w/o instance	26.686	0.8013	0.0812
Linear	25.689	0.7904	0.0864
w/o endpoint	26.576	0.8005	0.0810

5 Conclusion

We presented InstanceFlow-SLAM, which uses instance structure and within-window observations to reconstruct continuous Gaussian motion between keyframes. The method organizes shared local motion within each instance, represents nonlinear trajectories with observation-constrained SE(3) splines, and determines trajectory boundaries from valid observation support to limit the influence of unreliable terminal correspondences. By transferring fitted interior motion variation to the updated Gaussian representation, it combines continuous motion reconstruction with map updates while retaining keyframe-based mapping. Experiments show better average reconstruction quality and localization accuracy on Bonn and competitive performance on TUM RGB-D. Ablation studies further support the effectiveness of instance constraints, nonlinear fitting, and endpoint handling.

AI USE STATEMENT

We used generative AI to improve the clarity of the manuscript and assist with parts of data processing and related code. The authors reviewed the AI-assisted text, analyses, code, and processed data to ensure correctness and consistency with the experimental procedures. The authors made all final methodological decisions and scientific judgments and retain full responsibility for the content of the paper.

REFERENCES

Görkay Aydemir, Fatma Güney, and Weidi Xie. Real-world point tracking with verifier-guided pseudo-labeling. *arXiv preprint arXiv:2603.12217*, 2026.

Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE transactions on robotics*, 37(6):1874–1890, 2021.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4220–4230. IEEE, 2024.

Haochen Jiang, Yueming Xu, Kejie Li, Jianfeng Feng, and Li Zhang. Rodyn-slam: Robust dynamic dense rgb-d slam with neural radiance fields. *IEEE Robotics and Automation Letters*, 9(9):7509–7516, 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.

Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track & map 3d gaussians for dense rgb-d slam. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21357–21366. IEEE, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Runfa Blark Li, Mahdi Shaghaghi, Keito Suzuki, Xinshuang Liu, Varun Moparthi, Bang Du, Walker Curtis, Martin Renschler, Ki Myung Brian Lee, Nikolay Atanasov, et al. Dynagslam: Real-time gaussian-splatting slam for online rendering, tracking, motion predictions of moving objects in dynamic scenes. In *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2434–2444. IEEE, 2026.

Yanyan Li, Youxu Fang, Zunjie Zhu, Kunyi Li, Yong Ding, and Federico Tombari. 4d gaussian splatting slam. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 25019–25028. IEEE, 2025.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pp. 38–55. Springer, 2024.

Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18039–18048. IEEE, 2024.

Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pp. 127–136. Ieee, 2011.

Emanuele Palazzolo, Jens Behley, Philipp Lottes, Philippe Giguere, and Cyrill Stachniss. Refusion: 3d reconstruction in dynamic environments for rgb-d cameras exploiting residuals. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7855–7862. IEEE, 2019.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 2025.

Erik Sandström, Ganlin Zhang, Keisuke Tateno, Michael Oechsle, Michael Niemeyer, Youmin Zhang, Manthan Patel, Luc Van Gool, Martin Oswald, and Federico Tombari. Splat-slam: Globally optimized rgb-only slam with 3d gaussians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1686–1697, 2025.

Christiane Sommer, Vladyslav Usenko, David Schubert, Nikolaus Demmel, and Daniel Cremers. Efficient derivative computation for cumulative b-splines on lie groups. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11145–11153. IEEE, 2020.

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 573–580. IEEE, 2012.

Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. imap: Implicit mapping and positioning in real-time. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6209–6218. IEEE, 2021.

Yunsong Wang and Gim Hee Lee. Flow4dgs-slam: Optical flow-guided 4d gaussian splatting slam. *arXiv preprint arXiv:2604.22339*, 2026.

Wenhua Wu, Chenpeng Su, Siting Zhu, Tianchen Deng, Jianhao Jiao, Guangming Wang, Dimitrios Kanoulas, Zhe Liu, and Hesheng Wang. Cad-slam: Consistency-aware dynamic slam with dynamic-static decoupled mapping. *arXiv preprint arXiv:2505.19420*, 2025.

Junyu Xie, Tengda Han, Weidi Xie, and Andrew Zisserman. Gmos: Grounding moving object segmentation in 3d space and time. *arXiv preprint arXiv:2605.30352*, 2026.

Yueming Xu, Haochen Jiang, Zhongyang Xiao, Jianfeng Feng, and Li Zhang. Dg-slam: Robust dynamic gaussian splatting slam with hybrid pose optimization. *Advances in Neural Information Processing Systems*, 37:51577–51596, 2024.

Chi Yan, Delin Qu, Dan Xu, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Gs-slam: Dense visual slam with 3d gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19595–19604. IEEE, 2024.

Vladimir Yugay, Yue Li, Theo Gevers, and Martin R Oswald. Gaussian-slam: Photo-realistic dense slam with gaussian splatting. *arXiv preprint arXiv:2312.10070*, 2023.

Jianhao Zheng, Zihan Zhu, Valentin Bieri, Marc Pollefeys, Songyou Peng, and Iro Armeni. Wildgs-slam: Monocular gaussian splatting slam in dynamic environments. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11461–11471. IEEE, 2025.

Siting Zhu, Yuxiang Huang, Wenhua Wu, Chaokang Jiang, Yongbo Chen, I Chen, Hesheng Wang, et al. D˜2 gslam: 4d dynamic gaussian splatting slam. *arXiv preprint arXiv:2512.09411*, 2025.

Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12776–12786. IEEE, 2022.

A SUPPLEMENTARY DETAILS AND RESULTS

This supplementary material details the proposed method and provides additional qualitative results and analyses.

A.1 OVERALL ALGORITHM

Algorithm 1 specifies when observations become available and how fitting precedes mapping. For arriving frame k, instance observations for $t = k - 2$ are available through the fixed two-frame look-ahead. Continuous trajectories are fitted in a closed keyframe window $[a, b]$ before mapping.

Algorithm 1 Overall Pipeline of InstanceFlow-SLAM

Require: RGB-D stream $\{(I_k, D_k)\}$, intrinsics K, look-ahead $L = 2$
Ensure: Map $\mathcal{G}$, poses $\{T_t\}$, closed-window trajectories $\mathcal{X}$

$\mathcal{B} \leftarrow \varnothing$; $\mathcal{G} \leftarrow \varnothing$; $\mathcal{X} \leftarrow \varnothing$
for each arriving frame k **do**
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{(k, I_k, D_k)\}$; $t \leftarrow k - L$
  if $t < 0$ **then continue**
  $M_t \leftarrow \mathrm{CompleteMask}(\mathrm{GMOS}(\mathcal{B}, t))$
  if $\mathcal{G} = \varnothing$ **then** $(\mathcal{G}, T_t, \ell) \leftarrow \mathrm{Init}(I_t, D_t, M_t)$; $a \leftarrow t$; **continue**
  $T_t \leftarrow \mathrm{Track}(I_t, D_t, \mathcal{G})$
  if $\neg\,\mathrm{IsKeyframe}(t)$ **then continue**
  $b \leftarrow t$; $\mathcal{G}^{\mathrm{pre}} \leftarrow \mathrm{Propagate}(\mathcal{G}, a, b)$
  $\mathcal{U} \leftarrow \mathrm{TrackPoints}(\mathcal{B}, [a, b])$
  $\mathcal{O} \leftarrow \mathrm{InstanceConsistentObservations}(\mathcal{U}, M, \ell)$
  $y_{i\tau} \leftarrow T_\tau^{-1} \pi_K^{-1}(u_{i\tau}, D_\tau(u_{i\tau}))$ for valid depths
  $(\mathcal{G}^{\mathrm{pre}}, v) \leftarrow \mathrm{RepairOrSuppress}(\mathcal{G}^{\mathrm{pre}}, \ell, \mathcal{O})$
  $\mathcal{F} \leftarrow \varnothing$ ▷ fitted source Gaussians in this window
  for each instance m with valid motion observations **do**
    $\mathcal{V}_m \leftarrow \mathrm{AdaptiveNodes}(\mathcal{O}_m, \{y_{i\tau}\}; \Delta E)$
    $(\mathcal{F}_m, \mathcal{W}_m) \leftarrow \mathrm{BindAndSetSupport}(\mathcal{V}_m, \mathcal{O}_m, \ell)$
    $P_m^0 \leftarrow \mathrm{WeightedRigidInit}(\mathcal{F}_m, \mathcal{W}_m, \{y_{i\tau}\})$
    $P_m^* \leftarrow \mathrm{FitContinuousMotion}(P_m^0, \mathcal{F}_m, \mathcal{W}_m, \mathcal{O}_m)$
    $\{r_i, \Delta Q_i\}_{i \in \mathcal{F}_m} \leftarrow \mathrm{CacheMotionResiduals}(P_m^*, \mathcal{F}_m)$
    $\mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{F}_m$
  end for
  $(\mathcal{G}^{\mathrm{post}}, \rho, \ell, v^{\mathrm{out}}) \leftarrow \mathrm{Mapping}(\mathcal{G}^{\mathrm{pre}}, \mathcal{B}, T, \ell, v)$
  $(x^{\mathrm{out}}, R^{\mathrm{out}}, v^{\mathrm{out}}) \leftarrow \mathrm{TransferResiduals}(\mathcal{G}^{\mathrm{post}}, \rho, \mathcal{F})$
  $\mathcal{X} \leftarrow \mathcal{X} \cup \{([a, b], x^{\mathrm{out}}, R^{\mathrm{out}}, v^{\mathrm{out}})\}$; $\mathcal{G} \leftarrow \mathcal{G}^{\mathrm{post}}$; $a \leftarrow b$
end for
return $\mathcal{G}, \{T_t\}, \mathcal{X}$

INSTANCECONSISTENTOBSERVATIONS retains visible tracker points whose mask label agrees with the Gaussian label. BINDANDSETSUPPORT returns $\mathcal{F}_m$ and $\mathcal{W}_m$, the fixed same-instance assignments, weights, and support intervals; it uses either the window endpoint or the last valid anchor (Sec. A.2). FITCONTINUOUSMOTION constructs dual-quaternion-blended motion, applies endpoint correction, and optimizes the objective in Secs. A.3–A.4. TRANSFERRESIDUALS applies cached position and rotation residuals to each mapped descendant's updated baseline according to its source association (Sec. A.5). Node allocation, bindings, and support intervals are fixed during fitting; fitted controls and residuals are fixed during mapping. COMPLETEMASK invokes SAM2 as needed, while REPAIRORSUPPRESS repairs reliable same-instance conflicts before suppressing persistent artifacts. Static and unfitted dynamic Gaussians retain their native representation.

A.2 INSTANCE-LOCAL BINDING AND INITIALIZATION

Binding weights. Each Gaussian i binds to its four nearest nodes within its known instance, weighted by source-position distance. Let $\mathcal{N}_i$ denote these nodes, with source positions c_n:

$$w_{in} = \frac{\exp(-\|x_i^a - c_n\|^2/(2\sigma_m^2))}{\sum_{j \in \mathcal{N}_i} \exp(-\|x_i^a - c_j\|^2/(2\sigma_m^2))}, \qquad n \in \mathcal{N}_i. \tag{9}$$

Here, σ_m is the median positive pairwise node distance within instance m; weights sum to one and remain fixed within the window.

Support-matched rigid initialization. We initialize each node from the Gaussians it actually influences, so initialization and motion blending use the same spatial support:

$$(R^0_{n,k}, d^0_{n,k}) = \underset{R\in \mathrm{SO}(3),\, d\in\mathbb{R}^3}{\arg\min} \sum_{i:n\in\mathcal{N}_i} \eta_{ink} \|R x^a_i + d - y_{i,\tau_k}\|^2. \tag{10}$$

The weights η_{ink} combine binding weights and valid observation support. We solve weighted Procrustes with determinant correction and, when needed, repeat it with residual-based robust weights on the same support.

A.3 Continuous Transforms and Motion Blending

Spline boundary evaluation. At the boundary segments, we repeat the first or last relative control increment as needed and extrapolate one virtual control transform before the first control using the inverse of the first increment. This permits cubic evaluation throughout the window.

To evaluate node motion within an interior spline segment, we use the cumulative cubic basis functions

$$\beta_1(v) = \frac{v^3 - 3v^2 + 3v + 5}{6}, \qquad \beta_2(v) = \frac{-2v^3 + 3v^2 + 3v + 1}{6}, \qquad \beta_3(v) = \frac{v^3}{6}. \tag{11}$$

Dual-quaternion blending. To combine node motions into a rigid transform, we align quaternion signs, weight their dual-quaternion representations, and normalize the result. Represent each source-relative node transform by a unit quaternion q_n and translation d_n, with scalar-first quaternion convention. Its dual part is $q^{\mathrm{d}}_n = \frac{1}{2}(0, d_n) \otimes q_n$, where $\otimes$ denotes quaternion multiplication. Before blending, align quaternion signs to a reference associated node: $\varepsilon_n = -1$ if its quaternion dot product with the reference is negative, and $\varepsilon_n = 1$ otherwise. At a fixed Gaussian and time,

$$\begin{aligned} A &= \sum_{n\in\mathcal{N}_i} w_{in}\varepsilon_n q_n, \qquad B = \sum_{n\in\mathcal{N}_i} w_{in}\varepsilon_n q^{\mathrm{d}}_n, \\ q &= \frac{A}{\|A\|}, \qquad q^{\mathrm{d}} = \frac{B}{\|A\|} - q\left\langle q, \frac{B}{\|A\|}\right\rangle. \end{aligned} \tag{12}$$

For nonzero $\|A\|$, this normalization yields a unit real part and an orthogonal dual part. The blended rotation is represented by q, and translation is $2\,\mathrm{vec}(q^{\mathrm{d}} \otimes q^{-1})$.

A.4 Motion Observations and Data Terms

To constrain motion in world coordinates, we back-project Track-On-R observations u_{it} with valid depth using the camera pose:

$$y_{it} = T_t^{-1} \cdot \pi_K^{-1}(u_{it}, D_t(u_{it}))\,, \tag{13}$$

where T_t maps world coordinates to camera coordinates and π_K^{-1} denotes depth back-projection.

To fit image projections and 3D positions jointly, we use normalized robust residuals. Let $\hat{x}_i(t)$ be the fitted position, usually $\bar{x}_i(t)$. The data terms are

$$\mathcal{L}_{2\mathrm{D}} = \frac{\sum_{i,t} \omega_{it}\, \rho((\pi_K(T_t \cdot \hat{x}_i(t)) - u_{it})/\epsilon_u)}{\sum_{i,t} \omega_{it}}, \tag{14}$$

$$\mathcal{L}_{3\mathrm{D}} = \frac{\sum_{i,t} \omega_{it} d_{it}\, \rho((\hat{x}_i(t) - y_{it})/\epsilon_x)}{\max(1, \sum_{i,t} \omega_{it} d_{it})}. \tag{15}$$

Here, ω_{it} weights valid observations, d_{it} indicates valid depth, ρ is the robust Huber loss, and ϵ_u and ϵ_x normalize image-space and world-space error scales, respectively. Retained observations satisfy visibility, image-bound, and instance-consistency checks; a tracker query shared by multiple Gaussians divides its weight equally among them. Instances without valid fitting observations

are skipped. The Huber loss is applied componentwise to normalized vector residuals with unit transition threshold.

Regularization and optimized variables. To preserve local structure, we penalize changes in neighboring-Gaussian distances. Let $h > 0$ be the spatial normalization scale, $\mathcal{T}_f$ the fitting times, and $\mathcal{E}$ the directed neighbor pairs within an instance. With $\ell_{ij}^a = \|x_i^a - x_j^a\|$,

$$\mathcal{L}_{\text{local}} = \frac{1}{|\mathcal{T}_f||\mathcal{E}|} \sum_{t\in\mathcal{T}_f} \sum_{(i,j)\in\mathcal{E}} \left(\frac{\|\bar{x}_i(t) - \bar{x}_j(t)\| - \ell_{ij}^a}{h} \right)^2. \tag{16}$$

To discourage abrupt motion changes, we penalize second differences at uniform fitting times $t_0, \ldots, t_{J-1}$. For N_g fitted Gaussians,

$$\mathcal{L}_{\text{acc}} = \frac{1}{3N_g(J-2)} \sum_i \sum_{j=1}^{J-2} \left\| \frac{\bar{x}_i(t_{j+1}) - 2\bar{x}_i(t_j) + \bar{x}_i(t_{j-1})}{h} \right\|^2. \tag{17}$$

The penalty uses sampled position differences without squared-time-step normalization. To limit departure from initialization, we penalize the dimensionless increments $\delta_{n,k} = (\theta_{n,k}, v_{n,k})$ optimized around $P_{n,k}^0$:

$$P_{n,k} = \text{Exp}\begin{pmatrix} [\theta_{n,k}]_\times & hv_{n,k} \\ 0^\top & 0 \end{pmatrix} P_{n,k}^0, \qquad \mathcal{L}_{\text{ctrl}} = \frac{1}{6N_c} \sum_{n,k} \|\delta_{n,k}\|^2, \tag{18}$$

where N_c is the total number of node controls and $[\theta]_\times$ is the skew-symmetric cross-product matrix. Only these increments are optimized during motion fitting.

Endpoint observations and data terms. A corrected hard endpoint is independent of node controls and therefore supplies no gradient to them. Ordinary two-endpoint fitting uses interior observations. To retain terminal supervision where that branch is used, we evaluate its data term at $\hat{x}_i(b) = g_i(b)$, while the output retains the hard positional boundary.

A.5 Motion Transfer Details

Position residuals and transfer. To transfer interior motion while retaining updated map boundaries, we subtract the pre-mapping endpoint interpolation from the fitted trajectory $x_i^{\text{fit}}(t) = \bar{x}_i(t)$:

$$r_i(t) = x_i^{\text{fit}}(t) - \left[(1-s)x_i^{a,\text{pre}} + s x_i^{b,\text{pre}}\right], \qquad s = \frac{t-a}{b-a}. \tag{19}$$

For trajectories retaining both endpoint constraints, $r_i(a) = r_i(b) = 0$. We add this residual to the updated endpoint interpolation. Shortened trajectories use e_i, z_i, and α_i from Sec. 3.4 in both the baseline and residual.

Rotation and covariance. To transfer interior rotation while preserving endpoint orientations, let $Q_i(t)$ be the node-blended rotation matrix and remove its endpoint spherical interpolation:

$$\begin{aligned} Q_i^{\text{end}}(t) &= \text{Slerp}(Q_i(a), Q_i(b); s)\,, \\ \Delta Q_i(t) &= Q_i(t)\left[Q_i^{\text{end}}(t)\right]^{-1}. \end{aligned} \tag{20}$$

The residual acts by left multiplication in world coordinates on the updated native orientation $R_i^{\text{native}}(t)$:

$$R_i^{\text{out}}(t) = \Delta Q_i(t) R_i^{\text{native}}(t). \tag{21}$$

Multiplication order matters. The output remains in $\text{SO}(3)$, and $\Delta Q_i(a) = \Delta Q_i(b) = I$ preserves endpoint orientations. For shortened trajectories, replace b, s by e_i, α_i. Keeping native scales $\sigma_{i1}, \sigma_{i2}, \sigma_{i3}$ gives

$$\begin{aligned} \Sigma_i^{\text{out}}(t) &= R_i^{\text{out}}(t)\,\text{diag}(\sigma_{i1}^2, \sigma_{i2}^2, \sigma_{i3}^2) R_i^{\text{out}}(t)^\top, \\ \Sigma_i^{\text{out}}(t) &= \Delta Q_i(t) \Sigma_i^{\text{native}}(t) \Delta Q_i(t)^\top. \end{aligned} \tag{22}$$

This preserves covariance eigenvalues, positive semidefiniteness, and endpoint covariances, without additional scale or color optimization.

Motion inheritance. To preserve motion through densification and pruning, we maintain source-to-descendant associations within each window. Descendants inherit their source's position and rotation residuals and apply them to their own updated baseline. Gaussians without fitted source motion retain native motion. Residuals do not accumulate across windows. Node controls and residuals are fixed during mapping; transfer acts on the output without intermediate-frame map optimization.

A.6 Boundary and Residual Properties

We characterize the interior motion retained by boundary correction and mapping updates.

Effect of boundary correction. To isolate nonlinear variation on a supported interval $[a, e_i]$, $e_i > a$, define the endpoint affine trend

$$\mathcal{A}_i[g](t) = (1 - \alpha_i(t))g_i(a) + \alpha_i(t)g_i(e_i), \qquad \alpha_i(t) = \frac{t-a}{e_i - a}. \tag{23}$$

It then becomes a target-boundary interpolation plus a zero-boundary residual:

$$\bar{x}_i(t) = (1 - \alpha_i(t))x_i^a + \alpha_i(t)z_i + \underbrace{g_i(t) - \mathcal{A}_i[g](t)}_{r_i^{[a,e_i]}(t)}. \tag{24}$$

The time-affine correction preserves $\bar{x}_i''(t) = g_i''(t)$ where second derivatives exist inside the fixed supported interval, but can change first derivatives; this property does not extend across a rendering cutoff. If g_i is affine, the residual vanishes throughout, leaving target-boundary interpolation.

Effect of mapping updates. To quantify how mapping changes the fitted trajectory, let $\Delta x_i^a = x_i^{a,\text{post}} - x_i^{a,\text{pre}}$ and $\Delta x_i^b = x_i^{b,\text{post}} - x_i^{b,\text{pre}}$. For the two-keyframe branch, substitution gives

$$x_i^{\text{out}}(t) - x_i^{\text{fit}}(t) = (1 - s)\Delta x_i^a + s\Delta x_i^b. \tag{25}$$

Thus, mapping changes only the endpoint trend and retains the fitted interior residual. Shortened trajectories use e_i, α_i instead; no zero-residual condition is imposed at an unsupported b.

A.7 Additional Qualitative Results

A.7.1 Failure Case: Delayed Dynamic Discovery

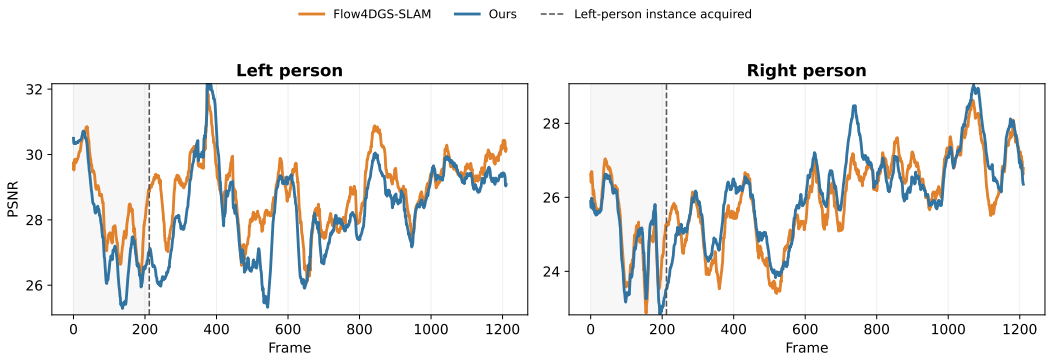

Figure 6: Per-person PSNR on TUM Sitting XYZ, evaluated on identical left/right person regions from Flow4DGS-SLAM's full-sequence dynamic masks. Orange and blue denote Flow4DGS-SLAM and Ours; the dashed line marks when our method acquires the left-person instance. Curves show centered 25-frame moving averages.

Our current system assumes that an object is identified as dynamic before its appearance is committed to the static map. This assumption can fail when an object is initially stationary and starts moving only later. In TUM Sitting XYZ, the left subject appears nearly stationary at the beginning and is absent from the accepted instance masks used by our run until source frame 220. The subject

is consequently eligible for static reconstruction during this initial period; no dynamic identity or motion trajectory is available for our continuous motion model to fit.

Figure 6 shows a localized asymmetry: after acquiring the left-person instance, our PSNR for that person is generally lower than Flow4DGS-SLAM's, while the right person's curve remains broadly comparable. Together with the frame-259 visual result, this pattern is consistent with a left-person static–dynamic conflict rather than a uniform degradation over the sequence.

Once the subject moves, assigning later observations to a new dynamic representation does not remove the earlier static copy. The resulting static–dynamic conflict can degrade both reconstruction and camera tracking. Figure 7 provides temporal evidence for this mechanism, but it is not a single-factor intervention. A complete solution requires object discovery that can retrospectively relabel previously static content, or lifecycle operations that remove or transfer the corresponding static Gaussians when motion begins.

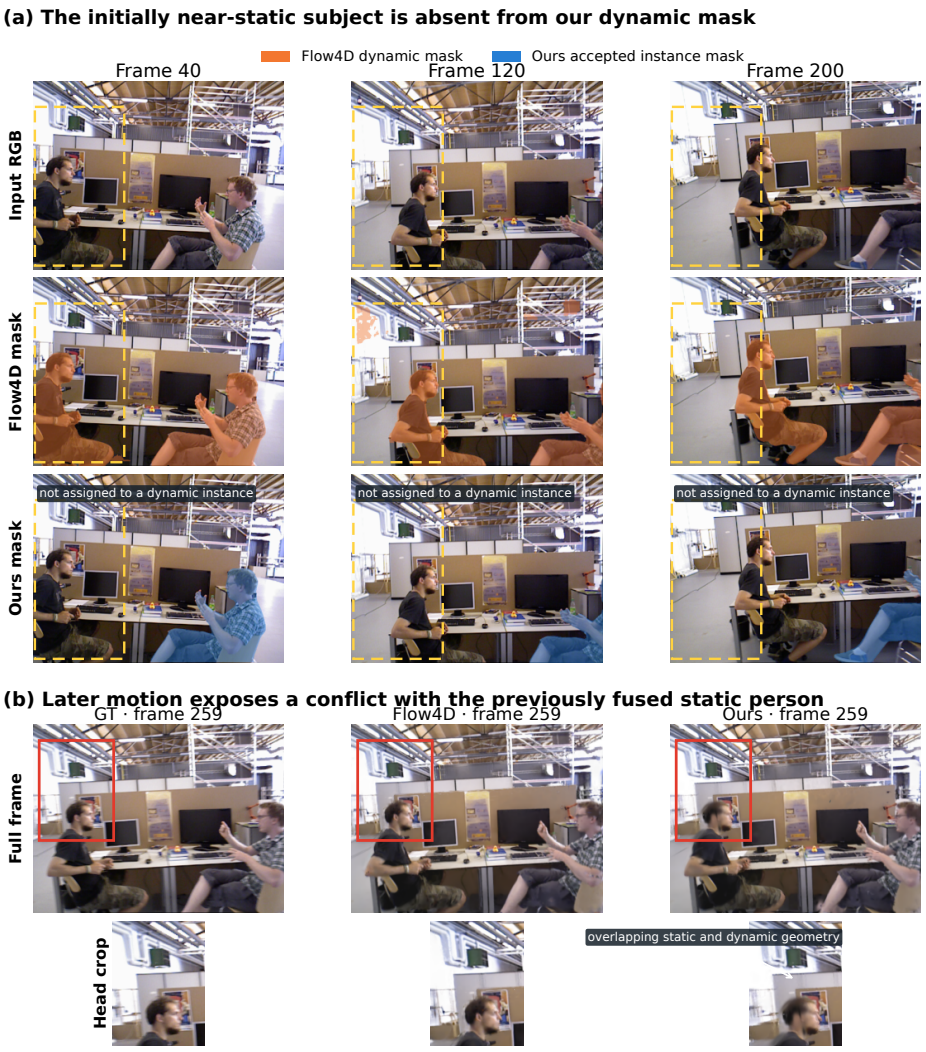

Figure 7: Failure caused by delayed motion discovery on TUM Sitting XYZ. In early frames, the left subject is nearly static. Flow4D's semantic dynamic mask covers both people, while our accepted instance mask covers only the right subject; yellow boxes indicate the missed subject. After the left subject becomes dynamic at frame 220, the frame-259 reconstruction shows a doubled, blurred head, consistent with new dynamic geometry overlapping stale geometry previously fused into the static map. This failure is due to upstream motion discovery and static-to-dynamic lifecycle handling, rather than continuous trajectory fitting once the object is already modeled as dynamic.

A.7.2 MULTI-METHOD QUALITATIVE COMPARISONS

From left to right, each row shows MonoGS, SplaTAM, 4DGS-SLAM, Flow4DGS-SLAM, Ours, and ground truth.

