REMEMBER, REVISIT, REVISE: SCAFFOLD MEMORY FOR LONG-HORIZON 3D GEOMETRY

Anonymous authors
Paper under double-blind review

ABSTRACT

In 3D computer vision, geometry foundation models estimate camera poses and 3D scene structure from video sequences, but maintaining consistent reconstructions over long horizons remains challenging. Feed-forward models process long sequences as independent, fixed-length frame windows and subsequently align their predictions, causing pose and reconstruction errors to accumulate across windows. Streaming models retain information from previous frames to inform future predictions, but cannot revise earlier estimates when subsequent observations reveal errors. To address these limitations, we introduce Scaffold MemOry for Revision (SMORE), a framework that equips existing geometry foundation models with a revisable external memory for long-horizon inference. Inspired by hippocampal associative memory in biological brains, which links spatial context with sensory experience, SMORE associates *where* an observer is with *what* it observes, enabling past observations to be stored, retrieved, verified against new evidence, and revised when inconsistencies are detected. Crucially, SMORE integrates with existing geometry foundation models while keeping them frozen, requiring no additional training or fine-tuning. We evaluate SMORE with eight competitive geometry models on seven datasets spanning camera pose estimation, multi-view depth estimation, 3D point-cloud reconstruction, simultaneous localization and mapping (SLAM), and novel view synthesis. Across all the evaluated geometry models, SMORE consistently improves long-sequence camera pose estimation, reduces multi-view depth error by up to 51%, reduces Chamfer distance for 3D point-cloud reconstruction by up to 44%, and achieves average trajectory error as low as 2.9 cm in long-horizon SLAM. By retaining corrected camera poses and observations, SMORE further improves the fidelity and spatial consistency of large-scale 3D scene reconstruction. These results establish revisable external memory as a flexible interface for extending pretrained geometry models beyond fixed-context prediction toward iterative, error-correcting inference over long visual sequences. Code, data, and models will be publicly available upon publication.

1 INTRODUCTION AND RELATED WORKS

Recovering camera motion and 3D scene structure from visual observations is fundamental to 3D computer vision, underpinning robotic navigation, autonomous systems, augmented reality, and embodied AI. Recent geometry foundation models have made substantial progress toward this goal. DUSt3R (Wang et al., 2024b) introduced point-map regression for unified geometric inference without known camera calibration or poses, followed by MASt3R (Leroy et al., 2024), Fast3R (Yang et al., 2025), VGGT (Wang et al., 2025a), VGGT-Ω (Wang et al., 2026a), and π^3 (Wang et al., 2026b). These models estimate camera poses, depth, correspondences, and 3D scene structure directly from images. Yet, a fundamental challenge remains: how can they maintain accurate, spatially consistent reconstructions over long horizons?

Existing geometry models largely approach long sequences in two ways, both with important limitations (Fig. 1). First, feed-forward models such as VGGT, VGGT-Ω, and π^3 (Wang et al., 2025a; 2026a;b) jointly infer geometry from groups of images. Long sequences are divided into fixed-length windows, reconstructed separately, and subsequently aligned into a common coordinate system. Small reconstruction or alignment errors can accumulate across windows, degrading camera poses

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.

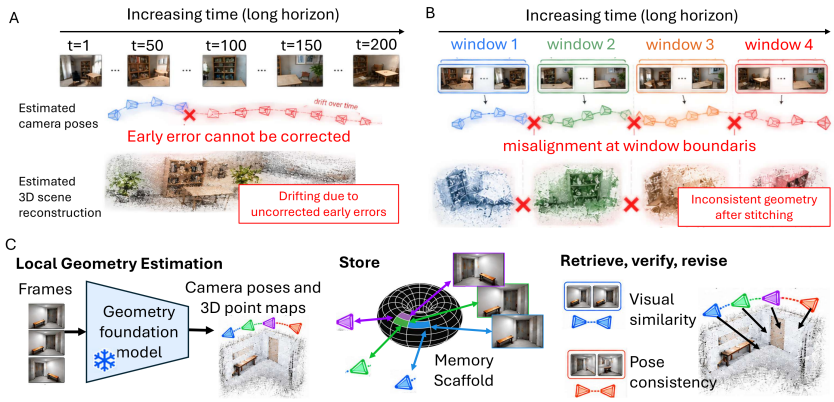


Figure 1: **Limitations of existing 3D geometry models for long horizon and overview of our SMORE. A Limitations.** Feed-forward models process long sequences in separate windows and subsequently stitch their predictions, causing errors to accumulate at window boundaries and yielding inconsistent camera poses and 3D geometry. **B Limitations.** Video streaming models retain information across video frames to inform future predictions, but cannot revise earlier estimates when subsequent observations reveal errors, leading to persistent pose drift and distorted geometry over long sequences. **C SMORE Overview.** SMORE introduces four memory operations: *Store* retains frames, camera poses, and 3D point maps in the content bank, while the scaffold associates spatial and visual information. *Retrieve* uses the current frame's visual descriptor to find candidate past frames. *Verify* jointly processes the current window and retrieved pairs to see whether their past poses support consistent window placement. *Revise* uses the accepted evidence to update the global placements of affected windows, moving their stored cameras and geometry together.

and 3D geometry over long trajectories. TALO (Zhang et al., 2026a) and AMB3R (Wang & Agapito, 2026) address this through submap alignment and compact volumetric representations, respectively, while VGGSfM (Wang et al., 2024a) and GlueMap (Pan et al., 2026) combine learned geometry with structure-from-motion. Nevertheless, long-horizon reconstruction still requires maintaining geometric consistency across observations or submaps.

Second, streaming geometry models such as CUT3R (Wang et al., 2025b), StreamVGGT (Zhuo et al., 2026), and STream3R (Lan et al., 2026) maintain information across successive observations through persistent states or causal caches. Such memory allows previous observations to inform future predictions, but retaining past context and explicitly revising previously estimated camera poses and scene geometry are different capabilities. Relatedly, EAGLE (Cao et al., 2026) combines visual and geometric memory for visual-query localization in egocentric video, but does not target retrospective correction of long-horizon 3D reconstruction. Later observations may reveal that an earlier estimate was incorrect, yet existing streaming memory does not explicitly retrieve, verify, and revise that estimate. Thus, remembering the past is not equivalent to revising it.

The need for retrospective correction is well established in simultaneous localization and mapping (SLAM), where errors are corrected through loop closure and global optimization, as in graph-based SLAM (Grisetti et al., 2010; Strasdat et al., 2010), ElasticFusion (Whelan et al., 2015), and ORB-SLAM3 (Hu et al., 2022). Learned systems such as DROID-SLAM (Teed & Deng, 2021), NICER-SLAM (Zhu et al., 2024), and SLAMFormer (Yuan et al., 2026) exploit geometric constraints across observations to correct accumulated errors. More recent works, including MASt3R-SLAM (Murai et al., 2025), VGGT-SLAM (Maggio et al., 2025), VGGT-SLAM++ (Mandal et al., 2026), and ViSTA-SLAM (Zhang et al., 2026b), combine learned geometry with loop closure, while VGGT-SLAM 2.0 (Maggio & Carlone, 2026) uses model attention to verify retrieved images. However, these are dedicated SLAM systems built around specific representations and optimization pipelines. This motivates a general question: can existing geometry foundation models be equipped with a shared external memory to recall, verify, and revise past predictions while remaining frozen?

Beyond computer vision, recurrent neural fields and continuous attractor networks (CANs) in neuroscience provide biologically inspired mechanisms for maintaining and representing continuous variables through persistent neural activity (Amari, 1977; Burak & Fiete, 2009). Vector-HaSH (Chandra et al., 2025) further combines structured spatial representations with

associative memory, linking spatial states with sensory observations. This principle suggests a natural way to organize long-horizon visual memory: associate *where* an observer is with *what* it observes, so that past observations can be retrieved when related spatial states are revisited.

Building on this principle, we introduce Scaffold MemOry for Revision (SMoRe), a revisable external memory for extending existing geometry foundation models to long-horizon inference. SMoRe goes beyond associative retrieval by supporting an explicit cycle of *storage, retrieval, verification, and revision*. As new observations arrive, SMoRe retrieves relevant past observations, compares them with current geometric evidence, verifies their consistency, and revises previously estimated poses and geometry when inconsistencies are detected. Corrected observations are retained in memory and can subsequently provide improved geometric context for later inference.

Importantly, SMoRe is external to the geometry backbone. It therefore does not require a new geometry model to be trained for long-horizon processing. Instead, existing geometry foundation models can be integrated with SMoRe while their model weights remain frozen, requiring neither additional training nor fine-tuning. This design separates the problem of learning strong geometric priors from the problem of maintaining and correcting information over time: the foundation model provides geometric inference, while SMoRe provides persistent and revisable memory. Consequently, the same memory framework can be applied across different geometry backbones and tasks.

We evaluate SMoRe with eight competitive geometry models on seven datasets spanning camera pose estimation, multi-view depth estimation, 3D point-cloud reconstruction, simultaneous localization and mapping (SLAM), and novel view synthesis. Across the evaluated geometry models, incorporating SMoRe consistently improves long-sequence camera pose estimation. It reduces multi-view depth error by up to 51%, reduces Chamfer distance for 3D point-cloud reconstruction by up to 44%, and achieves average trajectory error as low as 2.9 cm in long-horizon SLAM. The corrected poses and observations retained by SMoRe further improve the fidelity and spatial consistency of large-scale 3D scene reconstruction. Our contributions are threefold:

1. Revisable external memory for long-horizon 3D geometry. We introduce SMoRe, an external memory that associates spatial context with visual observations and supports explicit storage, retrieval, geometric verification, and revision of previously estimated poses and scene structure.

2. Training-free integration with frozen geometry foundation models. SMoRe provides a model-agnostic interface for augmenting existing geometry foundation models while keeping their weights frozen, enabling long-horizon error correction without additional training or fine-tuning.

3. Comprehensive evaluation across models and 3D tasks. We evaluate SMoRe with eight geometry models on seven datasets across camera pose estimation, multi-view depth estimation, 3D point-cloud reconstruction, SLAM, and novel view synthesis, demonstrating consistent improvements in long-horizon geometric accuracy and spatial consistency.

2 Our Proposed Scaffold MemOry for Revision (SMoRe)

We introduce SMoRe, a revisable external memory for extending pretrained geometry foundation models to long-horizon inference (Fig. 1). SMoRe consists of a spatial *scaffold* and a *content bank*. The content bank stores past observations and their geometric predictions, while the scaffold associates *where* an observation was captured with *what* was observed through a shared memory representation. As new observations arrive, SMoRe stores their spatial and visual representations and uses visual content to retrieve candidate revisits. Because associative retrieval alone cannot determine whether a candidate is a genuine revisit, retrieved observations are verified against their historical poses before being used for correction. A verified revisit provides a global spatial anchor for revising accumulated placement errors. The geometry foundation model remains frozen throughout; only the external memory is updated online. Before describing these memory operations, we first define the overlapping frame windows used for local reconstruction and the local and global coordinate systems used to place and revise these reconstructions.

Given RGB frames $I_1, \ldots, I_N$, we process them in overlapping windows $\mathcal{W}_k$. By default, each window contains 32 frames, with 16 shared between consecutive windows. See Table S6(Group G) for the effect of window size and overlap. Because a frame can appear in two overlapping windows, it is stored only once, using the first window in which it appears. We call this its *source window*,

denoted $\mathrm{src}(i)$ for frame I_i. For each window, the geometry foundation model predicts camera poses and 3D point maps in a local coordinate system defined by the camera pose of the first frame in that window. We define the global coordinate system as that of the first window and let $T_k \in \mathrm{Sim}(3)$ denote the transformation that places $\mathcal{W}_k$ in this global coordinate system. Consecutive windows are initially placed by aligning the predicted camera poses of their overlapping frames. Camera poses and point maps remain stored in their source window's local coordinate system, while T_k determines their global placement. The robust similarity alignment and the effect of changing T_k on stored geometry are detailed in Appendices B and C.

2.1 Associating Spatial and Visual Memory

After processing each window, SMoRE stores its new frames in the content bank together with their predicted camera poses and point maps. Rather than storing global poses in a lookup table indexed by frame identity, SMoRE builds an *associative memory*: spatial states and visual observations are linked through a shared memory representation. This is important because, at retrieval time, the identity and location of a revisited frame are unknown. A new visual observation must therefore serve as a partial cue for recalling previously associated spatial information.

Spatial scaffold. Following Vector-HaSH (Chandra et al., 2025), we represent each frame's global camera pose using a structured spatial scaffold. Camera position is encoded by three periodic spatial modules operating at different spatial scales, while yaw, pitch, and roll are represented by three continuous attractor rings. The periodic representation preserves the continuous structure of position and orientation, while combining multiple spatial scales reduces the ambiguity introduced by any single periodic representation. Their concatenation forms the scaffold state $\mathbf{g}_i$.

We map $\mathbf{g}_i$ through a fixed random linear projection to a 1024-dimensional memory representation $\mathbf{h}_i$, retaining only the 64 largest activations followed by normalization. This sparse intermediate representation is a deliberate design choice: activating only a small subset of memory units reduces interference among stored associations, thereby increasing effective memory capacity and making recall more robust to partial or noisy cues. We systematically vary the sparsity level and scaffold dimension and study their effects in Table S6(Groups B and D). A reverse mapping, learned using ridge regression, reconstructs the spatial scaffold state from $\mathbf{h}_i$. Thus, unlike a lookup table that can only return an entry given its explicit index, the scaffold supports associative reconstruction of spatial information from a shared memory state (see more analysis in Appendix K.1 and Table S6(Group A)). See Appendix D for details of scaffold encoding and projection.

Associating visual observations with the scaffold. To connect *what* is observed with *where* it was observed, we encode each frame I_i using a frozen DINOv2 ViT-S/14 encoder (Oquab et al., 2024), yielding a normalized 384-dimensional visual descriptor $\mathbf{s}_i$. We learn bidirectional associations between $\mathbf{s}_i$ and the shared memory representation $\mathbf{h}_i$: visual content can cue its associated memory state, while the memory state can reconstruct its associated visual representation. Both mappings are updated incrementally using recursive least squares (RLS) whenever a new frame is stored.

Together, these associations connect the spatial scaffold state $\mathbf{g}_i$, shared memory representation $\mathbf{h}_i$, and visual descriptor $\mathbf{s}_i$. Consequently, a new visual observation can provide a partial cue for recalling memories associated with similar past observations, rather than requiring the identity or location of the corresponding frame to be known in advance. This associative recall is central to SMoRE: it enables *what* is currently observed to cue *where* related observations were previously encountered. Because such recall alone cannot establish that two observations correspond to the same physical location, we use it only to propose candidate revisits, which are subsequently verified geometrically (Sec. 2.2). All memory updates are external to the pretrained models; the geometry foundation model and DINOv2 encoder remain frozen. Instead of DINOv2, we evaluate alternative visual features in Appendix K.2 and Table S6(Group C).

2.2 Retrieving and Verifying Revisits for Window Revision

Retrieving candidate revisits based on visual feature similarity. Given a current frame I_q, we compute its visual descriptor $\mathbf{s}_q$ and use the learned visual association to obtain a query memory representation $\widetilde{\mathbf{h}}_q$. Retrieval uses only visual information; the current camera pose and 3D geometry

are not used. We do not directly trust the spatial state recalled from $\widetilde{\mathbf{h}}_q$, because a novel observation may still activate an existing memory. Instead, we compare $\widetilde{\mathbf{h}}_q$ with all stored memory representations and select the five most similar frames. We then remove candidates that are within 64 frames of the current observation, originate from the immediately preceding window, or have DINOv2 cosine similarity below 0.5. The remaining frames are treated as candidate revisits and are geometrically verified next. Full retrieval details are provided in Appendix D.

Verifying revisits by historical pose consistency. For each retrieved candidate, we select a second nearby frame (± 3 frames) from the same source window to form a *retrieved pair*. We retain two pairs from different source windows, providing two independent historical references. If fewer than two valid pairs are available, no retrieval-based correction is applied. We use two pairs by default and study this choice in Table S6(Group E). When two pairs are available, the geometry foundation model jointly processes the 32 current-window frames and four retrieved frames, placing all 36 frames in a common local coordinate system. Each retrieved pair connects this joint prediction to its previously placed source window and independently proposes a global placement for the current window, denoted $T_k^{(a)}$ and $T_k^{(b)}$. We accept the revisit only if the two proposals are consistent in both rotation and translation: their rotational difference must be at most 3°, and their positional difference must be at most 0.15 times the spatial extent of the current reconstruction. Otherwise, the retrieval is rejected. Once verified, all four retrieved frames are jointly used to robustly estimate a single corrected global placement T_k^{corr}. Importantly, T_k^{corr} is fitted jointly from both historical references rather than obtained by averaging $T_k^{(a)}$ and $T_k^{(b)}$. The alignment, verification, and corrected-placement formulations are provided in Appendix F.

Progressive placement revision along the trajectory. The difference between the original placement T_k and corrected placement T_k^{corr} represents the accumulated placement error revealed by the revisit. Rather than applying the correction abruptly to the current window, we progressively distribute it across the windows between the earlier historical reference and the current window. The historical reference remains fixed, intermediate windows receive increasingly larger corrections, and the current window receives the full correction. We interpolate these corrections in $\mathrm{Sim}(3)$; the full formulation is provided in Appendix G. See Appendix K.3 and Table S6(Group F) for an analysis of alternative placement revision strategies. Only the global placement of each affected window is revised. Its stored camera poses and point maps remain expressed in the local coordinate system of its source window and therefore move together under the updated global placement.

3 EXPERIMENTS

We evaluate all competitive baselines with and without SMORE across five downstream tasks: camera pose estimation, multi-view depth estimation, 3D point-cloud reconstruction, simultaneous localization and mapping (SLAM), and novel view synthesis. We perform three runs per experiment and aggregate results across runs. For each task, we describe the problem setting and datasets, followed by the compared methods, metrics, and results. Unless otherwise stated, all main-paper SMORE results use only online retrieval, verification, and revision, without offline global optimization. We focus on this more challenging and practical setting, where corrections are made as observations arrive without access to the complete sequence. Results with offline pose-graph optimization are provided in Appendix K.

3.1 CAMERA POSE ESTIMATION

Problem setting and datasets. We evaluate long-horizon camera pose estimation on CO3Dv2 (Reizenstein et al., 2021) and RealEstate10K (RE10K) (Zhou et al., 2018). CO3Dv2 contains object-centric videos spanning 8–68 s, with 200 annotated frames per sequence. RE10K contains predominantly forward-moving real-estate videos; we use 100 clips spanning 6.7–9.3 s. For both datasets, we uniformly sample $N \in \{10, 50, 100, 200\}$ frames from each trajectory. **Baselines.** We evaluate six feed-forward geometry foundation models: DUSt3R (Wang et al., 2024b), Fast3R (Yang et al., 2025), MASt3R (Leroy et al., 2024), VGGT (Wang et al., 2025a), VGGT-Ω (Wang et al., 2026a), and π^3 (Wang et al., 2026b), together with two streaming models: StreamVGGT (Zhuo et al., 2026) and STream3R (Lan et al., 2026). For each model, we compare its original camera predictions with those obtained using SMORE on identical sampled frames. Feed-forward models use 32-frame

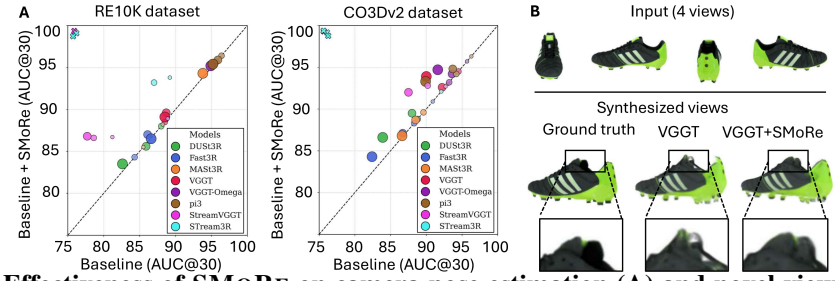


Figure 2: **Effectiveness of SMoRe on camera pose estimation (A) and novel view synthesis (B). A.** Camera pose accuracy across sequence lengths on RE10K (left) and CO3Dv2 (right). Each marker compares mean AUC@30 without (x-axis) and with SMoRe (y-axis); colors denote geometry models and marker size denotes the number of input frames, $N \in \{10, 50, 100, 200\}$. Points above the diagonal indicate improvement with SMoRe. Crosses indicate cases where the baseline runs out of memory (OOM), while the same model with SMoRe successfully performs camera pose estimation. **B.** Novel view synthesis on Google Scanned Objects (GSO). Given four input views, VGGT (Wang et al., 2025a) synthesizes a held-out target view with and without SMoRe. Black boxes indicate enlarged regions, highlighting finer reconstruction details.

windows with and without SMoRe. Streaming baselines use their native causal inference, while their SMoRe variants use the same 32-frame windowing policy. **Metrics.** We evaluate all pairs of sampled frames. For each pair, pose error is the larger of the rotation and translation-direction errors, so lower is better. AUC@30 instead measures the normalized area under the cumulative fraction of frame pairs whose pose error falls below thresholds up to 30°; therefore, higher AUC@30 is better.

Results. As shown in the right panel of Fig. 2A, SMoRe consistently improves camera pose estimation on CO3Dv2, with nearly all points lying above the diagonal. The gains become more pronounced for longer sequences, with larger markers generally farther above the diagonal. At $N = 200$, for example, VGGT improves from 90.0 to 93.9 AUC, VGGT-Ω from 91.6 to 94.7, and π^3 from 89.9 to 93.3. On RE10K (left), the gains are smaller but remain evident across sequence lengths, including shorter trajectories. This is consistent with its predominantly forward camera motion providing fewer opportunities to revisit previously observed regions. Notably, some streaming baselines run out of memory (OOM) on longer sequences in both datasets, whereas the same models equipped with SMoRe successfully perform camera pose estimation through memory-efficient windowed processing (crosses in Fig. 2A).

3.2 Multi-view Depth Estimation and 3D Point-Cloud Reconstruction

Problem setting and datasets. We evaluate *multi-view depth estimation* on Replica (Straub et al., 2019) and *3D point-cloud reconstruction* on ScanNet++ (Yeshwanth et al., 2023). Replica contains 18 photorealistic synthetic indoor environments with approximately 2,000 frames per scene, providing long camera trajectories and dense reference geometry for multi-view depth evaluation. ScanNet++ contains 1,006 high-fidelity real-world indoor scenes with high-resolution RGB observations and laser-scanned reference geometry, with long sequences spanning each indoor environment. For both datasets, we process the sequences using the 32-frame windowing strategy in Sec. 2. These long sequences require reconstruction across many successive windows, providing a challenging setting for evaluating whether SMoRe can reduce accumulated geometric inconsistencies through revisited observations. **Baselines.** We use the same eight geometry models (Sec. 3.1), comparing each baseline with its SMoRe counterpart. **Metrics.** We report reconstruction accuracy, completeness, and overall error in centimetres. Accuracy measures the distance from reconstructed to reference geometry, while completeness measures the reverse direction. Overall error is their mean; lower is better.

Results. Table 1 shows consistent improvements with SMoRe across all eight geometry models on both datasets. On Replica, SMoRe reduces overall error by approximately 39–51% across the evaluated models. For example, VGGT improves from 4.60 to 2.49 cm and VGGT-Ω from 3.84 to 2.00 cm. On ScanNet++, VGGT improves from 10.90 to 6.05 cm and VGGT-Ω from 9.25 to 4.95 cm, with similar gains observed for the streaming models. These results show that revising previously placed reconstructions with SMoRe improves long-horizon geometry estimation for both feed-forward and streaming models.

Table 1: **Long-horizon multi-view depth estimation on Replica and 3D point-cloud reconstruction on ScanNet++.** We report reconstruction accuracy (Acc.), completeness (Comp.), and overall error in centimetres. Lower are better (↓). Results as mean ± standard deviation of baslines with and withou SMoRe are presented. with the better results for each model shown in bold.

	Model	Method	Replica (centimetres)			ScanNet++ (centimetres)		
			Acc.↓	Comp.↓	Overall ↓	Acc. ↓	Comp.↓	Overall↓
Feed-forward	DUSt3R	Baseline	6.42±0.35	5.18±0.41	5.80±0.38	15.24±1.10	14.12±1.25	14.68±1.15
		+SMoRe	**3.85±0.22**	**3.24±0.19**	**3.55±0.20**	**8.51±0.72**	**7.88±0.65**	**8.20±0.68**
	Fast3R	Baseline	7.15±0.42	5.54±0.33	6.35±0.35	16.85±1.45	15.20±1.30	16.03±1.35
		+SMoRe	**4.21±0.25**	**3.55±0.28**	**3.88±0.26**	**9.22±0.85**	**8.45±0.80**	**8.84±0.82**
	MASt3R	Baseline	5.88±0.31	4.75±0.25	5.32±0.27	13.55±1.05	12.82±1.10	13.19±1.08
		+SMoRe	**3.15±0.18**	**2.92±0.15**	**3.04±0.16**	**7.45±0.60**	**6.95±0.55**	**7.20±0.58**
	VGGT	Baseline	4.95±0.28	4.25±0.22	4.60±0.24	11.25±0.95	10.55±0.88	10.90±0.91
		+SMoRe	**2.55±0.15**	**2.42±0.12**	**2.49±0.13**	**6.25±0.52**	**5.85±0.48**	**6.05±0.50**
	VGGT-Ω	Baseline	4.12±0.20	3.55±0.18	3.84±0.19	9.55±0.82	8.95±0.75	9.25±0.78
		+SMoRe	**2.05±0.11**	**1.94±0.10**	**2.00±0.10**	**5.15±0.41**	**4.75±0.38**	**4.95±0.39**
	π^3	Baseline	4.55±0.25	3.85±0.22	4.20±0.23	10.25±0.90	9.75±0.85	10.00±0.88
		+SMoRe	**2.25±0.14**	**2.12±0.12**	**2.19±0.13**	**5.55±0.45**	**5.25±0.42**	**5.40±0.43**
Streaming	StreamVGGT	Baseline	3.85±0.18	3.45±0.15	3.65±0.16	8.85±0.70	8.25±0.65	8.55±0.68
		+SMoRe	**1.92±0.09**	**1.75±0.08**	**1.84±0.08**	**4.65±0.35**	**4.35±0.32**	**4.50±0.33**
	STream3R	Baseline	3.55±0.15	3.25±0.12	3.40±0.13	8.25±0.62	7.65±0.58	7.95±0.60
		+SMoRe	**1.72±0.08**	**1.64±0.07**	**1.68±0.07**	**4.25±0.31**	**3.95±0.28**	**4.10±0.29**

3.3 Simultaneous Localization and Mapping

Problem setting and dataset. We evaluate SLAM on all seven indoor scenes of 7-Scenes (Shotton et al., 2013), where the objective is to estimate the camera trajectory while reconstructing the environment from an incoming video stream. Each sequence contains 17–33 s of continuous video, providing long trajectories for evaluating accumulated drift and revisit-driven correction. **Baselines.** We compare against published calibrated and uncalibrated SLAM methods: NICER-SLAM (Zhu et al., 2024), DROID-SLAM (Teed & Deng, 2021), MASt3R-SLAM (Murai et al., 2025), GLUEMAP (Pan et al., 2026), TALO (Zhang et al., 2026a), VGGT-SLAM++ (Mandal et al., 2026), ViSTA-SLAM (Zhang et al., 2026b), SLAM-Former (Yuan et al., 2026), CUT3R (Wang et al., 2025b), VGGT-SLAM (Maggio et al., 2025) with Sim(3) and L(4), VGGSfM (Wang et al., 2024a), and VGGT-Ω (Wang et al., 2026a). Published baselines retain their reported protocols. Calibrated methods receive ground-truth camera intrinsics, whereas uncalibrated methods estimate them from the input frames. We compare the uncalibrated VGGT-Ω baseline with uncalibrated VGGT-Ω+SMoRe. **Metrics.** We report absolute trajectory error (ATE) RMSE in metres after similarity alignment between the estimated and ground-truth camera trajectories. This alignment removes global scale differences, enabling fair comparison across methods; lower is better.

Results. As shown in Fig. 3A, SMoRe reduces the average ATE RMSE of VGGT-Ω from 4.5 to 2.9 cm. Notably, this is lower than the best reported calibrated baseline using ground-truth camera intrinsics (3.1 cm) and the best reported uncalibrated baseline (3.8 cm). These results show that revisit-driven revision with SMoRe improves long-horizon trajectory consistency without fine-tuning the geometry foundation model. Fig. 3B visualizes the resulting 3D reconstruction for an example 7-Scenes sequence. We transform the predicted local point maps into the global coordinate system using the estimated camera poses and stitch them into a single reconstruction. Chaining VGGT-Ω predictions accumulates placement errors, resulting in visible misalignment around the cabinet and countertop. In contrast, SMoRe uses revisited observations to correct their global placement, producing a more spatially consistent reconstruction with finer geometric details. Fig. 3C shows the pairwise positional error between frames. SMoRe reduces the average pairwise error from 8.1 to 6.5 cm. The improvement is particularly evident for temporally distant frame pairs (red boxes), where chained VGGT-Ω exhibits larger errors than its SMoRe counterpart. This demonstrates that SMoRe better preserves geometric consistency between observations separated by long temporal intervals. Finally, Fig. 3D shows the positional error over time. VGGT-Ω+SMoRe generally maintains lower error than VGGT-Ω throughout the sequence, with particularly pronounced reductions following verified revisits (green lines). This illustrates how revisiting previously observed regions allows SMoRe to correct accumulated trajectory errors as the sequence progresses.

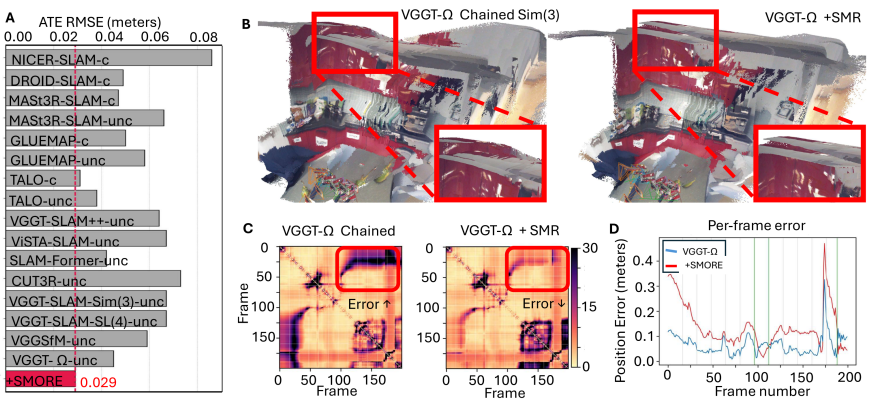


Figure 3: **Long-horizon SLAM on 7-Scenes. A.** ATE RMSE (metres) compared with calibrated (c) and uncalibrated (unc) SLAM methods; lower is better. The red bar and dotted line indicate the ATE RMSE of uncalibrated VGGT-Ω+SMoRE. **B.** Stitched 3D point-map reconstruction obtained using camera poses from chained VGGT-Ω versus VGGT-Ω+SMoRE on an example 7-Scenes video sequence. Red boxes highlight a revisited region where SMoRE improves spatial consistency and reconstruction detail. **C.** Pairwise positional errors between frames for VGGT-Ω and VGGT-Ω+SMoRE; red boxes highlight temporally distant frame pairs where SMoRE reduces accumulated error. **D.** Per-frame positional error (metres) over time. Green lines indicate verified revisits, after which SMoRE corrects accumulated trajectory error.

3.4 Novel View Synthesis

Problem setting and datasets. We evaluate novel view synthesis, where a small set of input views is used to render an object from unseen viewpoints. We use a held-out split of Objaverse-LVIS (Deitke et al., 2023) and all 1,033 Google Scanned Objects (GSO) (Downs et al., 2022). Each object is evaluated using four input views and ten target views rendered at 256×256 resolution. **Baselines.** We compare VGGT-Ω, VGGT, π^3, STream3R, Fast3R, and StreamVGGT (Wang et al., 2026a; 2025a; 2026b; Lan et al., 2026; Yang et al., 2025; Zhuo et al., 2026). For each geometry model, we train a lightweight Gaussian decoder using its original predicted geometry and then keep the decoder fixed (Appendix J). Because geometry models can produce different 3D predictions for the same frame under different input orderings, we repeatedly process the four input views under different orderings and geometrically aggregate consistent predictions. We analyze the effect of ordering variation in Appendix J. The resulting geometry is then supplied to the same fixed decoder for novel view synthesis. Thus, comparisons with and without SMoRE differ only in the estimated geometry used for rendering, while the rendering model remains unchanged. Details of repeated prediction aggregation are provided in Appendix H. **Metrics.** We report PSNR (↑) and SSIM (↑), and LPIPS (↓).

Results. As shown in Table 2, SMoRE improves novel view synthesis across all six geometry models and all three metrics on both test sets. On held-out Objaverse-LVIS, π^3 improves from 22.20 to 22.79 dB in PSNR and StreamVGGT from 21.21 to 21.62 dB. On GSO, VGGT improves from 20.71 to 20.95 dB and StreamVGGT from 18.30 to 18.59 dB. LPIPS also decreases for all six models on both datasets, indicating improved perceptual similarity. Qualitative examples are shown in Fig. 2B and Fig. S2. In Fig. 2B, for example, VGGT+SMoRE better preserves the geometry and appearance of the shoe tongue than VGGT alone. More generally, the improved geometry yields sharper object boundaries and better-preserved object structure in the synthesized views.

3.5 Analysis on design choices and computational requirements of SMoRE

We analyze key design choices of SMoRE on 7-Scenes SLAM (ATE RMSE, ↓) and CO3Dv2 camera pose estimation (AUC@30, ↑). We group the results by design factor in Table S6, with full results provided in Appendix K. From Table S6, we observe: **Memory scaffold (Group A).** Replacing our associative scaffold with a lookup table degrades performance, supporting associative memory as a more effective structure for memory storage, retrieval, and revision. **Memory sparsity (Group B).** Varying the sparsity of $\mathbf{h}$ reveals a trade-off between sparse coding and information preservation, with our default providing a good balance. **Visual descriptors (Group C).** We compare

Table 2: **Novel view synthesis on held-out Objaverse-LVIS (Deitke et al., 2023) and Google Scanned Objects (GSO) (Downs et al., 2022).** PSNR and SSIM are higher-is-better; LPIPS is lower-is-better. Values are mean ± standard deviation. Best results for each model are in bold.

	Objaverse-LVIS			GSO		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
VGGT-Ω Baseline	20.84±0.041	0.839±0.009	0.193±0.007	18.89±0.018	0.796±0.007	0.204±0.006
+SMoRE	**20.89±0.040**	**0.841±0.008**	**0.191±0.007**	**19.07±0.018**	**0.799±0.007**	**0.200±0.006**
VGGT Baseline	22.79±0.038	0.863±0.007	0.147±0.006	20.71±0.015	0.821±0.007	0.152±0.005
+SMoRE	**22.88±0.038**	0.863±0.008	**0.146±0.007**	**20.95±0.016**	**0.823±0.007**	**0.143±0.005**
π^3 Baseline	22.20±0.040	0.857±0.009	0.169±0.009	21.91±0.016	0.844±0.007	0.132±0.006
+SMoRE	**22.79±0.040**	**0.882±0.009**	**0.154±0.009**	**22.12±0.016**	**0.849±0.007**	**0.126±0.006**
STream3R Baseline	21.07±0.040	0.836±0.009	0.195±0.007	18.60±0.016	0.781±0.008	0.222±0.007
+SMoRE	**21.15±0.040**	0.836±0.009	**0.190±0.007**	**18.79±0.018**	**0.784±0.008**	**0.209±0.008**
Fast3R Baseline	16.99±0.047	0.834±0.009	0.276±0.011	14.07±0.020	0.784±0.008	0.322±0.008
+SMoRE	**17.17±0.047**	**0.842±0.010**	**0.271±0.012**	**14.20±0.020**	**0.789±0.008**	**0.311±0.008**
StreamVGGT Baseline	21.21±0.040	0.838±0.009	0.192±0.008	18.30±0.016	0.776±0.008	0.226±0.008
+SMoRE	**21.62±0.039**	**0.843±0.008**	**0.180±0.007**	**18.59±0.016**	**0.781±0.008**	**0.219±0.008**

DINOv2 (Oquab et al., 2024) with alternative visual descriptors (Appendix K.2). DINOv2 performs best, highlighting the importance of strong visual representations for reliable candidate retrieval. **Scaffold dimension (Group D).** Performance is stable across the tested dimensions, with smaller dimensions slightly degrading performance and larger ones providing limited additional gains. **Retrieval evidence (Group E).** Too little retrieval evidence degrades performance, while additional pairs provide diminishing gains, motivating our default setting. **Progressive placement revision (Group F).** Progressive correction in SMoRE outperforms uniformly distributing the correction along the trajectory, suggesting that larger corrections should be applied to more recent frames. **Window size and overlap (Group G).** Larger windows and greater overlap generally improve performance but increase computational and memory costs, motivating our default choice of 32-frame windows with 16 overlapping frames (50% overlap).

Computational requirements. We evaluate the GPU memory, CPU memory, RAM size, PFLOPs, and runtime of geometry foundation models with and without SMoRE (Appendix L.1). In Table S7, as sequence length N increases, SMoRE keeps the geometry model's active GPU working set largely bounded because inference operates on fixed-size windows and retrieves only a small number of historical frames when needed. In contrast, feed-forward baselines must maintain intermediate predictions to chain consecutive windows, while streaming models retain growing causal states or caches and can eventually exceed GPU memory limits. Although the external memory of SMoRE grows with the stored visual history, this growth occurs primarily outside the geometry model's active GPU context, allowing long sequences to be processed with substantially better memory scalability. Reducing peak GPU memory does not make SMoRE computationally free; instead, it trades a bounded GPU working set for additional memory management. Table S8 shows that computation in GPU remain dominated by the underlying geometry foundation model, while SMoRE adds modest CPU, PFLOP, and runtime overhead. In return, SMoRE enables substantially better memory scaling; for example, STream3R runs out of GPU memory at $N = 100$, whereas STream3R+SMoRE can still process the full sequence using bounded-window inference.

4 Conclusion and Future Works

We introduced SMoRE, a revisable external memory that extends frozen geometry foundation models beyond fixed-context prediction through storage, retrieval, geometric verification, and revision. Across eight geometry models, seven datasets, and five downstream tasks, our results show that SMoRE complements both feed-forward and streaming models, using verified revisits to correct accumulated errors over long visual sequences without fine-tuning the underlying geometry models. Several directions remain for future work. Retrieval verification introduces additional computation, motivating more efficient mechanisms for recalling reliable spatial information from the memory scaffold. Moreover, SMoRE currently focuses on revising global window placement; future work could extend revision to local geometric predictions when later observations provide stronger evidence.

AI USE STATEMENT

Generative AI tools were used solely to polish the manuscript's language and were not involved in the model designs or experiments.

REPRODUCIBILITY STATEMENT

All geometry foundation models remain frozen and are not fine-tuned for SMoRe. Code, models, and data will be made publicly available upon publication.

ETHICS STATEMENT

This work raises no ethical concerns beyond those generally associated with AI systems processing visual data, including privacy and responsible data use.

REFERENCES

Amar Ali-Bey, Brahim Chaib-Draa, and Philippe Giguere. Mixvpr: Feature mixing for visual place recognition. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2997–3006. IEEE, 2023.

Amar Ali-Bey, Brahim Chaib-Draa, and Philippe Giguere. Boq: A place is worth a bag of learnable queries. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17794–17803. IEEE, 2024.

Shun-ichi Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological cybernetics*, 27(2):77–87, 1977.

Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5297–5307, 2016.

Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking visual geo-localization for large-scale applications. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4868–4878. IEEE, 2022.

Gabriele Berton, Gabriele Trivigno, Barbara Caputo, and Carlo Masone. Eigenplaces: Training viewpoint robust models for visual place recognition. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11046–11056. IEEE, 2023.

Andrej Bicanski and Neil Burgess. Neuronal vector coding in spatial cognition. *Nature Reviews Neuroscience*, 21(9):453–470, 2020.

Yoram Burak and Ila R Fiete. Accurate path integration in continuous attractor network models of grid cells. *PLoS computational biology*, 5(2):e1000291, 2009.

Yifei Cao, Yu Liu, Guolong Wang, Zhu Liu, Kai Wang, Xianjie Zhang, Jizhe Yu, and Xun Tu. Eagle: Episodic appearance-and geometry-aware memory for unified 2d-3d visual query localization in egocentric vision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 2634–2642, 2026.

Sarthak Chandra, Sugandha Sharma, Rishidev Chaudhuri, and Ila Fiete. Episodic and associative memory from spatial scaffolds in the hippocampus. *Nature*, 638:739–751, 2025. doi: 10.1038/s41586-024-08392-y.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13142–13153, 2023.

Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. 2012.

Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3D scanned household items. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2553–2560, 2022.

Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proceedings of the IEEE international conference on computer vision*, pp. 873–881, 2015.

Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010. doi: 10.1109/MITS.2010.939925.

Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987. doi: 10.1364/JOSAA.4.000629.

Weiwei Hu, Qinglei Lin, Lihuan Shao, Jiaxu Lin, Keke Zhang, and Huibin Qin. A real-time map restoration algorithm based on orb-slam3. *Applied Sciences*, 12(15):7780, 2022.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35 (1):73–101, 1964. doi: 10.1214/aoms/1177703732.

Sergio Izquierdo and Javier Civera. Optimal transport aggregation for visual place recognition. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17658–17668. IEEE, 2024.

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

Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *2011 IEEE international conference on robotics and automation*, pp. 3607–3613. IEEE, 2011.

Yushi Lan, Yihang Luo, Fangzhou Hong, Shangchen Zhou, Honghua Chen, Zhaoyang Lyu, Shuai Yang, Bo Dai, Chen Change Loy, and Xingang Pan. STream3R: Scalable sequential 3D reconstruction with causal transformer. In *International Conference on Learning Representations (ICLR)*, 2026.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with MASt3R. In *European Conference on Computer Vision (ECCV)*, pp. 71–91, 2024.

Dominic Maggio and Luca Carlone. Vggt-slam 2.0: Real-time dense feed-forward scene reconstruction. *arXiv preprint arXiv:2601.19887*, 2026.

Dominic Maggio, Hyungtae Lim, and Luca Carlone. VGGT-SLAM: Dense RGB SLAM optimized on the SL(4) manifold. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

Avilasha Mandal, Rajesh Kumar, Sudarshan Sunil Harithas, and Chetan Arora. VGGT-SLAM++. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1008–1018, 2026.

Paul Merrell, Amir Akbarzadeh, Liang Wang, Philippos Mordohai, Jan-Michael Frahm, Ruigang Yang, David Nistér, and Marc Pollefeys. Real-time visibility-based fusion of depth maps. In *2007 IEEE 11th international conference on computer vision*, pp. 1–8. Ieee, 2007.

Riku Murai, Eric Dexheimer, and Andrew J. Davison. MASt3R-SLAM: Real-time dense SLAM with 3D reconstruction priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16695–16705, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Linfei Pan, Johannes Schönberger, and Marc Pollefeys. Global structure-from-motion meets feedforward reconstruction. *arXiv preprint arXiv:2605.26103*, 2026.

Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10881–10891. IEEE, 2021.

Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European conference on computer vision*, pp. 501–518. Springer, 2016.

Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2930–2937, 2013.

Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Scale drift-aware large scale monocular SLAM. In *Robotics: Science and Systems (RSS)*, 2010. doi: 10.15607/RSS.2010.VI.010.

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991. doi: 10.1109/34.88573.

Hengyi Wang and Lourdes Agapito. Amb3r: Accurate feed-forward metric-scale 3d reconstruction with backend. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14612–14625, 2026.

Jianyuan Wang, Nikita Karaev, Christian Rupprecht, and David Novotny. Vggsfm: Visual geometry grounded deep structure from motion. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21686–21697. IEEE, 2024a.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5294–5306, 2025a.

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

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10510–10522, 2025b.

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

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. In *International Conference on Learning Representations (ICLR)*, 2026b.

Thomas Whelan, Stefan Leutenegger, Renato F. Salas-Moreno, Ben Glocker, and Andrew J. Davison. ElasticFusion: Dense SLAM without a pose graph. In *Robotics: Science and Systems (RSS)*, 2015. doi: 10.15607/RSS.2015.XI.001.

Jianing Yang, Alexander Sax, Kevin J. Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3R: Towards 3d reconstruction of 1000+ images in one forward pass. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21924–21935, 2025.

Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025.

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12–22. IEEE, 2023.

Yijun Yuan, Zhuoguang Chen, Kenan Li, Weibang Wang, Minghui Qin, Zhijian Fang, Weicheng Zheng, and Hang Zhao. SLAM-Former: Putting SLAM into one transformer. In *European Conference on Computer Vision (ECCV)*, 2026.

Fengyi Zhang, Tianjun Zhang, Kasra Khosoussi, Zheng Zhang, Zi Huang, and Yadan Luo. Talo: Pushing 3d vision foundation models towards globally consistent online reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21870–21879, 2026a.

Ganlin Zhang, Shenhan Qian, Xi Wang, and Daniel Cremers. ViSTA-SLAM: Visual SLAM with symmetric two-view association. In *International Conference on 3D Vision (3DV)*, pp. 396–406, 2026b.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

Zihan Zhu, Songyou Peng, Viktor Larsson, Zhaopeng Cui, Martin R. Oswald, Andreas Geiger, and Marc Pollefeys. NICER-SLAM: Neural implicit scene encoding for RGB SLAM. In *International Conference on 3D Vision (3DV)*, pp. 42–52, 2024.

Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming visual geometry transformer. In *International Conference on Learning Representations (ICLR)*, 2026.

A Appendix Overview

The appendix expands the method and experiments in the same order in which they are needed to understand SMoRe. We first describe the coordinate systems used throughout the framework and how predictions from different windows are aligned with similarity transformations (Appendix B). We then explain how frames are assigned to source windows, which quantities remain in local coordinates, and how changes to a window's global placement affect its stored camera poses and point maps (Appendix C).

We next provide the full construction of the external memory. Appendix D and E describes how global camera poses are encoded into scaffold addresses, how frame descriptors are associated with these addresses, and how stored frames are retrieved from a current visual cue. Appendix F then expands the revisit-verification procedure: retrieved frames are processed jointly with the current window, and previously placed source windows provide independent geometric evidence for deciding whether a revisit should be accepted. Appendix G describes how accepted revisits modify window placements during processing and how the complete set of placements is refined after the sequence has been processed.

Appendix H describes the dense 3D content stored for each frame, how this content remains in its source window's local coordinate system, and how repeated 3D predictions are aligned and combined. We then give the full evaluation protocols. Appendix I specifies the datasets, frame sampling, subsets, and task-specific metrics used for camera pose estimation, multi-view depth estimation, 3D point-cloud reconstruction, and SLAM, while Appendix J provides the corresponding data, rendering, geometry-processing, and decoder-training details for novel view synthesis.

Finally, we provide the analyses supporting the design choices in the main paper. Appendix K contains the complete controlled ablations of retrieval, revisit verification, placement correction, scaffold configuration, window size, and repeated 3D predictions. Appendix L reports the additional runtime and memory costs introduced by SMoRe, and Appendix M collects the shared implementation and inference settings. We conclude with additional diagnostics that separate changes to the current window's local predictions from changes to its global placement (Appendix N) and examine how retrieval and storage behave as the content bank grows (Appendix O).

B How Camera-Pose Alignments Are Computed

The main text uses camera-pose alignment in two places. First, in Sec. 2 opening, consecutive windows are aligned using the camera poses of their shared frames to obtain their global placements T_k. Second, in Sec. 2.2, the newly predicted poses of a retrieved pair are aligned with the stored poses of the same frames in their source window to obtain the relative transformation $\widehat{Z}_{o,k}$. Both operations use the same similarity-alignment procedure described below.

Similarity transformation. Suppose the same frames have camera predictions in two coordinate systems, A and B. We seek the transformation that expresses the predictions from A in coordinate system B. We write this similarity transformation as

$$S = (s, R, t),$$

where $s > 0$ is scale, $R \in \mathrm{SO}(3)$ is rotation, and $t \in \mathbb{R}^3$ is translation. For a 3D point x,

$$S \cdot x = sRx + t.$$

For a camera pose (R_j, c_j), where R_j is its orientation and c_j its camera centre,

$$S \cdot (R_j, c_j) = (RR_j,\ sRc_j + t)\,.$$

Estimating the alignment. Because the two predictions correspond to the same frames, their camera centres provide known correspondences $c_j^A \leftrightarrow c_j^B$. We estimate the transformation from coordinate system A to B by

$$\widehat{S}_{A\to B} = \underset{s>0,\, R\in\mathrm{SO}(3),\, t}{\arg\min} \sum_j w_j \left\| sRc_j^A + t - c_j^B \right\|_2^2. \tag{1}$$

Thus, no additional frame matching or nearest-neighbour search is needed. Scale, rotation, and translation are estimated jointly from the corresponding camera centres using standard similarity alignment (Umeyama, 1991; Horn, 1987).

This is the operation used in Sec. 2: for two consecutive windows, the corresponding cameras are those of their shared frames. The resulting relative transformation places the new window with respect to the preceding window, and composing these transformations gives its global placement T_k.

The same operation is used in Sec. 2.2. For a retrieved pair from source window $\mathcal{W}_o$, the two cameras predicted in the joint pass are aligned with the stored poses of the same two frames in $\mathcal{W}_o$. This gives

$$\widehat{Z}_{o,k} = \widehat{S}_{\text{joint}\to o},$$

which relates the current joint prediction to the source window. The source window already has global placement T_o, so the corresponding proposal for the current window is

$$T_k^{(a)} = T_o \widehat{Z}_{o,k}.$$

Making the alignment robust. Camera predictions can contain outliers, so we refine the similarity fit for five reweighting rounds. After each round, the centre residual is

$$r_j = \left\| \widehat{S}_{A\to B} \cdot c_j^A - c_j^B \right\|_2,$$

and the weight used in the next round is

$$w_j = \min\left(1,\ \kappa \frac{\widetilde{r}}{r_j}\right), \qquad \kappa = 2.5,$$

where $\widetilde{r}$ is the median residual. A zero-residual correspondence keeps unit weight. After fitting the camera centres, we also compare the transformed camera orientation RR_j^A with R_j^B and reject correspondences with excessive rotational disagreement.

Additional scale check for consecutive windows. For consecutive-window alignment, an incorrect scale would affect every later window in the trajectory. We therefore compare the fitted scale s with an estimate s_{pair} obtained from the median ratio of pairwise camera-centre distances. We accept the alignment only if

$$\frac{s}{s_{\text{pair}}} \in \left[\frac{1}{\gamma_s}, \gamma_s\right], \qquad \gamma_s = 1.5.$$

This scale check is applied to the sequential window alignment described in Sec. 2.

Two-camera limitation. The shared-frame alignment between consecutive windows normally has multiple camera correspondences. A retrieved pair in Sec. 2.2, however, contains only two cameras. Two camera centres determine a baseline but do not by themselves uniquely constrain rotation about that baseline. The camera-orientation check provides additional consistency information, and SMoRe further requires two retrieved pairs from different source windows to produce consistent global-placement proposals before accepting a revisit.

C How Window Placements Affect Stored Geometry

In Sec. 2.1, each frame is stored once using its source window, and its camera pose and point map remain in that window's local coordinate system. Later, Sec. 2.2 changes only the global placement T_k of a window. Here we make explicit what this means for the stored geometry and for relative camera poses.

Source-window storage. Because consecutive windows overlap, the same frame may be predicted in more than one window. We store it only once, using the first window in which it appears. For frame I_i, this window is $\mathcal{W}_{\text{src}(i)}$.

Suppose I_i is stored in source window $\mathcal{W}_k$, with local camera pose (R_i, c_i) and local 3D points x_i. These quantities remain fixed in the coordinate system of $\mathcal{W}_k$. Their global values are obtained only when the window placement $T_k = (a_k, Q_k, \mathbf{b}_k)$ is applied:

$$R_i^{\text{global}} = Q_k R_i, \qquad c_i^{\text{global}} = a_k Q_k c_i + \mathbf{b}_k,$$

and

$$x_i^{\text{global}} = a_k Q_k x_i + \mathbf{b}_k.$$

Thus, the content bank stores local geometry, while T_k determines where that geometry currently lies in the global reconstruction.

What changes after a placement correction? When Sec. 2.2 replaces T_k by an updated global placement, the stored local camera poses and point maps are not recomputed. Instead, the new transformation is applied to the same local quantities. Consequently, all cameras and 3D points belonging to $\mathcal{W}_k$ move together.

This separation is useful because a revisit can correct where a window is placed without modifying the geometric relationships already predicted inside that window.

Relative poses within one source window. Consider two frames I_i and I_j stored in the same source window $\mathcal{W}_k$. After applying T_k, their global rotations satisfy

$$\left(R_j^{\text{global}}\right)^\top R_i^{\text{global}} = R_j^\top R_i.$$

Their relative translation, expressed in frame I_j, becomes

$$\left(R_j^{\text{global}}\right)^\top \left(c_i^{\text{global}} - c_j^{\text{global}}\right) = a_k R_j^\top (c_i - c_j).$$

Therefore, changing T_k leaves their relative rotation and translation direction unchanged. Only the translation magnitude can change through the window scale a_k.

This is why a placement correction primarily changes relationships *between* windows rather than the relative camera geometry already stored *within* one source window. It also explains why the pairwise-pose metric, which uses rotation and translation-direction errors, is unchanged for two frames whose local predictions remain fixed in the same source window.

When this property does not apply. The argument above assumes that the stored local predictions (R_i, c_i) themselves are unchanged. In Sec. 2.2, however, processing retrieved frames jointly with the current window can also produce new local camera poses and point maps. That is a separate effect from changing T_k. Appendix N isolates these two sources of change experimentally.

D How the Scaffold Stores and Retrieves Frames

Sections 2.1 and 2.2 introduce the two memory operations used by SMoRe. When a frame is stored, its estimated global camera pose is converted into a compact memory address. Later, the visual content of a current frame is used to predict an address and retrieve previously stored frames. Here we explain how the pose representation, memory address, and visual descriptor are connected, why the scaffold uses multiple spatial scales and a sparse address, and which parts of this associative pathway are actually used during trajectory revision.

Representing the global camera pose. When frame I_i is first stored, its source window has a current global placement $T_{\text{src}(i)}$. Applying this placement to the frame's local camera pose gives the global camera pose used to construct its scaffold representation. As described in Sec. 2.1, we represent camera position and orientation separately and concatenate them into a single scaffold state $\mathbf{g}_i$.

Camera position is first expressed in normalized scene units, with median scene depth equal to one. We use three spatial modules with periods

$$\lambda_1 = 2.4, \qquad \lambda_2 = 3.2, \qquad \lambda_3 = 4.0.$$

For module m, a position $\mathbf{p}_i = (x_i, y_i, z_i)$ is converted to periodic coordinates

$$\boldsymbol{\phi}_{m,i} = \frac{2\pi}{\lambda_m}(x_i, y_i, z_i) \bmod 2\pi.$$

Thus, each individual module is periodic: positions separated by λ_m along an encoded axis produce the same phase in that module.

A single periodic module cannot distinguish locations separated by its period. The three modules reduce this ambiguity because their periods are different: a displacement that wraps one module generally does not wrap the other two at the same location. We use

$$(2.4, 3.2, 4.0) = 0.8(3, 4, 5),$$

so the complete three-module pattern repeats only after

$$0.8 \ \mathrm{lcm}(3, 4, 5) = 48$$

normalized units along an encoded axis. The representation is therefore less ambiguous than any individual module, although it is still periodic rather than globally unique over an unbounded scene. These periods are fixed across the main experiments; we do not claim that they are an optimal set of spatial periods.

For each spatial module, the horizontal phase is represented on a 48×48 periodic grid and height on a 48-unit ring. These numbers specify the resolution with which a module represents its phase; they are distinct from the spatial period λ_m. We use 48×48 as the default horizontal resolution. The sensitivity study in Table S6 additionally evaluates 16×16 and 64×64 modules and shows that the method does not depend on using the finest tested resolution.

Representing camera orientation. Camera orientation is converted to yaw, pitch, and roll using the ZYX convention. Each angle is represented on a separate 256-unit periodic ring. A ring is natural for an angular variable because the beginning and end of the interval correspond to the same orientation after a full 360° rotation.

The three spatial modules and three orientation rings are concatenated, giving

$$\mathbf{g}_i \in \mathbb{R}^{N_g}, \qquad N_g = 3(48^2 + 48) + 3(256) = 7{,}824.$$

Thus, $\mathbf{g}_i$ is the scaffold representation of *where* frame I_i was estimated to have been captured when it entered memory.

From the scaffold state to a compact memory address. Directly associating visual descriptors with the full $7{,}824$-dimensional scaffold state would make the associative layer unnecessarily large. We therefore project the scaffold state into a smaller address space:

$$\mathbf{z}_i = W_{gh}\mathbf{g}_i, \qquad W_{gh} \in \mathbb{R}^{1024 \times 7824}.$$

The entries of W_{gh} are sampled independently from $\mathcal{N}(0, 1/7824)$ and the matrix remains fixed throughout inference.

We then retain only the largest 64 entries:

$$\mathbf{v}_i = \sigma_{64}(\mathbf{z}_i), \qquad \mathbf{h}_i = \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2},$$

where σ_{64} sets all but the 64 largest values to zero. The resulting

$$\mathbf{h}_i \in \mathbb{R}^{1024}$$

is the memory address of frame I_i.

The purpose of the top-k operation is to make different addresses selective: unrelated scaffold states activate only a small subset of the 1024 address units, reducing overlap between stored representations. The main configuration retains $k = 64$ units, i.e., 6.25% of the address layer. We use this as a fixed sparsity setting throughout the main experiments. The address dimension is examined in Table S6: increasing it from the default $N_h = 1024$ to 8192 does not consistently improve the final results, while reducing it to 512 produces only a modest change. Thus, the reported gains do not rely on an exceptionally large address layer.

Normalizing $\mathbf{h}_i$ also makes comparisons between addresses depend on their represented direction and overlap rather than on the absolute magnitude produced by the random projection.

The memory address does not contain the RGB frame or its dense geometry. Instead, $\mathbf{h}_i$ is linked to the corresponding content-bank entry. That entry stores the RGB frame, its source window, and its camera pose and point map in the local coordinate system of that source window. Appendix C explains how the current global placement $T_{\mathrm{src}(i)}$ is applied when this content is used. The address is assigned when the frame is first stored. Later placement corrections update $T_{\mathrm{src}(i)}$ but do not recompute $\mathbf{h}_i$. This separation allows the stored local geometry to move with its source window without rewriting the content bank or its memory index.

Recovering a scaffold state from an address. The mapping above provides the direction $\mathbf{g} \to \mathbf{h}$. The scaffold additionally contains a return mapping

$$W_{hg} \in \mathbb{R}^{7824\times 1024}$$

for the reverse direction. Given an address,

$$\widehat{\mathbf{g}}_i = \psi(W_{hg}\mathbf{h}_i),$$

where ψ normalizes each spatial or orientation field independently.

Unlike W_{gh}, which is a fixed random projection, W_{hg} is fitted once before inference. We sample valid scaffold states $\{\mathbf{g}_p\}$, construct their corresponding addresses $\{\mathbf{h}_p\}$ using the same W_{gh} and sparsification rule, and solve

$$W_{hg} = \arg\min_{W} \sum_{p} \|\mathbf{g}_p - W\mathbf{h}_p\|_2^2 + \lambda_{\mathrm{ridge}}\|W\|_F^2.$$

The sampled scaffold states use a maximum phase spacing of $\lambda_{\min}/6 = 0.4$. This fitting procedure uses no RGB frames, geometry-foundation-model outputs, or evaluation labels. It prepares the scaffold's return mapping once and is separate from the online memory updates described below.

The spatial part of the associative memory can therefore be written as

$$\mathbf{g}_i \underset{W_{hg}}{\overset{W_{gh}}{\rightleftharpoons}} \mathbf{h}_i.$$

Associating visual content with the same address. The scaffold tells us *where* a stored frame was captured, but at test time we begin with *what* the current frame looks like. Therefore, each frame I_i is independently encoded using the frozen DINOv2 ViT-S/14 encoder (Oquab et al., 2024). We use its normalized 384-dimensional CLS representation

$$\mathbf{s}_i \in \mathbb{R}^{384}$$

as the frame descriptor. DINOv2 is the default descriptor used in the main experiments; Table S6 compares it with several alternative place-recognition descriptors.

We maintain two mappings between the visual descriptor and the memory address:

$$W_{sh} \in \mathbb{R}^{1024\times 384}, \qquad W_{hs} \in \mathbb{R}^{384\times 1024},$$

such that

$$W_{sh}\mathbf{s}_i \approx \mathbf{h}_i, \qquad W_{hs}\mathbf{h}_i \approx \mathbf{s}_i.$$

The first converts appearance into an address, while the second provides the reverse address-to-appearance association.

Why are these mappings learned online? Unlike the pose-to-address mapping, the association between a particular visual observation and its address is not known before that frame is encountered. New pairs $(\mathbf{s}_i, \mathbf{h}_i)$ therefore arrive throughout the sequence. We update both visual associations incrementally using recursive least squares (RLS), so that new observations can be added without retraining either the DINOv2 encoder or the geometry foundation model.

For a generic association $W\mathbf{x}_i \approx \mathbf{y}_i$, the RLS update is

$$\begin{aligned} \mathbf{k}_i &= \frac{P_{i-1}\mathbf{x}_i}{1 + \mathbf{x}_i^\top P_{i-1}\mathbf{x}_i}, \\ W^{(i)} &= W^{(i-1)} + \left(\mathbf{y}_i - W^{(i-1)}\mathbf{x}_i\right)\mathbf{k}_i^\top, \\ P_i &= P_{i-1} - \mathbf{k}_i\mathbf{x}_i^\top P_{i-1}, \end{aligned} \tag{2}$$

where P_i is the RLS covariance matrix and $\mathbf{k}_i$ determines the contribution of the new association.

For W_{sh} we set

$$\mathbf{x}_i = \mathbf{s}_i, \qquad \mathbf{y}_i = \mathbf{h}_i,$$

while for W_{hs} the two roles are exchanged. These updates modify only the external memory.

The complete associative pathway. The four mappings together form

$$\boxed{\mathbf{g}_i \underset{W_{hg}}{\overset{W_{gh}}{\rightleftarrows}} \mathbf{h}_i \underset{W_{hs}}{\overset{W_{sh}}{\rightleftarrows}} \mathbf{s}_i}$$

and connect the spatial and visual information associated with a stored frame. Here, $\mathbf{g}_i$ represents its global spatial state, $\mathbf{s}_i$ represents its visual content, and $\mathbf{h}_i$ is the shared sparse representation through which the two are associated.

The four matrices are obtained in different ways: W_{gh} is sampled once and remains fixed; W_{hg} is fitted once from sampled scaffold states; and W_{sh} and W_{hs} are updated online as frames are stored. None of these operations updates the geometry foundation model.

Retrieving a stored frame from a current observation. For a current frame I_q, we compute its descriptor $\mathbf{s}_q$ and use only the appearance-to-address direction of the associative memory:

$$\widetilde{\mathbf{h}}_q = \frac{W_{sh}\mathbf{s}_q}{\|W_{sh}\mathbf{s}_q\|_2 + 10^{-12}}.$$

We then compare this predicted address with every stored address,

$$\mathrm{score}(q, j) = \widetilde{\mathbf{h}}_q^\top \mathbf{h}_j.$$

A larger score means that the appearance of the current frame predicts an address closer to the address stored for I_j.

We retain at most the five highest-scoring candidates and apply the eligibility conditions from Sec. 2.2:

$$j < r_k - \Delta, \qquad \mathrm{src}(j) \neq k - 1, \qquad \mathbf{s}_q^\top \mathbf{s}_j \geq \tau_{\mathrm{app}}. \tag{3}$$

The first condition prevents a temporally nearby frame from being treated as a long-range revisit. The second removes the immediately preceding source window, which is already connected to the current window through their overlap. The third requires the retrieved frame to remain visually compatible with the query. The surviving frames are only candidates; their geometric consistency is checked in Sec. 2.2 and expanded in Appendix F.

Candidate retrieval follows

$$\mathbf{s}_q \xrightarrow{W_{sh}} \widetilde{\mathbf{h}}_q \longrightarrow \mathbf{h}_j \longrightarrow I_j, \mathrm{src}(j).$$

In particular, we do not decode $\widetilde{\mathbf{h}}_q$ through W_{hg} and use the resulting state as the current frame's global pose. Once a frame is retrieved, its source window identifies its current global placement $T_{\mathrm{src}(j)}$, while new camera predictions from the geometry foundation model provide the relative transformation used to verify the revisit.

E Scaffold Dynamics and Controlled Memory Diagnostics

The previous section described the scaffold as the mapping $\mathbf{g} \leftrightarrow \mathbf{h} \leftrightarrow \mathbf{s}$ used to store and retrieve frames. Here we briefly describe the continuous attractor structure underlying $\mathbf{g}$ and report controlled tests of its behavior. These tests characterize the memory representation itself; they are separate from the end-to-end trajectory experiments.

Continuous attractor fields. Each spatial or orientation component of the scaffold is represented by a periodic neural field, following continuous-attractor models of spatial and directional coding (Burak & Fiete, 2009; Bicanski & Burgess, 2020). The field maintains a localized region of high activation,

which we call an *activity bump*. The position of this bump around a ring or torus represents the encoded coordinate.

For example, an orientation ring represents one angular variable such as yaw. Moving the bump around the ring changes the represented angle, while a full revolution returns to the same state. The spatial modules use the same principle on periodic two-dimensional grids and height rings. Thus, the scaffold stores pose through the *location* of the activity pattern rather than through its overall magnitude.

Maintaining and moving the represented pose. Local excitation together with broader inhibition allows an activity bump to remain stable once formed (Amari, 1977). A motion input can translate the bump across the periodic field, so changes in camera position or orientation correspond to changes in its phase. Observed camera poses provide anchors for the represented state; the scaffold therefore does not rely only on accumulating motion over time.

The main SMoRE pipeline does not use these dynamics as a replacement for the geometry foundation model's camera predictions. Instead, the predicted global pose provides the state that is encoded when a frame is stored, as described in Sec. 2.1.

Returning from an address to a valid scaffold state. Appendix D introduced the return mapping W_{hg} from the sparse address $\mathbf{h}$ to the scaffold state $\mathbf{g}$. For a state $\mathbf{g}$, one complete scaffold return is

$$\Phi(\mathbf{g}) = \psi(W_{hg}\,\sigma_{64}(W_{gh}\mathbf{g}))\,,$$

where ψ normalizes the individual spatial and orientation fields.

This return mapping is useful because an associative cue need not reproduce every component of the original scaffold state exactly. The return operation encourages the reconstructed representation to remain consistent with the learned scaffold structure. A perturbation that makes different modules mutually inconsistent can therefore be reduced by the return mapping. By contrast, a coherent displacement that moves all modules to another valid pose is itself a valid scaffold state; determining whether such a state corresponds to the correct physical location requires the geometric verification described in Sec. 2.2.

What these diagnostics establish. Fig. S1 checks three properties required by the memory design: a pose representation can remain stable, its phase can move continuously with motion, and a stored scaffold state can be recalled through the associative representation over the tested memory load. These experiments support the internal behavior of the scaffold, while the usefulness of scaffold retrieval for SMoRE is evaluated separately by the end-to-end comparison with flat descriptor retrieval in Table S6.

F How Retrieved Frames Verify a Revisit

Section 2.2 describes how SMoRE verifies the frames returned by the scaffold before changing any window placement. Here we expand that procedure. The key idea is that a retrieved frame is not trusted from appearance alone: it must provide a geometric connection between the current window and a previously placed source window, and a second source window must support a compatible placement.

Forming a retrieved pair. For each retrieved frame I_j, we select a nearby stored frame I_{j-3} or I_{j+3}. The partner must belong to the same source window as I_j and satisfy the sequential eligibility rule from Sec. 2.2. If neither partner frame is valid, the retrieved frame is discarded.

The two frames form a *retrieved pair*. A pair is used instead of a single frame because two camera poses provide a geometric baseline for aligning the new prediction with the stored source window. We use an offset of three frames as a fixed design choice so that the second frame is nearby but distinct.

Why use two different source windows? We retain two retrieved pairs, denoted a and b. The two frames within each pair come from the same source window, but the two pairs must come

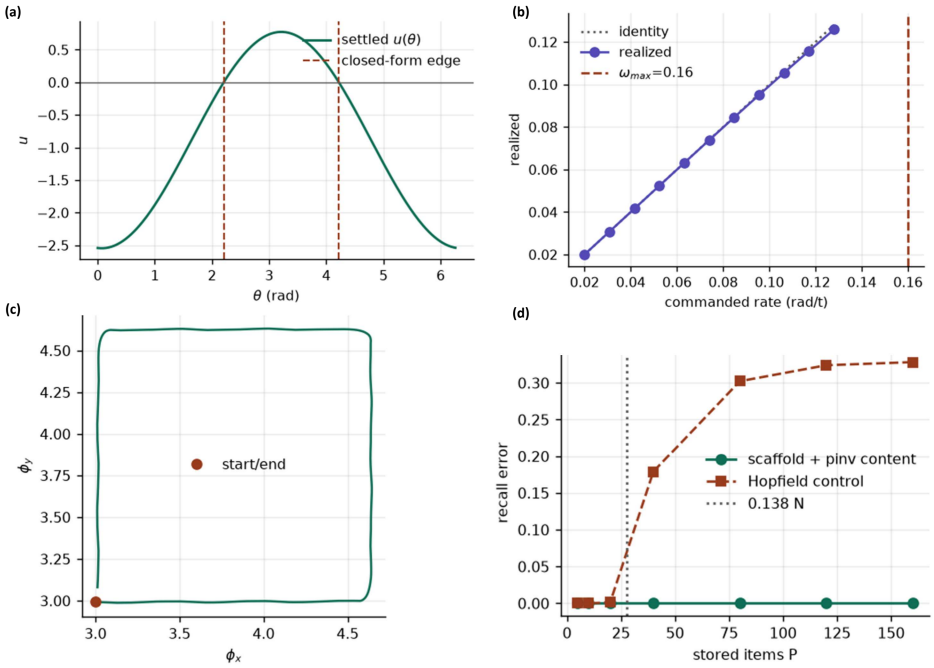


Figure S1: **Controlled diagnostics of the scaffold representation.** **(a)** A stationary orientation ring maintains a localized activity bump; the bump centre represents the encoded angle, while the orange dashed lines show the predicted bump boundaries. **(b)** When angular motion is supplied, the measured bump velocity closely follows the commanded velocity over the tested range. **(c)** The spatial phases during a closed two-dimensional path return close to their starting point, illustrating path integration in the periodic scaffold coordinates. **(d)** Associative recall error as the number of stored states increases. The scaffold configuration with the fitted content return map maintains low error through the largest tested load ($P = 160$), whereas the Hopfield-style control accumulates larger error. These are controlled memory diagnostics and do not measure end-to-end reconstruction accuracy or establish an unrestricted memory-capacity limit.

from different source windows. Because every stored frame records its source window, we enforce$\mathrm{src}(j_a) \neq \mathrm{src}(j_b)$.

This distinction is important. Each source window has its own global placement T. If both pairs came from the same source window, both placement proposals would depend on the same historical reference. Using two different source windows instead provides two separately placed references with which to check the current window.

We use two pairs by default. One pair provides no second historical placement with which to test agreement, while a third pair requires two additional frames. Table S6 compares one, two, and three pairs and shows that two retain most of the measured benefit of three with fewer additional inputs.

Joint prediction with the current window. If two valid pairs are available, the frozen geometry foundation model processes the 32 current-window frames together with the four retrieved frames, giving 36 input frames in total. The 32 current frames retain their sequential order, and the four retrieved frames are appended as the two pairs.

Because all 36 frames are processed together, their camera poses are predicted in one common local coordinate system. This joint prediction creates a direct geometric relationship between the current window and the retrieved historical frames; the existing chain of window placements is not used to create that relationship.

Obtaining a global-placement proposal. Consider retrieved pair a, whose source window is $\mathcal{W}_{o_a}$. The two frames in this pair have camera poses from the new joint prediction and stored camera poses in the local coordinate system of $\mathcal{W}_{o_a}$.

Using the similarity-alignment procedure from Appendix B, we align these two sets of camera poses and obtain

$$\widehat{Z}_{o_a,k},$$

the measured transformation relating the current joint prediction to source window $\mathcal{W}_{o_a}$.

Since $\mathcal{W}_{o_a}$ already has the global placement T_{o_a}, pair a proposes the current window placement

$$T_k^{(a)} = T_{o_a} \widehat{Z}_{o_a,k}.$$

Repeating exactly the same procedure for pair b gives

$$T_k^{(b)} = T_{o_b} \widehat{Z}_{o_b,k}.$$

Thus, the two retrieved pairs independently of the retrieval score provide two geometric proposals for where the current window should lie in the global reconstruction. The scaffold is used to find the historical frames; the geometry foundation model and their stored source-window placements determine these proposals.

Checking whether the two proposals agree. We accept the revisit only if $T_k^{(a)}$ and $T_k^{(b)}$ place the current window sufficiently close to one another.

Let E_k denote the median distance of the current-window camera positions from their mean position after global placement. We use this quantity to scale the translation tolerance to the size of the current reconstruction. The two proposals must satisfy

$$\angle\Big(R(T_k^{(a)}), R(T_k^{(b)})\Big) \le 3^\circ, \qquad \Big\|c(T_k^{(a)}) - c(T_k^{(b)})\Big\|_2 \le 0.15 E_k. \tag{4}$$

The first condition checks rotational agreement, while the second checks translation relative to the scale of the scene.

If either condition fails, the retrieved frames are rejected and no retrieval-based placement correction is applied. If both conditions pass, the revisit is accepted and the resulting geometric constraint is used to revise the window placements, as described in Appendix G.

What verification does and does not guarantee. The two source windows provide two separate geometric checks, but this does not guarantee that every accepted revisit is correct. Repeated scene structure, inaccurate historical placements, or weak two-camera geometry can still produce compatible but incorrect proposals. The agreement test therefore acts as a practical consistency check rather than a proof of correspondence.

Jointly processing the retrieved frames can also change the current window's local camera poses and point maps before the agreement decision is made. Rejecting a revisit prevents the global placement correction, but does not automatically restore the prediction obtained from the current window alone. We examine this separate effect in Appendix N.

G How Accepted Revisits Revise the Trajectory

Section 2.2 describes the online correction applied after a revisit has been verified, while we also do the final refinement after the full sequence has been processed. Both operations modify the global window placements T_k; the camera poses and point maps stored inside each source window remain in their local coordinate systems, as explained in Appendix C.

From a verified revisit to a placement correction. Before a revisit, the current window $\mathcal{W}_k$ has global placement T_k. The two retrieved pairs first produce the proposals $T_k^{(a)}$ and $T_k^{(b)}$ described in Appendix F. These proposals are used only for the agreement test in Eq. 4.

If the two proposals agree, we combine the frames from both retrieved pairs and fit one similarity transformation using the same robust alignment as Eq. 1. Specifically,

$$T_k^{\text{corr}} = \underset{S\in\text{Sim}(3)}{\arg\min} \left[\sum_{j\in a} w_j \left\| S\cdot c_j^{\mathcal{A}} - T_{o_a}\cdot c_j^{o_a} \right\|_2^2 + \sum_{j\in b} w_j \left\| S\cdot c_j^{\mathcal{A}} - T_{o_b}\cdot c_j^{o_b} \right\|_2^2 \right]. \tag{5}$$

Here, $S \in \text{Sim}(3)$ is a candidate similarity transformation, comprising scale, rotation, and translation, that maps the joint-pass coordinate system into the global coordinate system. The weight w_j is the robust weight assigned to retrieved frame j by the alignment procedure in Eq. 1, which downweights frames whose camera poses are inconsistent with the fitted transformation. Furthermore, $c_j^{\mathcal{A}}$ is the camera centre of retrieved frame j predicted in the joint pass $\mathcal{A}$, while $c_j^{o_a}$ or $c_j^{o_b}$ is its stored camera centre in source window $\mathcal{W}_{o_a}$ or $\mathcal{W}_{o_b}$, respectively. Applying T_{o_a} or T_{o_b} places the corresponding stored camera centre in the global coordinate system. Thus, T_k^{corr} is the single robust placement that jointly aligns the predictions of both retrieved pairs with their stored global poses; it is not an average of $T_k^{(a)}$ and $T_k^{(b)}$.

The transformation required to move the current placement to this corrected placement is

$$C = T_k^{\text{corr}} T_k^{-1}. \tag{6}$$

Thus, C represents the accumulated placement error revealed by the revisit.

Why distribute the correction across earlier windows? The error revealed at window k was generally accumulated over several previous window alignments. Applying the full correction only to $\mathcal{W}_k$ would therefore create a discontinuous jump between neighboring windows. Instead, SMoRe distributes the correction over the windows between the earlier reference window $\mathcal{W}_o$ and the current window.

For a window $\mathcal{W}_\ell$ with $o < \ell \le k$, we define

$$\alpha_\ell = \frac{\ell - o}{k - o}.$$

The reference window has correction weight 0, while the current window has weight 1. Intermediate windows receive progressively larger fractions of the correction.

Applying a fraction of a similarity transformation. A similarity transformation contains rotation, translation, and scale, so it cannot be scaled directly by multiplying its transformation matrix by α_ℓ. We instead use the logarithm and exponential maps:

$$T_\ell \leftarrow \text{Exp}(\alpha_\ell \,\text{Log}\, C)\, T_\ell, \qquad o < \ell \le k. \tag{7}$$

Here,

$$\text{Log}\, C = (\boldsymbol{\omega}, \boldsymbol{\nu}, \sigma) \in \mathbb{R}^7$$

represents the correction using three rotation values, three translation-related values, and one log-scale value. Multiplying this vector by α_ℓ selects the desired fraction of the correction, and Exp converts it back into a valid similarity transformation.

Consequently,

$$\alpha_\ell = 0 \quad \Rightarrow \quad \text{no correction,}$$

while

$$\alpha_\ell = 1 \quad \Rightarrow \quad \text{the full correction } C.$$

The update therefore changes the trajectory smoothly between the historical reference and the current window rather than inserting a single abrupt correction.

What changes after the update? Only the global placements T_ℓ are modified. The camera poses and point maps stored for frames in $\mathcal{W}_\ell$ remain unchanged in their source-window coordinates. Applying the updated T_ℓ therefore moves the cameras and their associated 3D geometry together. This is the same separation between local geometry and global placement described in Appendix C.

The online update is immediately available to later windows. Therefore, a correction made at one revisit can influence the global placements used when subsequent frames are stored and when later revisits are verified.

Recording constraints between windows. In addition to updating the running trajectory, SMoRe retains the relative geometric measurements obtained while processing the sequence. There are two types of connections.

A *sequential connection* is obtained from the shared frames of two neighboring windows. A *revisit connection* is obtained from an accepted historical retrieval. For a connection $e = (\ell, k)$, let Z_e denote the measured relative transformation between windows $\mathcal{W}_\ell$ and $\mathcal{W}_k$.

The current global placements themselves imply the relative transformation

$$T_\ell^{-1} T_k.$$

If the reconstruction is perfectly consistent with the measurement Z_e, these two relative transformations should agree.

Final global refinement. Online corrections are applied one revisit at a time. After all windows have been processed, SMoRe jointly refines the complete set of window placements using all retained sequential and accepted revisit connections.

We fix T_1 to remove the global gauge freedom and optimize

$$\min_{T_2,\dots,T_K} \sum_{e=(\ell,k)\in\mathcal{E}} \rho_H\Big(\big\| \mathrm{Log}\left(Z_e^{-1} T_\ell^{-1} T_k \right) \big\|_{\Lambda_e}^2 \Big), \tag{8}$$

where $\mathcal{E}$ is the set of retained connections. The quantity

$$Z_e^{-1} T_\ell^{-1} T_k$$

measures the disagreement between the relative placement predicted by the current trajectory and the relative transformation measured from the frames. Taking Log converts this disagreement into rotation, translation, and scale components that can be optimized together.

The diagonal matrix Λ_e controls the relative contribution of these components, while the Huber loss ρ_H (Huber, 1964) reduces the influence of unusually large residuals. This is a pose-graph optimization over window placements rather than bundle adjustment over individual 3D points (Grisetti et al., 2010; Kümmerle et al., 2011; Dellaert, 2012).

Handling inconsistent revisit constraints. An accepted revisit can still be incorrect, for example, because of repeated scene structure. We therefore perform the global solve robustly. After the first optimization, revisit connections with large residuals are rejected according to the configured residual criterion, and the remaining graph is optimized once more. Sequential connections are retained because they define the basic chain of the reconstruction.

Online revision versus final refinement. The two stages serve different purposes. The distributed update in Eq. 7 corrects the trajectory while the sequence is still being processed, so later observations benefit from the revised placements. The optimization in Eq. 8 instead considers all measurements together after the sequence is complete and removes remaining global inconsistencies. Both stages update only the window placements.

H How Repeated 3D Predictions Are Combined

The repeated geometry predictions in SMoRe is different from the trajectory revision described in Appendix G. Trajectory revision changes the global placement T_k of a window after a verified revisit. Here, instead, we keep the same input frames and combine several predictions of their 3D geometry. This operation is used for novel view synthesis and the corresponding controlled ablations; it is separate from the long-horizon Replica and ScanNet++ reconstruction experiments, which use revisit-driven trajectory revision.

Why make repeated predictions? Some geometry foundation models produce slightly different camera poses and point maps when the same frames are presented in a different order. Rather than selecting one ordering, SMoRe treats these predictions as repeated estimates of the same underlying geometry and combines those that agree.

An *ordering* is simply a permutation of the same input frames. For example,

$$(I_1, I_2, I_3, I_4) \qquad \text{and} \qquad (I_3, I_1, I_4, I_2)$$

contain exactly the same four frames but place them at different input positions. Each ordering corresponds to one separate invocation of the frozen geometry foundation model; it does not introduce any additional observations. After inference, every prediction is associated back with its original frame. The number of orderings is an inference-time choice. In the main novel view synthesis experiments we use four orderings. The effect of changing this number is studied in Table S4.

Aligning predictions from different orderings. Different passes can predict their geometry in slightly different local coordinate systems and scales. Before comparing their point maps, we therefore align the repeated predictions into a common coordinate system using the same frame identities across passes. Their relative scales are re-measured symmetrically so that no single ordering is treated as the privileged reference. After this alignment, a fixed frame and pixel refer to the same physical quantity across all orderings. We can therefore compare their predicted 3D points directly.

Combining the per-pixel predictions. Let

$$\mathbf{x}^{(r)} \in \mathbb{R}^3, \qquad r \in \{1, \dots, K\},$$

denote the aligned 3D prediction for one pixel under ordering r. We first compute the coordinate-wise median

$$\widetilde{\mathbf{x}} = \mathrm{med}_{\mathrm{coord}} \left\{ \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(K)} \right\},$$

where the median is taken separately along the x, y, and z coordinates.

We then measure how far each prediction lies from this initial estimate:

$$d_r = \left\| \mathbf{x}^{(r)} - \widetilde{\mathbf{x}} \right\|_2, \qquad \mathcal{I} = \{ r : d_r \le \epsilon \}.$$

The set $\mathcal{I}$ therefore contains the orderings whose 3D predictions agree with the median within tolerance ϵ.

If at least two predictions agree, we retain their coordinate-wise median:

$$\widehat{\mathbf{x}} = \mathrm{med}_{\mathrm{coord}} \left\{ \mathbf{x}^{(r)} : r \in \mathcal{I} \right\}, \qquad |\mathcal{I}| \ge 2.$$

Using agreement before fusion prevents one strongly different prediction from directly determining the final point.

What happens when fewer than two predictions agree? Discarding every uncertain pixel would create unnecessary holes in the reconstruction. We therefore use the median of all valid predictions as a fallback when fewer than two predictions satisfy the agreement test.

For novel view synthesis, objects are represented in a normalized coordinate system, so we instead use the fixed tolerance

$$\epsilon = 0.01$$

and do not apply the $0.10z$ omission rule. If only one valid prediction is available and it is not contradicted by another valid prediction, that point is retained. When confidence predictions are available, their per-pixel median is stored separately from the fused 3D position.

Why use a median? The purpose of the fusion is to retain geometry that is supported across repeated predictions while reducing the effect of an occasional order-dependent outlier. A coordinate-wise median provides a simple robust estimate for this purpose. It is computed only after all repeated predictions have been aligned into the same coordinate system.

This procedure is related to consistency-based depth-map fusion (Merrell et al., 2007; Galliani et al., 2015; Schönberger et al., 2016), but the source of repeated evidence is different: here we compare multiple predictions of the *same frame and pixel* obtained from different input orderings, rather than reprojecting predictions from different camera views.

Table S1: **Datasets and evaluation protocols.** The table summarizes the data used for each downstream task. Detailed sampling and scoring procedures are given below.

Task	Dataset	Evaluation subset
Camera pose	CO3Dv2 (Reizenstein et al., 2021)	37 validated trajectories; $N \in \{10, 50, 100, 200\}$ sampled frames
Camera pose	RE10K (Zhou et al., 2018)	1,653 clips at $N = 10$; 100 clips for the $N \in \{50, 100, 200\}$ length study
Multi-view depth	Replica (Straub et al., 2019)	18 scenes; $N = 200$ frames per scene
3D point-cloud reconstruction	ScanNet++ (Yeshwanth et al., 2023)	50 scenes; $N = 200$ frames per scene
SLAM	7-Scenes (Shotton et al., 2013)	All seven indoor scenes
Novel view synthesis	Objaverse-LVIS (Deitke et al., 2023)	$\sim$19K training objects and a held-out evaluation split
Novel view synthesis	GSO (Downs et al., 2022)	All 1,033 objects; no GSO object is used for training

How the resulting geometry is stored. For a model that directly predicts point maps, the fused 3D points are used directly. If a model instead predicts depth D_i, a valid pixel u is first unprojected using the predicted camera intrinsics K_i, and the corresponding camera pose places the point in the local coordinate system of its source window:

$$\mathbf{x}_i = g_i^{(\mathrm{src}(i))} \cdot \pi^{-1}(u; D_i(u), K_i)\,,$$

where π^{-1} denotes depth-based camera unprojection.

As in Appendix C, these points remain in the local coordinate system of their source window. Their global positions are obtained only when the current window placement is applied:

$$\mathbf{x}_i^{\mathrm{global}} = T_{\mathrm{src}(i)} \cdot \mathbf{x}_i.$$

Consequently, a later trajectory correction can move an entire stored surface by changing $T_{\mathrm{src}(i)}$ without modifying its local point coordinates.

Relation to trajectory revision. The repeated-prediction operation and revisit-driven trajectory revision address different sources of error. Revisit verification changes *where* a window is placed in the global reconstruction, whereas repeated prediction changes the estimate of the *local 3D geometry* obtained from a fixed set of frames.

Combining repeated 3D predictions. We separately vary how many different frame orderings are combined. For VGGT on GSO, four orderings improve PSNR from 20.33 to 20.60 dB, while eight show a slight improvement to 21.40 dB (Table S4). However, additional orderings do not benefit every geometry foundation model, supporting the fixed four-order budget used in the main experiments. The combination rule also matters: discarding a point when fewer than two predictions agree reduces PSNR to 18.41 dB, compared with 20.60 dB when the median fallback is retained.

I Datasets and Evaluation Protocols

The main experiments in Sec. 3 evaluate SMoRe on five downstream tasks using seven datasets. Here we provide the exact evaluation subsets, frame-selection rules, and scoring procedures used for each task. Within every paired Baseline/+SMoRe comparison, the same geometry foundation model and the same input frames are used. When computational constraints require a reduced subset for a particular model, its Baseline and +SMoRe results are evaluated on the same subset.

I.1 Camera Pose Estimation

The camera-pose experiments in Sec. 3.1 evaluate how pose accuracy changes as more frames from the same video trajectory are processed. We use CO3Dv2 and RealEstate10K because they provide different camera-motion patterns: CO3Dv2 contains object-centric trajectories with substantial viewpoint changes, whereas RE10K is dominated by forward camera motion and therefore contains fewer natural revisits.

CO3Dv2. We use 37 CO3Dv2 sequences with validated camera trajectories. The videos span approximately 8–68 s, with a median duration of 26 s. For each trajectory, we evaluate

$$N \in \{10, 50, 100, 200\}$$

input frames. Frames are sampled uniformly across the available trajectory rather than taking the first N frames. Thus, increasing N increases the temporal density with which the same underlying trajectory is observed.

The controlled pose ablations in Appendix K use a separate subset of twelve CO3Dv2 trajectories at $N = 200$ and should not be interpreted as the full 37-sequence evaluation.

RealEstate10K. RE10K contains real-estate videos with predominantly forward camera motion (Zhou et al., 2018). The 100 clips used for the length study span 6.7–9.3 s, with a median duration of 9.3 s. We uniformly sample $N \in \{50, 100, 200\}$ frames from these clips. For the $N = 10$ evaluation, we use the larger set of 1,653 usable test clips.

The $N = 10$ and longer-sequence evaluations therefore use different numbers of clips and should not be interpreted as a nested set of the same sequences.

Pose metric. Following the relative-pose evaluation used in the main experiments, we evaluate every pair of sampled frames. For a pair (i, j), we compute its rotation error and translation-direction error and define the pairwise pose error as

$$e_{ij} = \max\left(e_{ij}^{\text{rot}}, e_{ij}^{\text{trans}}\right).$$

Translation magnitude is not scored because monocular geometry is defined only up to scale.

AUC@30 is the normalized area under the cumulative distribution of e_{ij} between 0° and 30°. Higher AUC therefore means that a larger fraction of frame pairs have accurate relative camera poses. Table S2 reports the mean and standard deviation over the evaluated sequences or clips.

I.2 Multi-view Depth Estimation and 3D Point-Cloud Reconstruction

The experiments in Sec. 3.2 evaluate multi-view depth estimation on Replica (Straub et al., 2019) and 3D point-cloud reconstruction on ScanNet++ (Yeshwanth et al., 2023). Both are evaluated in the same long-horizon setting using 32-frame windows with 16-frame overlap, as described in Sec. 2. Each window is first reconstructed locally by the frozen geometry foundation model and then placed in the shared global coordinate system using the overlapping frames. Baseline and +SMoRe use exactly the same input frames, windowing strategy, and geometry foundation model. Baseline connects the windows sequentially without long-range memory, whereas +SMoRe additionally retrieves earlier observations, verifies candidate revisits, and uses accepted geometric evidence to revise the global placements of the reconstructed windows.

Replica. Replica (Straub et al., 2019) provides photorealistic synthetic indoor environments together with accurate camera information and dense reference 3D geometry. It contains 18 such scenes with about 2000 frames per scene. Its controlled geometry allows us to measure how reconstruction errors accumulate as a scene is processed across many overlapping windows. For each evaluated trajectory, Baseline and +SMoRe process the same frames in the same temporal order; the only difference is whether the external memory is used to retrieve and revise previously placed reconstruction windows.

ScanNet++. ScanNet++ (Yeshwanth et al., 2023) contains high-resolution observations of real indoor environments together with high-quality reference 3D geometry. It contains 1006 scenes with about 8000 frames per scene. We only use a subset of 50 scenes with the corresponding first 200 frames. Compared with Replica, it evaluates the same long-horizon reconstruction procedure under real image appearance, camera motion, and more complex scene geometry. We again use identical input frames and 32-frame windows for Baseline and +SMoRe, so improvements measure the effect of memory-based revisit verification and trajectory revision rather than differences in the input data or geometry foundation model.

Table S2: **Camera pose estimation accuracy on RE10K (Zhou et al., 2018) and CO3Dv2 (Reizenstein et al., 2021).** We report AUC@30↑ for increasing numbers of input frames N. Values are mean ± standard deviation across evaluated sequences (CO3Dv2) or clips (RE10K). For each geometry foundation model, Baseline denotes its original inference procedure and +SMoRe denotes the same model equipped with our framework. For StreamVGGT and STream3R, Baseline uses native causal inference, whereas +SMoRe uses our windowed framework. OOM indicates that inference exceeded the tested 46 GB GPU memory budget. Within each model and sequence length, the higher mean is shown in bold; equal means are left unbolded.

	Model	Method	RE10K (N)				CO3Dv2 (N)			
			10	50	100	200	10	50	100	200
Feed-forward	DUSt3R	Baseline	65.6±2.8	88.0±0.7	**85.9±0.7**	82.6±0.9	88.9±1.3	88.8±0.7	88.0±0.6	83.9±0.8
		+SMoRe	65.6±2.8	88.0±0.7	85.6±0.8	**83.5±0.9**	88.9±1.3	88.8±0.7	**89.5±0.5**	**86.6±0.7**
	Fast3R	Baseline	71.2±2.5	84.3±1.5	86.1±1.3	**86.6±1.3**	90.9±1.6	88.3±1.6	86.7±1.3	82.4±1.3
		+SMoRe	71.2±2.5	84.3±1.5	**87.0±1.4**	86.5±1.4	90.9±1.6	88.3±1.6	**87.1±1.3**	**84.3±1.3**
	MASt3R	Baseline	85.5±2.1	95.3±0.2	95.2±0.3	93.8±0.3	93.0±1.3	89.6±1.5	88.5±1.3	86.6±1.1
		+SMoRe	85.5±2.1	95.3±0.2	**95.3±0.3**	**94.3±0.3**	93.0±1.3	89.6±1.5	**88.7±1.3**	**86.8±1.2**
	VGGT	Baseline	88.7±2.6	88.5±1.9	88.7±1.8	88.5±1.7	95.7±0.8	94.5±0.9	92.2±1.1	90.0±0.7
		+SMoRe	88.7±2.6	88.5±1.9	**89.6±1.8**	**89.1±1.8**	95.7±0.8	94.5±0.9	**92.6±1.2**	**93.9±0.6**
	VGGT-Ω	Baseline	88.9±2.0	95.7±1.1	95.4±1.1	94.9±1.1	95.7±0.9	93.2±1.3	93.6±0.6	91.6±0.6
		+SMoRe	88.9±2.0	95.7±1.1	**95.5±1.1**	**95.2±1.1**	95.7±0.9	93.2±1.3	**94.2±0.7**	**94.7±0.5**
	π^3	Baseline	88.2±2.1	96.4±0.8	95.9±1.0	95.2±1.1	96.3±0.7	94.2±1.3	93.7±0.6	89.9±0.7
		+SMoRe	88.2±2.1	96.4±0.8	95.9±1.0	**95.4±1.0**	96.3±0.7	94.2±1.3	**94.8±0.6**	**93.3±0.7**
Streaming	StreamVGGT	Baseline	81.2±0.6	78.6±0.5	77.7±0.6	OOM	92.7±1.2	90.2±0.8	87.5±0.8	OOM
		+SMoRe	**86.7±0.6**	**86.6±0.5**	**86.8±0.6**	**85.7±0.7**	92.7±1.2	**92.8±0.8**	**92.0±0.8**	**85.9±1.0**
	STream3R	Baseline	89.2±0.2	87.0±0.2	OOM	OOM	92.1±1.0	OOM	OOM	OOM
		+SMoRe	**93.8±0.2**	**93.2±0.2**	**92.7±0.4**	**89.3±0.6**	92.1±1.0	**89.0±0.9**	**87.6±1.1**	**87.6±0.8**

Dense-geometry metrics. For both datasets, we report *accuracy*, *completeness*, and *overall* reconstruction error. Accuracy measures the distance from the reconstructed geometry to the reference geometry, while completeness measures the reverse distance from the reference geometry to the reconstruction. We define the overall error as the mean of accuracy and completeness. All distances are reported in centimetres, and lower values indicate better reconstruction. Table 1 reports the mean and standard deviation over the corresponding evaluation set.

I.3 Camera Pose Results Across Sequence Lengths

The full numerical results corresponding to Fig. 2 are reported in Table S2. The table evaluates the same Baseline/+SMoRe pairs described in Appendix I.1 as the number of input frames increases from $N = 10$ to $N = 200$.

For the feed-forward geometry foundation models, the two methods are identical at the shortest sequence lengths whenever the sequence fits within the initial processing context. Differences become more apparent as the sequence grows and predictions must be connected across multiple windows. This pattern is strongest on CO3Dv2: at $N = 200$, for example, VGGT improves from 90.0 to 93.9 AUC@30, VGGT-Ω from 91.6 to 94.7, and π^3 from 89.9 to 93.3. The improvements on RE10K are generally smaller, consistent with its predominantly forward camera motion providing fewer opportunities to retrieve previously observed regions.

The streaming models expose a complementary effect. Their native causal inference becomes increasingly expensive with sequence length and exceeds the tested 46 GB memory budget for several longer inputs. The windowed SMoRe configuration remains executable at these lengths while retaining access to earlier observations through the external memory. These comparisons use the same sampled frames whenever both variants can be evaluated.

I.4 Full 7-Scenes SLAM Results

Table S3 reports the full per-scene SLAM results for 7-Scenes. These are the detailed numbers summarized in the main-text SLAM comparison. We report absolute trajectory error (ATE) RMSE in metres, after similarity alignment to the ground-truth trajectory, so lower is better.

Table S3: **Full 7-Scenes SLAM results.** We report per-scene and average ATE RMSE (m↓) on the seven 7-Scenes sequences: chess, fire, heads, office, pumpkin, kitchen, and stairs. Lower is better. We compare NICER-SLAM, DROID-SLAM, MASt3R-SLAM, GLUEMAP, TALO, VGGT-SLAM++, ViSTA-SLAM, SLAM-Former, CUT3R, VGGT-SLAM, and VGGSfM (Zhu et al., 2024; Teed & Deng, 2021; Murai et al., 2025; Pan et al., 2026; Zhang et al., 2026a; Mandal et al., 2026; Zhang et al., 2026b; Yuan et al., 2026; Wang et al., 2025b; Maggio et al., 2025; Wang et al., 2024a). Calibrated methods receive camera intrinsics, whereas uncalibrated methods do not. A superscript * marks an uncalibrated version of a method that is also reported in a calibrated setting. Our results use frozen VGGT-Ω with 32-frame windows. Within each calibration block, the lowest value in each scene column is shown in bold.

Model	Method	chess	fire	heads	office	pumpkin	kitchen	stairs	avg
Calibrated methods									
NICER-SLAM	–	0.033	0.069	0.042	0.108	0.200	0.039	0.108	0.086
DROID-SLAM	–	0.036	0.027	0.025	0.066	0.127	0.040	0.026	0.049
MASt3R-SLAM	–	0.053	0.025	0.015	0.097	0.088	0.041	**0.011**	0.047
GLUEMAP	–	0.036	0.031	0.018	0.079	0.092	0.046	0.051	0.050
TALO	–	0.034	0.021	0.012	0.054	0.061	0.028	**0.010**	**0.031**
Uncalibrated methods									
VGGT-SLAM++	–	0.034	**0.023**	0.017	0.104	0.127	0.085	0.060	0.064
ViSTA-SLAM	–	0.036	0.033	0.019	0.103	0.133	0.058	0.093	0.067
SLAM-Former	–	0.039	0.033	0.018	0.070	0.065	0.035	0.033	0.042
CUT3R	–	0.046	0.043	0.055	0.120	0.096	0.061	0.086	0.073
MASt3R-SLAM*	–	0.063	0.046	0.029	0.103	0.114	0.074	0.032	0.066
VGGT-SLAM	Sim(3), w=32	0.037	0.026	0.018	0.104	0.133	0.061	0.093	0.067
VGGT-SLAM	SL(4), w=32	0.036	0.028	0.018	0.103	0.133	0.058	0.093	0.067
VGGSfM	–	0.052	0.038	0.021	0.076	0.104	0.059	0.061	0.059
GLUEMAP*	–	0.042	0.039	0.022	0.088	0.105	0.054	0.059	0.058
TALO*	–	0.028	**0.023**	0.014	0.049	0.082	0.033	0.038	0.038
VGGT-Ω (uncalibrated)									
VGGT-Ω	Baseline	0.025	0.024	**0.012**	0.047	0.161	0.026	0.020	0.045
VGGT-Ω	+SMoRe	**0.021**	0.031	0.013	0.031	**0.061**	**0.025**	0.022	**0.029**

The table separates calibrated and uncalibrated methods because these settings are not directly comparable: calibrated methods receive camera intrinsics, whereas uncalibrated methods do not. Within the VGGT-Ω block, the SMoRe rows show the effect of online revision and the final pose-graph refinement on the same frozen geometry foundation model.

I.5 Novel View Synthesis

The novel view synthesis experiment in Sec. 3.4 asks whether the repeated 3D prediction from Appendix H improves rendered views when the rendering model itself is held fixed.

We train the rendering decoder on approximately 19K Objaverse-LVIS objects (Deitke et al., 2023). Evaluation uses a held-out subset of Objaverse-LVIS and, separately, all 1,033 Google Scanned Objects (GSO) (Downs et al., 2022). No GSO object is used to train the decoder, so GSO provides a separate test collection.

Each evaluated object uses four input views and ten held-out target views at 256×256 resolution. Baseline and +SMoRe use the same RGB inputs, cameras, decoder weights, and target views; only the geometry supplied to the decoder changes. The complete camera, rendering, and decoder-training protocol is given in Appendix J.

Image metrics. We report PSNR, SSIM, and LPIPS (Zhang et al., 2018). PSNR measures pixel-level reconstruction fidelity and SSIM measures structural similarity; higher is better for both. LPIPS measures perceptual distance between the rendered and target images; lower is better. Table 2 reports the mean and standard deviation across the evaluated objects.

J Novel View Synthesis Protocol and Decoder

Section 3.4 evaluates whether the improved 3D geometry produced by SMoRe also leads to better rendered views. To isolate this effect, we keep the rendering model fixed: for each geometry

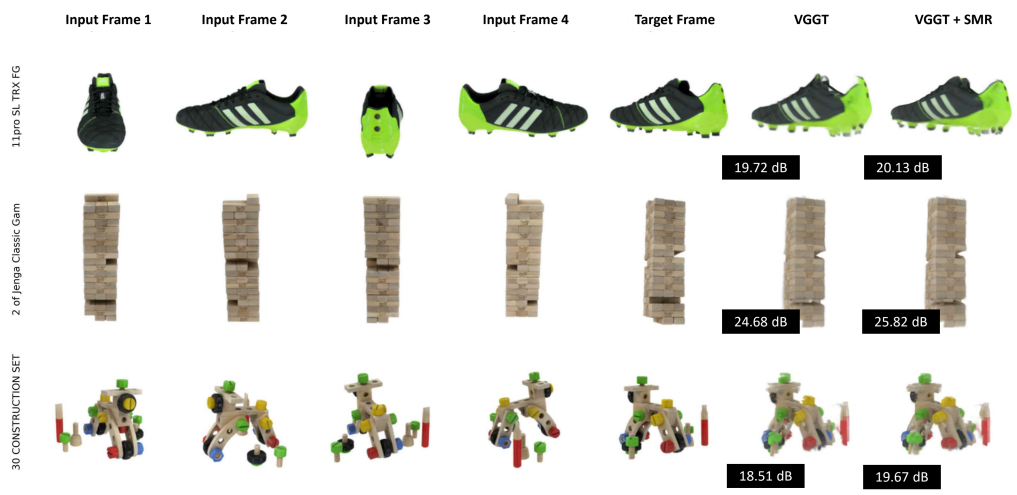


Figure S2: **More visualization examples of Novel View Synthesis using SMoRe.** Each row shows a different object from the GSO dataset (Downs et al., 2022). From left to right, it shows the 4 input images, followed by the target frame, the Baseline (VGGT here) rendering, and the same baseline with SMoRe.

foundation model, we train one decoder using its original geometry and then use exactly the same frozen decoder for both Baseline and +SMoRe. Thus, any difference at test time comes from the geometry supplied to the renderer rather than from retraining the rendering model.

There is no long-range trajectory revision or pose-graph optimization in this experiment. The NVS experiment uses the repeated-prediction procedure from Appendix H on the same four input frames.

J.1 Rendering and Camera Setup

Each object is normalized to a sphere of radius 0.5 centred at the origin. Cameras are placed on a sphere of radius 2.0 with a vertical field of view of $40°$. All images are rendered at 256×256 resolution using the OpenCV camera convention in a y-up world.

We render the object images with Blender Cycles using 32 adaptive samples, denoising, a white background, and one fixed sun illumination per object. The rendered alpha channel provides the object mask.

For Objaverse-LVIS, approximately 19K training objects are sampled across 1,156 LVIS categories using a fixed seed. We render 16 views per object. Camera azimuth is sampled uniformly from $[0°, 360°)$ and elevation from $[-20°, 60°]$.

For GSO, the four input cameras are placed at elevation $20°$ and azimuths

$$0°, \quad 90°, \quad 180°, \quad 270°.$$

Ten target cameras are sampled from the same angular range used during training, using a fixed seed for each object. No GSO object is used to train the decoder.

J.2 Geometry Supplied to the Decoder

The geometry foundation model receives the four RGB input frames and predicts cameras together with either point maps or depth. Because the decoder renders in the known normalized object coordinate system, we first align the predicted cameras to the four input cameras using a similarity transformation. The same transformation is applied to the predicted 3D geometry.

When the model provides native point maps, we use them directly. When it provides depth, we unproject the depth using the corresponding predicted camera parameters. The resulting geometry is resampled onto the 256×256 rendering grid without cropping.

The Baseline and +SMoRE inputs differ only in how this geometry is obtained. Baseline uses one prediction from the four input frames. +SMoRE processes the same four frames under four input orderings, aligns the resulting predictions, and combines their per-pixel 3D points using the consensus procedure in Appendix H. Thus,

$$\text{Baseline:} \qquad \{I_1, \ldots, I_4\} \rightarrow \text{one geometry estimate},$$

whereas

$$+\text{SMoRE}: \qquad \{I_1, \ldots, I_4\} \rightarrow \text{repeated predictions} \rightarrow \text{aligned consensus geometry}.$$

The RGB inputs, target cameras, and rendering decoder are otherwise unchanged.

J.3 Gaussian Decoder

We use a lightweight U-Net with approximately 1.2M parameters to convert the geometry associated with each input pixel into parameters for 3D Gaussian rendering. The decoder receives eight channels per pixel:

$$\underbrace{3}_{\text{RGB}} + \underbrace{3}_{\text{3D position}} + \underbrace{1}_{\text{depth}} + \underbrace{1}_{\text{confidence}} = 8.$$

Only valid geometry inside the object mask contributes Gaussians.

For each valid input point, the decoder predicts a position offset, scale, rotation, opacity, and colour residual. The position offset is bounded by 0.02 normalized object units, corresponding to 4% of the object radius. This restriction prevents the decoder from freely reconstructing the scene independently of the geometry supplied by the backbone.

The predicted Gaussians are rendered with gsplat (Kerbl et al., 2023; Ye et al., 2025) on a white background. Initialization is close to directly splatting the coloured 3D points, so the decoder acts primarily as a rendering readout of the supplied geometry.

Training. A separate decoder is trained for each geometry foundation model using only that model's Baseline geometry. The geometry foundation model itself remains frozen.

Training uses AdamW for 30K steps with an initial learning rate of 2×10^{-4}, followed by warm-up and cosine decay. At each step, we sample one object, four input views, and four target views. The training loss is

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + 0.5\,\mathcal{L}_{\text{LPIPS}}.$$

Neither SMoRE nor the geometry foundation model is optimized using this loss.

After training, the decoder weights are frozen. The same decoder is then evaluated once using Baseline geometry and once using +SMoRE geometry. This paired design ensures that improvements cannot be attributed to a decoder trained specifically for SMoRE.

Evaluation. At test time, each object is rendered from ten held-out target cameras. We compute full-image PSNR, SSIM, and LPIPS against the corresponding target images, using the protocol described in Appendix I.5. Because the decoder, RGB inputs, cameras, and renderer are identical for each Baseline+SMoRE pair, this experiment measures how changes in the supplied 3D geometry propagate to the final rendered views.

J.4 How Many Repeated Predictions Should Be Combined?

The dense experiments use the repeated-prediction procedure described in Appendix H. In Table S4, we first vary the number of input-order passes for VGGT on the same 299 GSO objects, and then test the scale-alignment and fallback choices. The input frames, target views, decoder weights, renderer, and reported Gaussian count of 76.7K are fixed.

Two and four passes provide almost identical gains on this controlled VGGT subset, whereas eight passes provide the largest improvement but double the inference budget of the four-pass setting. The main experiments therefore use four passes as a shared compute budget rather than as an empirically optimal number for every geometry foundation model.

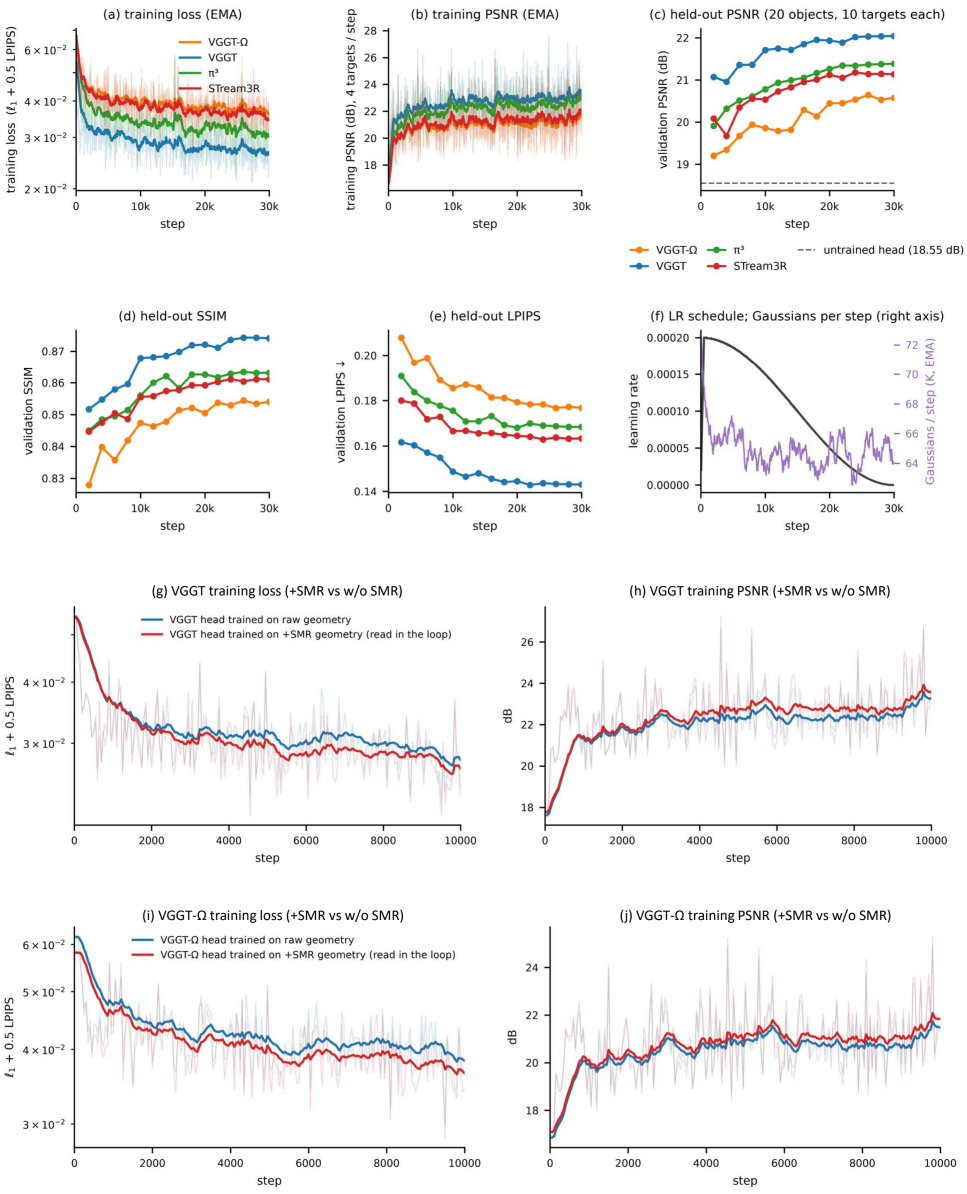


Figure S3: **Training diagnostics for the Gaussian decoders.** Shown are representative decoder-training runs for VGGT-Ω, VGGT, π^3, and STream3R. The panels report training loss, training PSNR, validation performance on held-out objects, the learning-rate schedule, and the number of valid Gaussians. Each decoder is trained using Baseline geometry and frozen before the Baseline+SMORE comparison. The figure therefore monitors decoder convergence rather than training of the geometry foundation models or SMORE.

Removing the symmetric scale re-measurement gives 20.64 dB compared with 20.60 dB for the default four-pass read. Because the reported confidence interval is relative to the single-pass reference rather than to the default four-pass configuration, this experiment does not show that scale re-measurement itself provides a significant improvement.

The fallback rule has a clearer effect. When fewer than two predictions agree, replacing the median fallback with abstention reduces PSNR from 20.60 to 18.41 dB. We therefore retain the median fallback in the main NVS configuration.

Table S4: **Repeated-prediction and fusion ablations on GSO.** VGGT is evaluated on the same 299 objects. ΔPSNR and its paired 95% interval are measured relative to the single-pass reference. The main NVS experiments use four input-order passes.

Setting	PSNR↑	SSIM↑	LPIPS↓	ΔPSNR	Improved
Single pass	20.33	0.819	0.159	–	–
2 passes	20.59	0.821	0.151	$+0.26$ $[+0.15, +0.36]$	69%
4 passes (default)	20.60	0.821	0.151	$+0.27$ $[+0.16, +0.37]$	68%
8 passes	**21.40**	**0.837**	**0.136**	$\mathbf{+1.07}$ $[+0.94, +1.21]$	86%
4 passes, no scale re-measurement	20.64	0.822	0.150	$+0.31$ $[+0.22, +0.40]$	69%
4 passes, abstain fallback	18.41	0.818	0.185	-1.92	–

Table S5: **Eight-pass cross-model diagnostic.** Each entry reports reference PSNR → eight-pass PSNR.

Model	GSO PSNR↑	Held-out Objaverse-LVIS PSNR↑
VGGT (Wang et al., 2025a)	$20.71 \rightarrow \mathbf{21.81}$	$22.79 \rightarrow \mathbf{22.97}$
VGGT-Ω (Wang et al., 2026a)	$18.89 \rightarrow 18.85$	$20.84 \rightarrow 20.83$
π^3 (Wang et al., 2026b)	$21.91 \rightarrow 21.91$	$22.19 \rightarrow 22.19$
STream3R (Lan et al., 2026)	$18.79 \rightarrow 17.26$	$21.15 \rightarrow 20.58$

Does increasing the pass count help every geometry model? To test whether the eight-pass improvement above generalizes across geometry foundation models, we also evaluate a separate eight-pass diagnostic shown in Table S5. This is not a matched pass-count sweep: each row is compared with its own reported reference.

The effect of additional passes is model dependent. Eight passes improve VGGT, are approximately neutral for VGGT-Ω and π^3, and reduce the reported STream3R scores. This supports using a common four-pass budget across geometry foundation models in the main comparison rather than increasing the number of passes solely because it benefits VGGT on one subset.

K Controlled Ablations

We expand the model analysis of Sec. 3.5 in the same order as the main text. We first test whether the scaffold improves retrieval over direct descriptor matching, and then vary the visual descriptor used to query the scaffold. We next examine how much historical evidence is needed to verify a revisit and whether an accepted revisit must revise the trajectory online. Finally, we study sensitivity to the scaffold and window configurations and analyze the repeated-prediction procedure used for dense geometry. We use the ATE(m) on 7-scenes dataset (Shotton et al., 2013) and AUC@30 on CO3Dv2 dataset (Reizenstein et al., 2021) in this ablation study.

Evaluation protocol. Unless stated otherwise, the pose ablations use seven 7-Scenes seq-01 trajectories with 200 frames and VGGT-Ω (Wang et al., 2026a), together with twelve 200-frame CO3Dv2 trajectories and VGGT (Wang et al., 2025a). Paired variants use the same input frames. The common raw references are 0.045 m ATE on 7-Scenes and 88.8 AUC@30 on CO3D. We use *Online* for the trajectory available during sequence processing and *Final* for the result after the final global refinement. Lower ATE is better, whereas higher AUC@30 is better. Dense-read ablations use the separate GSO protocol described below.

K.1 Does the Scaffold Improve Retrieval?

The first comparison (Table S6)(Group A) isolates how stored frames are ranked. The default scaffold maps the DINOv2 cue into the memory-address space and ranks stored addresses, whereas the flat baseline directly ranks the same stored DINOv2 descriptors by cosine similarity. The stored frames, top-five candidate budget, geometric verification, and subsequent trajectory revision are unchanged.

The scaffold improves both benchmarks under the same downstream revision procedure. On 7-Scenes, final ATE decreases from 0.034 to 0.029 m, while on CO3Dv2 final AUC@30 increases from 90.9

Table S6: **Ablation studies of SMORE.** We vary one design factor at a time while keeping the remaining pipeline fixed. The groups follow the same order as the design analysis in Sec. 3.5. Except for the window-size study, the common raw references are 0.045 m ATE on 7-Scenes and 88.8 AUC@30 on CO3Dv2. *Online* denotes the running SMORE result, while *Final* includes the final global refinement in Appendix G. For Groups E and F, the value in parentheses after the 7-Scenes Final ATE is the mean number of accepted revisits per sequence. The default configuration uses the memory scaffold, 64 active units in the memory address $\mathbf{h}$, DINOv2, two retrieved pairs, $N_h = 1024$, a 48^2 horizontal torus, and $W = 32$ with overlap ratio 0.5.

	7-Scenes ATE (m)↓		CO3Dv2 AUC@30↑	
Configuration	Online	Final	Online	Final
Group A. Memory scaffold				
Scaffold address (default)	0.030	0.029	93.4	93.3
Direct DINOv2 cosine	0.035	0.034	91.3	90.9
Group B. Memory sparsity				
$\sigma = 64$ (default)	0.030	0.029	93.4	93.3
$\sigma = 16$	0.038	0.041	82.7	85.3
$\sigma = 128$	0.032	0.031	93.1	92.9
Group C. Visual descriptors				
DINOv2 CLS (default) (Oquab et al., 2024)	0.030	0.029	93.4	93.3
NetVLAD (Arandjelovic et al., 2016)	0.050	0.051	92.2	91.9
EigenPlaces (Berton et al., 2023)	0.066	0.068	92.4	92.4
CosPlace (Berton et al., 2022)	0.263	0.265	92.6	92.4
MixVPR (Ali-Bey et al., 2023)	0.053	0.053	92.9	92.8
SALAD (Izquierdo & Civera, 2024)	0.056	0.057	92.9	92.8
BoQ (Ali-Bey et al., 2024)	0.127	0.128	92.9	92.7
DINOv2 + EigenPlaces	0.048	0.049	93.3	93.2
DINOv2 + SALAD	0.133	0.134	92.9	92.7
Group D. Scaffold dimension				
$N_h = 1024$, torus 48^2 (default)	0.030	0.029	93.4	93.3
$N_h = 512$	0.031	0.029	92.5	92.4
$N_h = 8192$	0.030	0.028	93.5	93.2
Torus 16^2	0.030	0.026	93.6	93.2
Torus 64^2	0.033	0.029	92.9	92.9
Group E. Retrieval evidence				
2 retrieved pairs (default)	0.030	0.029 (L=4.1)	93.4	93.3
1 retrieved pair	0.053	0.053 (L=3.7)	93.2	93.1
3 retrieved pairs	0.029	0.029 (L=4.1)	93.4	93.3
Group F. Progressive placement revision				
Linearly weighted correction (default)	0.030	0.029 (L=4.1)	93.4	93.3
Distributed correction (equally weighted)	0.030	0.030 (L=4.0)	93.3	93.2
Group G. Window size and overlap				
$W = 32$, overlap ratio $= 0.7$	0.027	0.026	90.2	90.3
$W = 32$, overlap ratio $= 0.5$ (default)	0.030	0.029	93.4	93.3
$W = 32$, overlap ratio $= 0.3$	0.033	0.033	89.8	90.1
$W = 16$, overlap ratio $= 0.7$	0.027	0.026	90.8	91.2
$W = 16$, overlap ratio $= 0.5$	0.030	0.029	89.8	90.6
$W = 16$, overlap ratio $= 0.3$	0.034	0.033	88.2	88.4
$W = 64$, overlap ratio $= 0.7$	0.025	0.024	87.2	86.9
$W = 64$, overlap ratio $= 0.5$	0.030	0.029	91.2	91.9
$W = 64$, overlap ratio $= 0.3$	0.032	0.030	87.2	86.9

to 93.3. This comparison supports the scaffold as the retrieval index used by SMORE. It does not attribute the gain specifically to the internal continuous-attractor dynamics; those properties are tested separately in Appendix E.

K.2 WHICH VISUAL DESCRIPTOR SHOULD QUERY THE SCAFFOLD?

We next keep the scaffold and the downstream revision procedure fixed and change the visual descriptor used to retrieve stored frames (Table S6)(Group C). We compare the default DINOv2 CLS descriptor (Oquab et al., 2024) with NetVLAD (Arandjelovic et al., 2016), EigenPlaces (Berton et al.,

2023), CosPlace (Berton et al., 2022), MixVPR (Ali-Bey et al., 2023), SALAD (Izquierdo & Civera, 2024), and BoQ (Ali-Bey et al., 2024), together with two concatenated descriptors.

The production DINOv2 configuration uses its fixed 0.5 appearance gate. The alternative descriptors are evaluated without descriptor-specific gates so that a separate threshold is not tuned for each method. The top-five retrieval budget, geometric verification, online revision, and final refinement remain unchanged.

None of the tested alternatives improves over the production DINOv2 configuration on the final reconstruction. This result also shows that retrieving visually plausible frames is not sufficient by itself: the retrieved frames must provide useful geometric evidence for the subsequent verification and trajectory revision. Appendix N separately examines how changing the retrieved historical frames can affect the current window's local prediction before verification.

K.3 How Much Historical Evidence Is Needed?

Section 2.2 uses two retrieved pairs from different source windows to verify a revisit. In Table S6(Group E), we first vary the number of retrieved pairs while keeping the rest of the pipeline unchanged. We then disable the online placement update to distinguish verifying a revisit from actually using it to revise the trajectory.

Using only one retrieved pair substantially degrades 7-Scenes ATE, whereas three pairs give similar online performance to two on both benchmarks. Three pairs keep the 7-Scenes ATE to 0.029. We therefore use two pairs as the default evidence.

The online placement update is also important. When revisits are proposed and verified but not applied online, 7-Scenes ATE increases from 0.030 to 0.042 m and CO3Dv2 AUC@30 decreases from 93.4 to 91.3. The final global refinement can recover part of this loss, but those corrections are unavailable while later windows are being processed. The no-update row remains different from the raw trajectory because the default configuration retains the joint-pass local predictions; this effect is isolated in Appendix N.

The additional rows test how accepted measurements are applied. The default online update re-solves the affected local pose graph using the accepted revisit measurement and the sequential constraints. The *Distributed correction* row instead applies the direct interpolation of the measured correction across the affected windows. Removing outlier rejection from the final global solve mainly reduces the final CO3Dv2 score, while separately re-measuring the historical frames changes the reported accuracy only modestly.

K.4 Sensitivity to Scaffold and Window Size

We next vary the dimensions of the scaffold representation and the window used by the geometry foundation model in Table S6(Group D). These experiments test sensitivity rather than removing a component: changing these quantities also changes the memory resolution or the input context available to the geometry foundation model.

Performance is relatively stable across the tested scaffold dimensions. Increasing the address dimension from 1024 to 8192 does not consistently improve the final result, and increasing the torus resolution to 64^2 can reduce accuracy. Therefore, it shows that the memory structure itself is good at recall, and increasing the dimensionality of the address does not necessarily show us gains. On the other hand (Group B), optimal sparsity $\sigma = 64$ is set so that sparse addresses are near orthogonal to each other with minimum overlap, while too much sparsity results in loss of information.

Window size has a stronger effect because it changes both the geometry predictions and the number of junctions between windows. We observe that the performance over both 7-scenes and CO3Dv2 improves with larger window size and moderate overlap ratio. As the overlap ratio increases, the number of windows increases, and so does the chaining error evident in Table S6. Also observe that a smaller window size increases the number of windows too, which is why we see degraded performance for $W = 16$. Since the gains are so marginal, we select the best window size and overlap ratio with a simple grid search over these two parameters.

Summary. Together, these ablations support the main design choices used by SMoRE: scaffold-based retrieval improves over direct descriptor matching, DINOv2 provides the strongest tested retrieval cue, two retrieved pairs provide an effective verification budget, and applying accepted revisit measurements online substantially improves the running trajectory. The scaffold dimensions, window size, and activation sparsity of the scaffold address were all tuned to maximize performance across all tasks.

L Compute and Memory Measurements

We examine how SMoRE changes the computational and memory requirements of geometry foundation models as sequences become longer. For feed-forward models, long sequences are processed as overlapping windows whose local reconstructions must be connected across the trajectory, while streaming models retain causal context from earlier frames inside the model. In contrast, SMoRE keeps geometry-model inference bounded to the current window and stores long-horizon information in its external memory. Earlier observations are brought back into the geometry foundation model only when they are retrieved for revisit verification.

All measurements use one RTX 6000 Ada GPU with a tested 46 GB GPU-memory budget, batch size one, and public model weights. We use the CO3Dv2 camera-pose experiment from Sec. 3.1. The default SMoRE configuration uses 32-frame windows; during revisit verification, up to four retrieved historical frames can be processed together with the current window, giving at most 36 frames in one geometry-model invocation.

L.1 Peak GPU Memory with Increasing Sequence Length

We first study peak GPU allocation as the number of input frames increases from $N = 10$ to $N = 200$. Table S7 shows how the bounded-window design of SMoRE limits the GPU working set over longer sequences.

For feed-forward models, the sequence is reconstructed using overlapping windows and the resulting local predictions are chained through their shared frames. As the sequence grows, the baseline can retain increasing amounts of intermediate predictions and geometric information needed to connect these windows. With SMoRE, the geometry foundation model still operates on a bounded current window, while earlier observations are stored in the external scaffold and content bank and are selectively reintroduced only when needed. For example, VGGT grows from 9.1 GB at $N = 10$ to 23.9 GB at $N = 200$ in the baseline, whereas SMoRE remains between 8.5 and 9.6 GB. VGGT-Ω similarly grows from 5.7 to 19.5 GB in the baseline but remains near 7 GB with SMoRE.

Streaming models exhibit a different source of GPU-memory growth. Their native causal inference retains internal states or cached tokens from previous frames so that earlier observations can influence later predictions. These GPU-resident states grow with the sequence. StreamVGGT therefore increases from 8.6 GB at $N = 10$ to 22.7 GB at $N = 100$ and exceeds the tested 46 GB budget at $N = 200$. STream3R exceeds the same budget from $N = 50$ onward. With SMoRE, long-range information is instead maintained externally and the geometry model receives bounded windows plus only a small number of retrieved frames. StreamVGGT therefore uses 11.4 GB at $N = 200$, while STream3R remains feasible at 25.4 GB.

The total memory used by SMoRE is not constant: the content bank grows as additional observations are stored. The important distinction is that this history does not remain part of the geometry foundation model's active GPU context. Consequently, peak GPU allocation is governed mainly by the bounded model invocation rather than by the complete trajectory.

L.2 Compute and Runtime at $N = 100$

Reducing peak GPU memory does not make SMoRE computationally free. Instead, SMoRE trades a bounded GPU working set for additional operations required to maintain and verify its external memory. Table S8 measures this trade-off at $N = 100$ and reports accumulated CPU processing time, peak host RAM, peak GPU allocation, total floating-point computation, and end-to-end wall-clock runtime. CPU time is accumulated across host processing and can therefore differ from the wall-clock runtime, while RAM reports the peak resident host-memory footprint.

Table S7: **Peak GPU allocation (GB) with increasing sequence length using the CO3Dv2 dataset during the camera pose estimation experiment in Sec. 3.1.** *Raw/Native* uses each model's chained inference mode and native inference for streaming models. OOM indicates that the tested 46 GB GPU budget was exceeded; n/f indicates that the run was not feasible under the host-memory or runtime budget.

	$N = 10$		$N = 50$		$N = 100$		$N = 200$	
Model	Raw/Native	+SMoRe	Raw/Native	+SMoRe	Raw/Native	+SMoRe	Raw/Native	+SMoRe
DUSt3R (Wang et al., 2024b)	4.9	4.4	OOM	25.8	n/f	28.8	n/f	32.1
Fast3R (Yang et al., 2025)	6.3	5.7	11.4	9.1	13.8	9.2	18.7	9.2
MASt3R (Leroy et al., 2024)	3.2	3.2	20.7	6.7	n/f	7.6	n/f	8.8
VGGT (Wang et al., 2025a)	9.1	8.5	11.1	9.5	15.1	9.6	23.9	9.6
VGGT-Ω (Wang et al., 2026a)	5.7	5.7	8.1	6.8	11.9	6.9	19.5	7.0
π^3 (Wang et al., 2026b)	8.8	8.8	16.3	13.0	26.5	14.1	OOM	14.6
StreamVGGT (Zhuo et al., 2026)	8.6	8.1	14.8	11.0	22.7	11.1	OOM	11.4
STream3R (Lan et al., 2026)	10.9	10.0	OOM	21.1	OOM	25.4	OOM	25.4

The additional computation comes from three main sources. First, incoming frames require visual-descriptor extraction, scaffold lookup, and content-bank operations. Second, when a candidate revisit is retrieved, the historical frames are processed jointly with the current window so that their geometric relationship can be re-estimated. Third, accepted revisits require similarity alignment and trajectory revision. These operations increase CPU processing and can add geometry-model forward passes, explaining the higher CPU time and PFLOP count for SMoRe.

The external memory also introduces a modest host-RAM cost. For most models, this increase is small at $N = 100$: VGGT changes from 10.77 to 10.78 GB, VGGT-Ω from 9.88 to 9.89 GB, π^3 from 11.58 to 11.59 GB, DUSt3R from 16.94 to 17.02 GB, and MASt3R from 17.23 to 17.29 GB. StreamVGGT increases from 10.45 to 10.79 GB, while Fast3R shows the largest increase, from 7.42 to 8.60 GB. This reflects the intended resource trade-off: persistent observations and memory metadata are maintained outside the geometry model, allowing the GPU-resident working set to remain bounded.

Peak GPU allocation and total computation should therefore be interpreted separately. SMoRe may execute more total operations while still using less peak GPU memory because additional verification passes are performed sequentially on bounded inputs. For example, VGGT increases from 2.20 to 2.44 PFLOP and from 0.33 to 0.54 min runtime, while its peak GPU allocation decreases from 15.1 to 9.6 GB. VGGT-Ω increases from 1.08 to 1.26 PFLOP while reducing GPU allocation from 11.9 to 6.9 GB, and π^3 reduces GPU allocation from 26.5 to 14.1 GB despite increasing from 1.52 to 1.82 PFLOP.

Finally, wall-clock runtime is dominated by the cost of the underlying geometry foundation model. The same SMoRe memory operations are applied across models, yet +SMoRe requires only 0.38 min for VGGT-Ω and 0.42 min for Fast3R, compared with 17.01 min for DUSt3R and 18.49 min for MASt3R. Thus, the absolute runtime cost is determined primarily by the geometry-model inference required during ordinary processing and revisit verification rather than by the scaffold itself.

M Implementation Settings and Reproducibility

This section summarizes the shared implementation settings used by SMoRe. Task-specific dataset sampling and evaluation protocols are given in Appendix I, while the novel-view-synthesis training and rendering settings are given in Appendix J. Table S9 therefore focuses on the settings that determine the scaffold, retrieval, geometric verification, and repeated-prediction procedures.

Frozen and updated components. All geometry foundation models use their public pretrained weights and remain frozen throughout SMoRe. The DINOv2 encoder used for retrieval is also frozen. The fixed projection W_{gh} is sampled once, while W_{hg} is fitted once before inference from sampled scaffold states. Only the appearance–address associations W_{sh} and W_{hs} are updated online as new frames are stored, using the RLS update in Eq. 2. Thus, trajectory improvement does not result from fine-tuning the geometry foundation model.

Table S8: **Compute and memory at $N = 100$ using the CO3Dv2 dataset during the camera pose estimation experiment in Sec. 3.1.** Measurements use one RTX 6000 Ada GPU with a tested 46 GB GPU-memory budget and batch size one. We report accumulated CPU processing time, peak host RAM, peak GPU allocation, total floating-point computation, and end-to-end runtime. *Baseline* uses each model's raw windowed baseline, while the streaming models use their native baseline. STream3R exceeds the GPU-memory budget under native inference and is marked OOM. +SMoRe uses 32-frame windowed inference with online retrieval, verification, and trajectory revision, without final global refinement. For native STream3R, the reported RAM is the peak observed before GPU OOM.

Model	Method	CPU time (min)	RAM (GB)	GPU (GB)	Compute (PFLOP)	Runtime (min)
DUSt3R (Wang et al., 2024b)	Baseline (raw)	40.78	16.94	25.8	10.58	12.21
	+SMoRe	52.77	17.02	28.8	13.71	17.01
Fast3R (Yang et al., 2025)	Baseline (raw)	0.84	7.42	13.9	0.66	0.13
	+SMoRe	2.20	8.60	9.2	0.71	0.42
MASt3R (Leroy et al., 2024)	Baseline (raw)	36.35	17.23	7.1	9.32	17.79
	+SMoRe	37.90	17.29	7.6	11.89	18.49
VGGT (Wang et al., 2025a)	Baseline (raw)	0.55	10.77	15.1	2.20	0.33
	+SMoRe	1.10	10.78	9.6	2.44	0.54
VGGT-Ω (Wang et al., 2026a)	Baseline (raw)	0.40	9.88	11.9	1.08	0.19
	+SMoRe	0.87	9.89	6.9	1.26	0.38
π^3 (Wang et al., 2026b)	Baseline (raw)	0.47	11.58	26.5	1.52	0.28
	+SMoRe	0.97	11.59	14.1	1.82	0.52
StreamVGGT (Zhuo et al., 2026)	Baseline (native)	0.83	10.45	22.7	1.28	0.62
	+SMoRe	1.20	10.79	11.1	1.59	0.57
STream3R (Lan et al., 2026)	Baseline (native)	–	9.92		OOM ($>$ 46 GB GPU)	
	+SMoRe	1.47	10.26	25.4	2.36	0.72

The novel-view-synthesis decoder is the only separately trained task-specific component. As described in Appendix J, one decoder is trained for each geometry foundation model using its Baseline geometry and is then frozen for both Baseline and +SMoRe evaluation. Consequently, the decoder is not optimized using the geometry produced by SMoRe.

Task-specific protocol differences. The settings in Table S9 describe the default trajectory configuration rather than a requirement that every downstream task use the same window structure. The camera-pose and SLAM experiments use the windowing protocol specified in their respective evaluation sections. The compute study in Appendix L measures ordinary 32-frame passes and joint passes containing up to four additional retrieved frames. Replica and ScanNet++ use the default 32-frame windows with 16-frame overlap and online revisit-driven trajectory revision. Novel view synthesis instead uses four input frames and four repeated input orderings, with no long-range trajectory revision or final global pose refinement.

Frame selection and dataset subsets. Window size should not be confused with the number of frames evaluated from a complete sequence. The exact dataset subsets and frame-selection rules are specified in Appendix I. In particular, paired Baseline+SMoRe comparisons use the same selected frames and the same geometry foundation model. This ensures that changes in the reported metrics are attributable to the corresponding SMoRe procedure rather than to different inputs.

Table S9: **Shared implementation settings.** These values define the default SMoRe configuration used in the main experiments unless a task-specific exception is stated explicitly. Spatial quantities used by the scaffold are expressed in median-depth-normalized scene units.

Setting	Value	Defined in
Windowing and retrieval		
Default window / overlap	$32/16$	Sec. 2
Appearance descriptor	DINOv2 ViT-S/14 CLS, 384-D	Sec. 2.2
Retrieval budget	top-5 stored frames	Sec. 2.2
Minimum revisit separation Δ	64 frames	Eq. 3
Appearance threshold τ_{app}	0.5	Eq. 3
Retrieved pairs	2	Appendix F
Partner-frame offset	$j \pm 3$	Appendix F
Pair-agreement threshold	$3^\circ, 0.15E_k$	Eq. 4
Local-geometry policy	joint-pass prediction	Appendix N
Similarity alignment		
Robust reweighting rounds	5	Appendix B
Robust-fit constant κ	2.5	Appendix B
Sequential scale gate γ_s	1.5	Appendix B
Scaffold representation		
Spatial periods	$2.4,\ 3.2,\ 4.0$	Appendix D
Horizontal torus resolution	48×48	Appendix D
Height-ring resolution	48	Appendix D
Orientation-ring resolution	256 per angle	Appendix D
Scaffold-state dimension N_g	$7{,}824$	Appendix D
Address dimension N_h	$1{,}024$	Appendix D
Active address units	64	Appendix D
Pose-to-address projection	$\mathcal{N}(0, 1/7824)$, fixed	Appendix D
Maximum formation spacing	$\lambda_{\min}/6 = 0.4$	Appendix D
Repeated 3D predictions		
Minimum agreeing predictions	2	Appendix H
Main NVS ordering count	4	Appendix J
NVS agreement tolerance	0.01	Appendix H
Dense-read fallback	coordinate-wise median	Appendix H

N Separating Local-Prediction Changes from Trajectory Revision

A retrieved revisit can affect the reconstruction in two different ways. First, processing historical frames together with the current window can change the geometry foundation model's *local prediction* for that window. Second, if the revisit passes geometric verification, the resulting measurement can be used to revise the *global placements* of the affected windows. These two effects are distinct: rejecting a revisit prevents its placement update, but does not necessarily undo a local prediction that has already changed during joint processing.

This section separates these effects. We first define the ordinary and joint passes used by the diagnostic, then compare three policies for deciding which local prediction is retained. We finally test these policies under the default retrieval proposals and under several failure cases.

Evaluation setting. The diagnostics use seven 7-Scenes trajectories with VGGT-Ω and twelve CO3Dv2 trajectories with VGGT, reusing the cached predictions from the controlled pose study. The raw references are 0.045 m ATE on 7-Scenes and 88.8 AUC@30 on CO3D. Because the purpose of this analysis is to isolate what happens during sequence processing, the tables below report the *online* trajectory before the final global refinement. Lower ATE is better and higher AUC@30 is better.

The main pose experiments retain the current-window prediction produced by joint processing with the retrieved historical frames. We refer to this as the *joint* policy below; it corresponds to the *anchored* configuration used in the diagnostic logs. The *ordinary* and *gated* policies are diagnostic variants and do not replace the configuration used for the main results.

Table S10: **Local-prediction policies used in the diagnostic.** All three policies can use the joint pass to measure the historical relationship. They differ only in which prediction is retained for the current window before sequential placement and possible trajectory revision.

Policy	Retained current-window prediction	Role of the joint pass
Joint (default)	Prediction from current + historical frames	Provides both the retained local prediction and the revisit measurement
Ordinary	Prediction from the current window alone	Used only to measure the historical relationship
Gated	Joint prediction if its distortion is small; otherwise ordinary	Provides the revisit measurement and a candidate local prediction

N.1 Why Rejecting a Revisit Does Not Restore the Original Prediction

An *ordinary pass* processes the 32 frames of the current window alone. When two retrieved pairs are available, the *joint pass* instead processes those same 32 current frames together with the four retrieved historical frames. The purpose of the joint pass is to place the current and historical frames in one common local coordinate system so that the revisit can be verified.

However, the joint pass also produces a new prediction for the current window itself. Under the default policy, this joint prediction becomes the current window's retained local camera poses and point maps. The subsequent revisit test therefore controls only whether a retrieval-based *global placement update* is applied. If the revisit is rejected, its placement correction is discarded, but the retained joint-pass local prediction is not replaced by the prediction from the ordinary pass.

Consequently, a rejected historical proposal can still influence later windows through the sequential overlap alignment. This is why revisit verification and local-prediction stability must be analyzed separately.

Measuring the change in the local prediction. To measure how much the historical frames change the current-window prediction, we align the camera predictions of the current frames from the joint pass with those from the ordinary pass. After this alignment, the logged quantity $\epsilon_k^{\mathrm{rot}}$ measures the remaining rotational disagreement for window k, in degrees.

This quantity measures agreement between two model predictions; it is not an error against ground truth. A small $\epsilon_k^{\mathrm{rot}}$ means that the joint pass remains close to the ordinary prediction, while a large value indicates that adding the historical frames substantially changed the predicted local camera geometry.

For the 50 joint-pass windows in the default seed-0 7-Scenes diagnostic, the median residual is 0.117° and the maximum is 0.311°. In a separate seed-1 run, however, one *chess* window reaches 46.1°. For that sequence, ATE increases from 0.025 to 0.459 m even though both tested revisit proposals are rejected. Thus, a large local-prediction change can affect the trajectory without an accepted false revisit.

N.2 Three Ways to Retain the Current-Window Prediction

We compare three policies that differ only in which current-window local prediction is retained after the joint pass. The joint pass is still available for historical re-measurement in all three cases.

Joint policy. The current-window camera poses and point maps are taken directly from the joint pass. This allows the historical frames to influence the local prediction, which can be useful when the retrieved evidence is correct, but also exposes the current prediction to an incorrect proposal before verification.

Ordinary policy. The current window retains the prediction obtained from its ordinary pass. The joint pass is used only to measure the relationship between the historical frames and the current window. The historical predictions from the joint pass are transformed into the retained ordinary coordinate system before verification. Thus, rejecting the revisit leaves the current window with exactly the local prediction it would have had without the historical frames.

Table S11: **Local-prediction selection with default retrieval proposals.** DINOv2 and scaffold seed 0 are used for seven 7-Scenes trajectories and twelve CO3Dv2 trajectories. *Joint* always retains the joint-pass prediction, *Ordinary* always retains the ordinary-pass prediction, and *Gated* switches between them using Eq. 9. All metrics are measured online before final global refinement.

Local prediction	τ_{dist}	7-Scenes ATE (m)↓	CO3Dv2 AUC@30↑
Joint (default)	–	0.030	93.4
Gated	0.5°	0.030	92.5
Gated	1°	0.030	93.0
Gated	2°	0.030	93.2
Gated	5°	0.030	93.4
Ordinary	–	0.030	92.4

Gated policy. The gated policy selects between the two predictions according to their measured rotational disagreement:

$$\mathcal{P}_k^{\mathrm{sel}} = \begin{cases} \mathcal{P}_k^{\mathrm{joint}}, & \epsilon_k^{\mathrm{rot}} \le \tau_{\mathrm{dist}}, \\ \mathcal{P}_k^{\mathrm{ord}}, & \epsilon_k^{\mathrm{rot}} > \tau_{\mathrm{dist}}, \end{cases} \tag{9}$$

where $\mathcal{P}_k^{\mathrm{joint}}$ and $\mathcal{P}_k^{\mathrm{ord}}$ denote the current-window predictions from the joint and ordinary passes, respectively, and τ_{dist} is the angular threshold.

This gate is applied *before* the revisit decision. Passing the gate does not mean that the revisit has been accepted; geometric verification in Appendix F is still required. The gate controls which local prediction is retained, whereas revisit verification controls whether a global placement measurement is accepted.

N.3 Does the Local Prediction Matter Without a Placement Update?

A direct control is obtained by disabling the online placement update. With the ordinary policy and the update disabled, the resulting trajectory matches the raw trajectory on all seven tested 7-Scenes sequences. With the joint policy and the same update disabled, mean ATE changes from 0.0451 to 0.0423 m.

The only difference between these two controls is therefore whether the current window retains the ordinary or joint-pass local prediction. This confirms that joint processing can change the trajectory even when no accepted revisit is written back to the global placements.

The seed-1 *chess* failure above demonstrates the same mechanism in the opposite direction: a large joint-pass distortion can strongly degrade the trajectory even when both proposed revisits are rejected. Together, these controls show that local re-prediction and explicit trajectory revision are separate effects.

N.4 How Much Joint-Pass Distortion Should Be Retained?

We first vary the distortion threshold using the default DINOv2 retrieval proposals and scaffold seed 0. Table S11 reports only the online trajectory so that the comparison isolates local-prediction selection from the final global refinement.

All tested thresholds give the same 7-Scenes ATE at the reported precision. This is consistent with the seed-0 diagnostic above: the largest measured distortion is only 0.311°, so every tested gate with threshold at least 0.5° retains the joint prediction for those measured windows.

CO3Dv2 shows a clearer trade-off. Always retaining the ordinary prediction gives 92.4 AUC@30, compared with 93.4 for the default joint policy. As the gate becomes less restrictive, the score increases from 92.5 at 0.5° to 93.4 at 5°. Thus, suppressing every change introduced by historical frames is not necessarily desirable: some joint-pass changes are useful. The diagnostic instead asks whether large changes can be rejected without discarding all of the benefit of joint processing.

Table S12: **Local-prediction isolation on selected 7-Scenes failure cases.** Values are mean online ATE (m)↓ across the seven trajectories. Within each row, the same retrieval proposals are used for all three local-prediction policies. The DINOv2 row uses scaffold seed 1; the remaining rows use the indicated visual descriptor. All gated results use $\tau_{\mathrm{dist}} = 1^\circ$.

Retrieval proposals	Joint	Gated, 1°	Ordinary
DINOv2, seed 1 (Oquab et al., 2024)	0.092	0.029	0.031
CosPlace (Berton et al., 2022)	0.263	0.048	0.048
BoQ (Ali-Bey et al., 2024)	0.127	0.052	0.051
DINOv2 + SALAD (Izquierdo & Civera, 2024)	0.133	0.050	0.049

N.5 Stress Tests on Failure Cases

The previous experiment uses the default seed and retrieval proposals, for which the measured 7-Scenes distortions are small. We therefore separately examine settings in which the joint prediction becomes unstable. These are stress tests rather than additional main configurations.

Table S12 reports the seven-scene mean online ATE for the seed-1 DINOv2 failure and for three appearance-cue settings that produce large errors with the joint policy. In each row, the retrieval proposal setting is held fixed while only the retained local-prediction policy changes.

The large errors in the joint column are greatly reduced when the ordinary prediction is retained or when the 1° gate rejects large deviations. For example, the seed-1 mean ATE decreases from 0.092 m with joint predictions to 0.029 m with gating. The same pattern appears for CosPlace, BoQ, and DINOv2+SALAD.

These rows are intentionally selected failure cases and should not be interpreted as a comparison establishing a universally optimal 1° threshold. In the separate descriptor diagnostic, ordinary geometry gives online ATEs within approximately 0.048–0.053 m across the nine tested appearance cues, indicating substantially lower sensitivity to which historical frames were proposed.

N.6 What These Diagnostics Establish

These experiments separate two decisions that are coupled in the default pipeline. The first is *which local prediction to retain* after the current and historical frames are processed together. The second is *whether the verified historical measurement should revise the global trajectory*. Revisit verification addresses the second decision but does not, by itself, protect the first.

The joint policy used in the main experiments allows useful historical frames to improve the current prediction, as suggested by the CO3D comparison in Table S11. However, the failure-case experiments show that an incorrect historical proposal can occasionally distort the local prediction before verification. Retaining the ordinary prediction removes this direct dependence, while gating provides an intermediate option that accepts only sufficiently small changes.

The gate used here measures rotational disagreement only. It does not guarantee small translation, scale, or dense-point error, and the tested thresholds should therefore be interpreted as diagnostics rather than as a new default policy. The main reported pose results continue to use the joint local prediction. The ordinary and gated variants are reported only to isolate the source of sensitivity.

Finally, these results do not contradict the placement property in Appendix C. That property assumes that the local prediction has already been selected and then asks what happens when its global placement T_k changes. The present section instead studies the earlier question of whether joint processing changes the local prediction itself.

O Memory Growth and Pruning

We additionally test retrieval as the memory bank grows using a separate retrieval-only diagnostic with up to 3,100 stored frames. A retrieved frame is considered a correct historical revisit when it comes from the same scene, lies within 10% of the scene's spatial extent from the query camera, and differs in orientation by at most 30°. Recall@1 is the fraction of eligible queries for which such a

revisit is ranked first, while Recall@5 measures whether one appears among the five highest-ranked retrievals. At 3,100 stored frames, scaffold retrieval achieves 0.36 Recall@1 compared with 0.25 for direct DINOv2 cosine retrieval, while Recall@5 is comparable at 0.42 versus 0.43.

The stored bank also contains substantial redundancy. A simple pose-based insertion rule reduces the bank from 3,100 to 1,119 frames while preserving at least one valid historical revisit for 157 of the original 159 eligible queries. On these retained queries, scaffold Recall@5 increases from 0.42 to 0.54. This indicates that substantial redundancy can be removed without eliminating most useful historical observations, although we do not claim an optimal pruning strategy or unbounded memory capacity.