SLOT3R: SET-ASSOCIATIVE SPATIAL MEMORY FOR STREAMING 3D RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Streaming 3D reconstruction must preserve evidence from each frame while processing an expanding scene online. Spatial memory is a natural fit because it organizes history by reconstructed 3D location. Yet Point3R uses spatial proximity both to associate a new observation with an existing memory entry and to decide whether to fuse it, conflating co-location with state identity. Because pointers summarize image patches, nearby pointers may encode distinct surfaces, viewpoints, or visibility conditions; averaging them can destroy complementary evidence before later frames disambiguate it. We argue that location should determine address, not whether observations must merge. Slot3R realizes this principle as a training-free, set-associative retrofit that lets multiple states coexist at a shared address while keeping the pretrained Point3R backbone frozen. A bounded sparse readout further decouples persistent storage from per-frame decoder access. At 300–500 sampled frames, Slot3R reduces Point3R's point-cloud accuracy error (Acc) by 57.1%–63.1% on 7Scenes and 64.0%–72.0% on NeuralRGBD, lowers Sim(3)-aligned absolute trajectory error (ATE) on all three pose benchmarks, and remains competitive on video-depth estimation. It completes all evaluated settings from 600 to 1000 sampled frames at about 19 FPS under the same protocol, whereas Point3R and InfiniteVGGT run out of memory at 800 frames and beyond.

1 INTRODUCTION

Recovering 3D scene geometry and camera motion from visual observations is a foundational capability for visual SLAM, robotic navigation, augmented reality, and embodied intelligence. Classical Structure-from-Motion and multi-view stereo pipelines rely on feature matching, triangulation, and iterative optimization (Snavely et al., 2006; Agarwal et al., 2009; Schönberger & Frahm, 2016; Furukawa & Ponce, 2010; Schönberger et al., 2016). Recent feed-forward models instead regress pointmaps, depth, and camera parameters from multiple views in a single forward pass (Wang et al., 2024; Yang et al., 2025; Wang et al., 2025c;a). Although this shift substantially reduces reconstruction time, most such models require the complete image collection before inference. Real-world visual systems instead receive observations sequentially: future frames are unavailable, each frame must be incorporated online, and the scene representation may need to expand throughout operation. Streaming 3D reconstruction must therefore preserve and organize accumulated visual evidence without repeatedly processing the full observation history.

A particularly attractive route is to anchor long-term memory directly to the reconstructed 3D scene (Wang & Agapito, 2024; Wu et al., 2025). Spatially indexed memory associates historical features with reconstructed 3D locations: new regions can allocate memory, whereas revisited regions can retrieve and update the states stored there. Compared with fixed-length recurrent states, spatial memory avoids repeatedly compressing an expanding scene into a limited token set (Wang et al., 2025b; Ren et al., 2026). Causal key–value caches instead retain history in temporal order, and their storage and attention costs grow with the stream unless tokens are compressed, selected, or discarded (Zhuo et al., 2026; Yuan et al., 2026; Zou et al., 2026). Spatially indexed memory explicitly associates evidence with the reconstructed 3D regions it describes. It therefore organizes history by scene coverage rather than frame order, making it a promising basis for long-term streaming reconstruction. A representative spatial-memory approach is Point3R(Wu et al., 2025), which maintains a persistent set of memory entries anchored to reconstructed 3D locations. As each new frame arrives, the model converts its visual evidence into spatially localized memory states and uses nearby stored entries to

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.

incorporate information from previously observed regions. This design provides a natural way to organize long-term history by scene location rather than by frame order.

However, an ambiguity arises when spatial proximity is used as a proxy for state identity. In learned spatial memory, a pointer location is a coarse geometric summary of the visual region it represents; for example, Point3R (Wu et al., 2025) derives it by averaging the predicted 3D points within an image patch. Consequently, observations of different surfaces, viewpoints, or visibility conditions may occupy nearby 3D locations while carrying distinct and complementary evidence. Merging them solely by proximity can prematurely and irreversibly collapse useful information, producing ambiguous memory states that affect subsequent predictions and updates. This motivates us to decouple spatial association from state consolidation: observations should be organized by location while remaining distinct until their content provides sufficient evidence for fusion.

We introduce Slot3R, a spatial pointer memory framework with *set-associative spatial storage*. The core idea is to use spatial proximity only to organize observations into local memory regions, while relying on their feature agreement to determine whether they should be consolidated or preserved as distinct states. Specifically, we quantize 3D space with a spherical hash, where each spatial bucket maintains up to K independent position–feature slots. For each incoming candidate, a frame-wise confidence filter first removes unreliable observations, after which feature similarity determines its memory update: a sufficiently similar and reliable candidate is fused with its matched slot, while a distinct candidate is stored separately in a free slot or replaces the least reliable one when the bucket is full. In this way, Slot3R preserves complementary evidence within the same spatial region instead of prematurely collapsing it based on spatial proximity alone.

A second challenge lies in scaling spatial memory to long streams: as the camera continuously explores new regions, more spatial buckets become occupied and the persistent memory grows with scene coverage, while each incoming frame typically requires only a small fraction of this history. This motivates us to decouple persistent storage from per-frame memory access, allowing the system to preserve an expanding scene history without exposing the entire memory to the decoder at every step. Slot3R retrieves relevant local evidence from neighboring spatial buckets using a spatial prior, and complements it with a small set of uniformly sampled global anchors for coarse nonlocal context. Only a fixed number of these selected tokens are passed to the pointer–image interaction module. As a result, Slot3R preserves long-term spatial evidence as the scene expands while keeping memory-dependent decoder cost bounded regardless of the total memory size.

Slot3R is entirely training-free and can be directly applied to a pretrained Point3R model without modifying or fine-tuning its network parameters. Slot3R sets a new state of the art on 7Scenes and remains competitive with state-of-the-art methods on NeuralRGBD. Across 300–500 sampled-frame evaluations, Slot3R reduces Point3R's Acc by 57.1%–63.1% on 7Scenes and 64.0%–72.0% on NeuralRGBD, while increasing average throughput by 14.3% and 5.6%, respectively. It also improves Sim(3)-aligned ATE on all three pose benchmarks, remains competitive on video depth, and completes all evaluated streams of up to 1000 sampled frames.

Our contributions are summarized as follows:

- We reveal that conflating spatial proximity with state identity can cause premature information collapse in streaming spatial memory.
- We introduce Slot3R, a training-free set-associative spatial memory that preserves distinct local states while selectively consolidating consistent observations.
- We develop a bounded sparse spatial readout that decouples allocated persistent storage from per-frame access cost.
- Without any additional training, Slot3R sets a new state of the art for long-sequence point-cloud reconstruction on 7Scenes, while scaling robustly to streams of up to 1000 frames.

2 Related Work

2.1 Offline Feed-Forward Visual Geometry

Feed-forward visual geometry replaces the correspondence-and-optimization chain of classical reconstruction with direct regression of geometry in a shared coordinate frame (Wang et al., 2024).

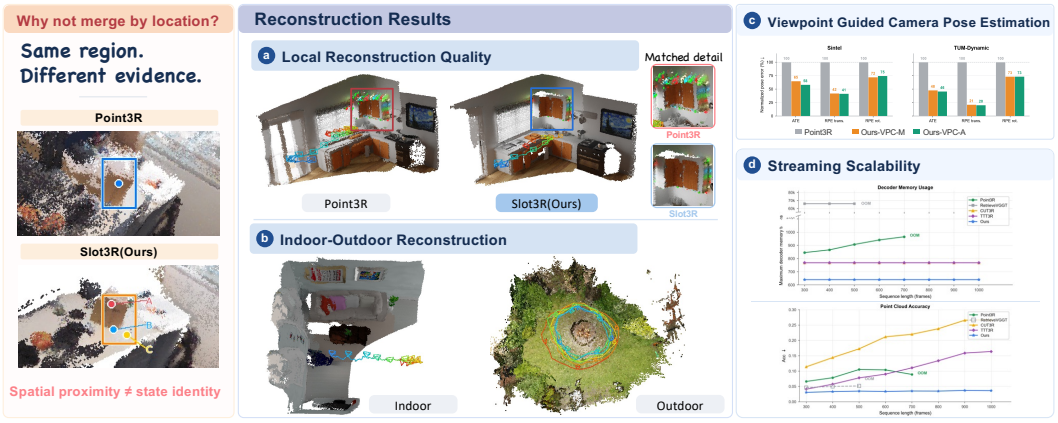


Figure 1: Reconstruction quality, viewpoint-guided pose estimation, and streaming scalability. (a) Local point-cloud comparison with Point3R. (b) Indoor and outdoor reconstructions produced by Ours. (c) Relative to Point3R, both VPC variants reduce all three pose errors on Sintel and TUM-Dynamic; Ours-VPC-A lowers ATE by 41.8%/53.1% and translation RPE by 58.6%/79.6%, respectively. (d) The 640-token readout keeps decoder-visible memory bounded: Slot3R completes 1000-frame streams with Acc ≤ 0.062, while Point3R fails with OOM at 800 frames.

Later offline models scale this up: many views in a single pass, joint prediction of cameras, depth, pointmaps, and tracks, and reference-free, permutation-equivariant architectures (Yang et al., 2025; Wang et al., 2025a;c; Chen et al., 2026c). Further work cuts the cost on large collections by chunking, partitioning, pruning tokens, and training scene states at test time (Deng et al., 2026a; Park et al., 2026; Shen et al., 2025; Xie et al., 2026; Jin et al., 2026). Similar formulations extend to dynamic and 4D scenes (Zhang et al., 2025b;a; Lin et al., 2026). Such models provide geometric priors for downstream 3D tasks. Most require full collections in offline or bidirectional modes, although ZipMap also supports sequential inference (Jin et al., 2026). Online methods instead provide immediate per-frame estimates with stream-scalable memory and computation.

2.2 Streaming Reconstruction

Streaming reconstruction preserves past observations in two memory forms. One compresses the stream into a bounded latent state, rewriting a fixed token pool across frames (Wang et al., 2025b). Per-frame cost is constant; later work refines the write rule or uses sequence models and test-time-trained states (Chen et al., 2025a; Dong et al., 2026; Ren et al., 2026; Jin & Ye, 2026; Deng et al., 2026b). LONG3R combines recurrent memory gating with adaptive 3D spatio-temporal management (Chen et al., 2025b). The second causalizes offline attention, caching earlier-frame keys and values as memory (Zhuo et al., 2026). Frame-level evidence then survives uncompressed, but the cache grows with the stream and is bounded by windows, hierarchies, or scoring, merging, evicting, and retrieving tokens under a fixed budget (Li et al., 2025; Chen et al., 2026b;a; Xu & Oishi, 2026; Liu et al., 2026; Fang et al., 2026; Wang et al., 2026; Cheng et al., 2026). Both designs struggle with long streams. Fixed capacity may overwrite early evidence in unbounded scenes (Ren et al., 2026). Bounded caches retain history using attention-, confidence-, or geometry-aware criteria (Chen et al., 2026a; Zou et al., 2026), but remain token-centric rather than maintaining distinct states at each 3D location. Geometry-informed external memory offers another route: Spann3R retrieves by feature affinity, with geometry encoded in its values (Wang & Agapito, 2024). Point3R instead indexes position–feature pointers by 3D location and averages nearby pointers into one state per region (Wu et al., 2025). Concurrent work adds viewing direction and retain-or-replace updates (Li et al., 2026). Our set-associative memory preserves multiple states per address; viewpoint-guided variants use a separate pose-conditioning bank.

3 METHOD

We introduce Slot3R, a training-free streaming 3D reconstruction method with a set-associative spatial memory that retains multiple feature states at each spatial address. We formulate the problem and review Point3R's memory in Sec. 3.1, followed by our memory organization in Sec. 3.2 and its confidence-aware update in Sec. 3.3, and finally the sparse readout with global anchors in Sec. 3.4. We describe optional viewpoint-guided pose conditioning in Sec. 3.5. Fig. 2 shows the pipeline.

3.1 PROBLEM SETUP AND PRELIMINARIES

Streaming 3D reconstruction takes an ordered RGB sequence $\mathcal{I} = \{I_t\}_{t=1}^{T}$ and predicts a camera pose T_t, a depth map D_t, and a pointmap X_t for each frame as it arrives. At time t, the model can access only the prefix $\mathcal{I}_{1:t}$. It therefore maintains a memory $\mathcal{M}_t$ that summarizes the observations processed so far. The central challenge is to organize and update $\mathcal{M}_t$ as the stream progresses.

As summarized in Fig. 2(a), our method builds on Point3R's spatial pointer memory (Wu et al., 2025). In Point3R, each incoming frame is encoded into image tokens that interact with the existing pointer memory to predict geometry and camera pose, after which the memory-aware features and predicted geometry are encoded into new spatial pointers and fused back into memory. Given the current image I_t and the previous memory $\mathcal{M}_{t-1}$, this process is written as:

$$\begin{aligned} &F_t = \mathrm{Encode}(I_t), \qquad \widetilde{F}_t = \mathrm{Read}(F_t, \mathcal{M}_{t-1}), \qquad (X_t, D_t, T_t) = \mathrm{Predict}(\widetilde{F}_t), \\ &\mathcal{C}_t = \mathrm{Pointerize}(\widetilde{F}_t, X_t, D_t, T_t), \qquad \mathcal{M}_t = \mathrm{Fuse}(\mathcal{M}_{t-1}, \mathcal{C}_t). \end{aligned} \tag{1}$$

Here F_t and $\widetilde{F}_t$ are the image and memory-aware tokens, respectively. The pointer candidates are $\mathcal{C}_t = \{(p_i, m_i)\}_{i=1}^{N_t}$, where N_t is the number of candidates, and $p_i \in \mathbb{R}^3$ and m_i denote a 3D position and its associated memory feature. However, Point3R averages spatially nearby pointer candidates, even when they may represent distinct surfaces or observations. To address this, we modify its inference-time memory to decouple spatial storage from feature fusion. Each pointer is assigned to a spatial address based on its 3D position, but this assignment alone does not determine whether it is fused with a feature state. Each address can maintain up to K distinct feature states.

3.2 SET-ASSOCIATIVE SPATIAL MEMORY

Specifically, to retain different feature states within the same spatial region, we organize the persistent memory $\mathcal{M}_t$ as a set-associative table over a discretization of 3D space, as illustrated in Fig. 2(b). An addressing function $h(\cdot)$ then maps each candidate position p_i to a discrete address $b_i := h(p_i)$. Each address indexes a bucket $\mathcal{B}_{b_i}$ containing $n_{b_i} \le K$ slots, where slot j stores $(p_{b_i,j}, m_{b_i,j}, c_{b_i,j})$, denoting its 3D position, memory feature, and confidence, respectively. In Fig. 2(b), the single blue marker for Point3R represents the memory state produced by averaging nearby pointers at the highlighted location. The differently colored markers for Slot3R represent appearance-distinct states retained in separate slots of the same spatial bucket, separating a shared spatial address from feature-state identity while preserving up to K local alternatives.

Specifically, to assign each pointer candidate to a spatial bucket and retrieve that bucket efficiently, we map its 3D position to a discrete address on a quantized spherical grid. Given $p_i = (x_i, y_i, z_i)$, we first express its position by its azimuth, elevation, and log-radius:

$$\theta_i = \mathrm{atan2}(y_i, x_i), \qquad \phi_i = \mathrm{atan2}\left(z_i, \sqrt{x_i^2 + y_i^2}\right), \qquad r_i = \log(\max(\|p_i\|_2, \rho_{\min})). \tag{2}$$

We partition these three dimensions into B_θ, B_ϕ, and B_r bins. Their bin widths are $\Delta_\theta = 2\pi/B_\theta$, $\Delta_\phi = \pi/B_\phi$, and $\Delta_r = (r_{\max} - r_{\min})/B_r$, where $r_{\min} = \log \rho_{\min}$ and $r_{\max} = \log \rho_{\max}$. We then quantize the three coordinates into integer bin indices:

$$\hat{\theta}_i = \lfloor (\theta_i + \pi)/\Delta_\theta \rfloor, \quad \hat{\phi}_i = \lfloor (\phi_i + \pi/2)/\Delta_\phi \rfloor, \quad \hat{r}_i = \lfloor (r_i - r_{\min})/\Delta_r \rfloor. \tag{3}$$

After clipping each index to its valid range, we concatenate the three indices to obtain the address:

$$b_i = h(p_i) = \mathrm{Concat}\left(\hat{\theta}_i, \hat{\phi}_i, \hat{r}_i\right). \tag{4}$$

The address b_i indexes its corresponding bucket $\mathcal{B}_{b_i}$, providing direct access to the slots associated with that spatial region during memory updates.

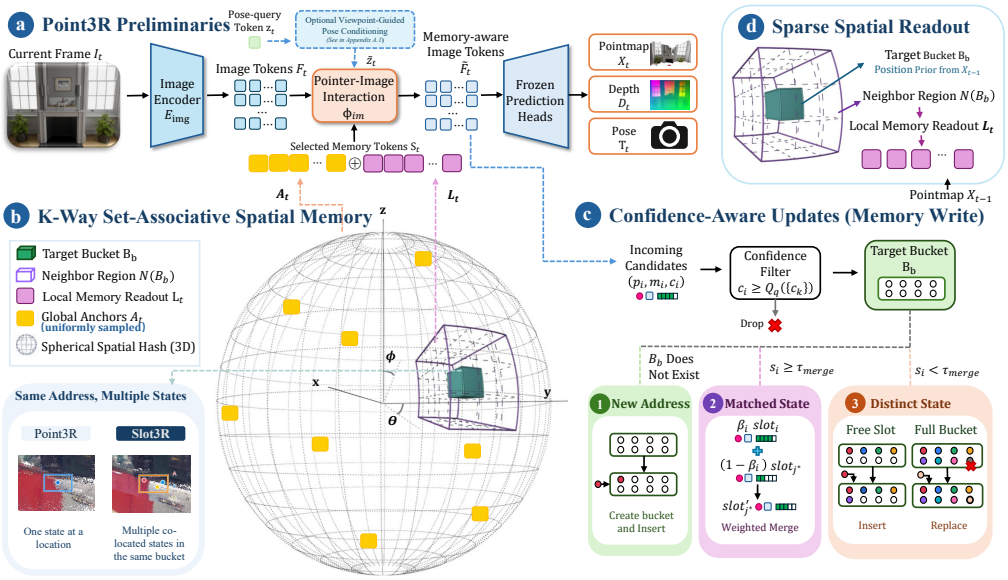


Figure 2: Overview of the core Slot3R pipeline. (a) The frozen Point3R backbone interacts image tokens with selected memory states and predicts geometry and camera pose (Sec. 3.1). (b) K-way set-associative spatial memory retains multiple states at each discrete 3D address (Sec. 3.2). (c) Confidence and feature similarity govern the memory-write path through rejection, insertion, fusion, and replacement (Sec. 3.3). (d) Previous-frame spatial priors retrieve neighboring buckets; the resulting local states are combined with uniformly sampled global anchors under a fixed readout budget (Sec. 3.4).

3.3 Confidence-Aware Memory Update

Figure 2(c) summarizes the confidence-aware write path. Prediction quality varies across patches, so we first discard candidates below the frame-wise q-quantile confidence threshold. A candidate mapped to a new bucket is inserted directly. For each candidate mapped to an existing bucket, we identify the most similar slot j^* and the least confident slot $j_{\min}$. If the maximum cosine similarity exceeds τ_{merge} and the candidate confidence is at least that of the matched slot, we update j^* through confidence-weighted fusion with coefficient β_i. If the similarity is below τ_{merge}, the candidate is admitted only if its confidence is at least that of $j_{\min}$; it then occupies a free slot or replaces $j_{\min}$ if the bucket is full. All remaining candidates are rejected. In this way, confidence determines whether memory is modified, while feature similarity determines how it is modified. Each update compares at most K slots and costs $O(K)$, independent of the total memory size. The confidence-based definition of β_i and the complete slot-level update are given in Appendix A.3 and Algorithm 1.

3.4 Sparse Spatial Readout with Global Anchors

The number of occupied memory buckets grows as the camera explores new regions, but each incoming frame observes only a small part of the scene. We therefore retrieve memory at two scales: local neighborhoods provide nearby geometric evidence, while a small set of global anchors provides coarse context from the rest of the scene. We combine the two sets and limit the number of memory tokens passed to the decoder with a fixed readout budget.

Local spatial readout. To retrieve local evidence for each image token, we first estimate where its current patch lies in the spatial memory. Following Point3R's 3D hierarchical positional encoding (Wu et al., 2025), we assign each current image token the position pooled from the same image-patch region of the previous-frame global pointmap as an approximate spatial prior. Consecutive frames typically overlap, so same-index patch regions often observe nearby scene content, although no exact correspondence is assumed. Slot3R additionally uses this prior for sparse re-

trieval because the current geometry is available only after pointer–image interaction. Let $\bar{p}_{t,i}$ denote the prior for the i-th current image token and $b_{t,i} = h(\bar{p}_{t,i})$ its spatial hash address, written as $b_{t,i} = (b^{\theta}_{t,i}, b^{\phi}_{t,i}, b^{r}_{t,i})$. Rather than query the entire table, we retrieve buckets within a neighborhood of this address. With d controlling the neighborhood extent and $\delta_\theta, \delta_\phi, \delta_r \in \mathbb{Z} \cap [-d, d]$, we define:

$$\mathcal{N}_d(b_{t,i}) = \big\{(b^{\theta}_{t,i} + \delta_\theta, b^{\phi}_{t,i} + \delta_\phi, b^{r}_{t,i} + \delta_r)\big\}. \tag{5}$$

During neighborhood expansion, out-of-range indices in all three dimensions are discarded.

For each token, we collect the position-feature slots from occupied buckets in $\mathcal{N}_d(b_{t,i})$. Concatenating slots across N_t tokens and removing duplicate references yields the frame-level local readout $\mathcal{L}_t$, as illustrated in Fig. 2(d).

Global anchors and readout budget. To retain coarse nonlocal context, we uniformly sample a small set of global anchors from the full memory, denoted by $\mathcal{A}_t$. The final readout S_t places $\mathcal{A}_t$ before the local candidates $\mathcal{L}_t$, removes duplicate references, and retains the first K_{read} entries. Thus, at most K_{read} memory tokens participate in pointer-image interaction, combining local geometric evidence with global context while bounding attention cost independently of persistent memory size.

3.5 Auxiliary Viewpoint-Guided Pose Conditioning

The set-associative memory in Secs. 3.2–3.4 is designed primarily to preserve spatial evidence for reconstruction. Camera-pose estimation, however, can additionally benefit from viewpoint-specific history. We therefore maintain an auxiliary viewpoint bank $\mathcal{R}_t$, adapted from the viewpoint-conditioned pointer representation and update of Li et al. (2026), in parallel with the core Slot3R memory $\mathcal{M}_t$. For frame t, nearby entries from $\mathcal{R}_{t-1}$ are retrieved using the same spatial priors as Sec. 3.4 and pooled with the pose-query token $\mathbf{z}_t$ to obtain a viewpoint-conditioned context $\mathbf{u}_t$. An agreement score $a_t \in [0, 1]$ suppresses unreliable retrievals. We mix this context into the pose query as:

$$\widetilde{\mathbf{z}}_t = (1 - w_t)\mathbf{z}_t + w_t\mathbf{u}_t, \qquad w_t = \begin{cases} a_t\lambda_{\mathrm{j}} g_t, & \text{Ours-VPC-M}, \\ a_t\lambda_{\mathrm{c}}, & \text{Ours-VPC-A}, \end{cases} \tag{6}$$

where $g_t \in [0, 1]$ measures recent rotational-motion change. Ours-VPC-M emphasizes the auxiliary context when rotational motion changes sharply, whereas Ours-VPC-A removes the motion-dependent gate and uses a stronger constant agreement-gated coefficient. Both variants remain training-free and use the same single decoder forward. Details are provided in Appendix A.1.

4 Experiments

4.1 Experimental Setup

Setup. We evaluate dense point-cloud reconstruction, camera trajectories, and video depth to assess accumulated spatial evidence, trajectory stability, and dense per-frame prediction, respectively. We additionally report forwarding speed across stream lengths to characterize long-sequence efficiency.

Baselines. We compare with CUT3R (Wang et al., 2025b), TTT3R (Chen et al., 2025a), StreamVGGT (Zhuo et al., 2026), STream3R (Lan et al., 2026), InfiniteVGGT (Yuan et al., 2026), RetrieveVGGT (Zou et al., 2026), Spann3R (Wang & Agapito, 2024), Point3R (Wu et al., 2025), and ZipMap (Jin et al., 2026). CUT3R and TTT3R propagate fixed-length recurrent states, while *ZipMap-Stream* adapts ZipMap's persistent scene state to sequential inference using unit-size chunks. StreamVGGT, STream3R, InfiniteVGGT, and RetrieveVGGT preserve history through causal attention and cache-based context. Spann3R and Point3R associate memory with reconstructed geometry; Point3R is our direct baseline because Ours retains its frozen backbone and changes only inference-time memory organization and readout.

Datasets. We evaluate point-cloud reconstruction on 7Scenes (Shotton et al., 2013) and Neural-RGBD (Azinović et al., 2022). Following state-of-the-art RetrieveVGGT (Zou et al., 2026), we use fixed sequence lengths of 300, 400, and 500 frames and sample every second frame ($k = 2$). Following Point3R (Wu et al., 2025), we evaluate camera pose on ScanNet (Dai et al., 2017), Sintel (Butler et al., 2012), and TUM-Dynamic (Sturm et al., 2012). For depth, we retain Bonn

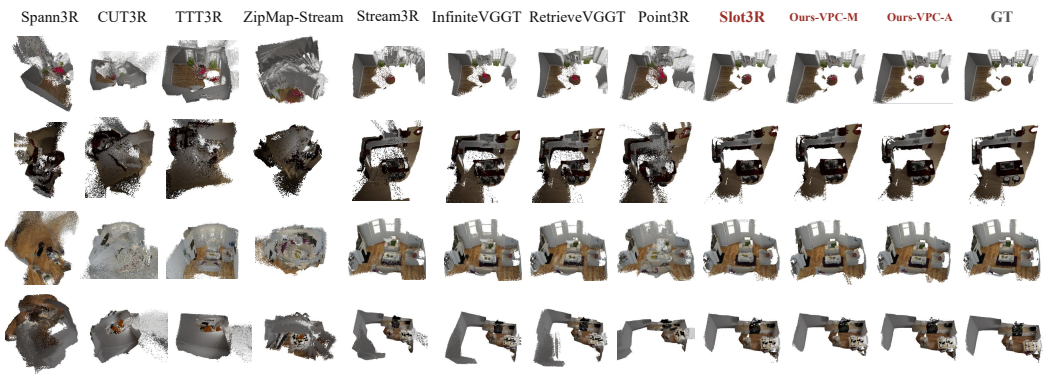


Figure 3: Qualitative point-cloud reconstruction comparison on four 500-frame NeuralRGBD sequences. Our variants recover more complete and coherent scene geometry than Point3R and more closely approach the ground truth.

Table 1: Point-cloud reconstruction on 7Scenes and NeuralRGBD with $k = 2$ at 300–500 frames. Pink, Orange, and Yellow denote the top three results per block; ties share and occupy the corresponding ranks. Δ reports the relative change from Point3R to Ours (Core), and OOM denotes failure. Avg. FPS averages completed runs. † marks reconstruction values reported by RetrieveVGGT (Zou et al., 2026); FPS is measured in our setup.

Method	Acc ↓			Comp ↓			NC ↑			Avg. FPS ↑
	300	400	500	300	400	500	300	400	500	
7Scenes										
CUT3R	0.1362	0.1697	0.1926	0.0744	0.1078	0.0894	0.5513	0.5447	0.5409	13.89
TTT3R†	0.0402	0.0502	0.0651	0.0245	0.0263	0.0295	0.5647	0.5575	0.5522	13.68
StreamVGGT	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
STream3R	0.0962	0.1013	OOM	0.0317	0.0426	OOM	0.5769	0.5744	OOM	5.82
InfiniteVGGT†	0.0432	0.0421	0.0398	0.0269	0.0269	0.0236	0.5695	0.5649	0.5617	3.63
RetrieveVGGT	0.0266	0.0276	0.0284	0.0275	0.0292	0.0294	0.5065	0.5063	0.5084	0.97
ZipMap-Stream	0.1096	0.1057	0.1021	0.2644	0.3001	0.3088	0.5223	0.5180	0.5150	5.12
Spann3R†	0.0765	0.0797	0.0707	0.0278	0.0275	0.0187	0.5464	0.5428	0.5402	41.80
Point3R	0.0634	0.0638	0.0745	0.0347	0.0308	0.0265	0.5596	0.5541	0.5510	16.77
Ours (Core)	0.0262	0.0274	0.0275	0.0154	0.0169	0.0158	0.5790	0.5789	0.5790	19.17
Ours-VPC-M	0.0257	0.0266	0.0282	0.0158	0.0164	0.0163	0.5789	0.5786	0.5792	17.54
Ours-VPC-A	0.0264	0.0269	0.0270	0.0159	0.0167	0.0163	0.5793	0.5786	0.5791	15.73
Δ Ours (Core) vs. Point3R	↓ 58.7%	↓ 57.1%	↓ 63.1%	↓ 55.6%	↓ 45.1%	↓ 40.4%	↑ 3.5%	↑ 4.5%	↑ 5.1%	↑ 14.3%
NeuralRGBD										
CUT3R	0.2260	0.3170	0.3471	0.0883	0.1245	0.1539	0.5846	0.5656	0.5539	17.51
TTT3R	0.1058	0.1425	0.1666	0.0323	0.0775	0.1027	0.6415	0.6269	0.6221	17.40
StreamVGGT	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
STream3R	0.0664	0.0828	OOM	0.0149	0.0211	OOM	0.6880	0.6935	OOM	5.81
InfiniteVGGT†	0.0532	0.0651	0.0696	0.0244	0.0348	0.0370	0.6431	0.6503	0.6433	3.63
RetrieveVGGT	0.0438	0.0456	0.0370	0.0248	0.0316	0.0397	0.6166	0.6158	0.5132	1.07
ZipMap-Stream	0.1745	0.1985	0.2141	0.2962	0.4452	0.5305	0.5288	0.5288	0.5291	5.01
Spann3R†	0.1002	0.1417	0.1829	0.0262	0.0447	0.0487	0.5953	0.5855	0.5794	35.37
Point3R	0.1132	0.1665	0.2106	0.0377	0.0557	0.0706	0.6116	0.6039	0.6025	18.01
Ours (Core)	0.0408	0.0571	0.0589	0.0151	0.0207	0.0236	0.6662	0.6743	0.6728	19.02
Ours-VPC-M	0.0403	0.0579	0.0606	0.0150	0.0207	0.0218	0.6630	0.6711	0.6692	17.57
Ours-VPC-A	0.0401	0.0561	0.0611	0.0147	0.0206	0.0242	0.6631	0.6711	0.6699	16.25
Δ Ours (Core) vs. Point3R	↓ 64.0%	↓ 65.7%	↓ 72.0%	↓ 59.9%	↓ 62.8%	↓ 66.6%	↑ 8.9%	↑ 11.7%	↑ 11.7%	↑ 5.6%

(Palazzolo et al., 2019) and KITTI (Geiger et al., 2012), following Point3R and CUT3R (Wang et al., 2025b), and evaluate ScanNet. Methods use the same frames and metric backend within each dataset block.

Table 2: Video depth estimation on ScanNet, Bonn, and KITTI under per-sequence alignment and direct metric-scale evaluation. $\delta < 1.25$ is reported as a percentage.

Method	Alignment	ScanNet		Bonn		KITTI	
		AbsRel ↓	$\delta < 1.25$ ↑	AbsRel ↓	$\delta < 1.25$ ↑	AbsRel ↓	$\delta < 1.25$ ↑
CUT3R	Per-sequence	0.056	97.1	0.074	94.4	0.111	88.3
TTT3R	Per-sequence	0.049	97.8	0.064	95.8	0.100	91.3
StreamVGGT	Per-sequence	0.268	57.1	0.059	97.2	0.173	72.2
STream3R	Per-sequence	0.037	98.9	0.058	97.5	0.077	95.2
InfiniteVGGT	Per-sequence	0.269	57.0	0.060	97.3	0.172	72.4
RetrieveVGGT	Per-sequence	0.268	57.1	0.057	97.2	0.175	71.9
ZipMap-Stream	Per-sequence	0.040	98.6	0.073	97.1	0.110	88.9
Spann3R	Per-sequence	0.176	73.7	0.231	72.3	0.336	51.9
Point3R	Per-sequence	0.053	97.1	0.066	95.8	0.093	93.5
Ours (Core)	Per-sequence	0.046	97.8	0.063	96.2	0.088	93.5
Ours-VPC-M	Per-sequence	0.046	97.8	0.063	96.3	0.088	93.3
Ours-VPC-A	Per-sequence	0.046	97.8	0.064	96.3	0.088	93.4
CUT3R	Metric-scale	0.084	94.2	0.103	88.6	0.127	84.3
TTT3R	Metric-scale	0.082	95.0	0.090	94.2	0.111	89.1
StreamVGGT	Metric-scale	0.490	11.0	0.644	0.3	0.969	0.0
STream3R	Metric-scale	0.547	1.8	0.065	96.7	0.084	95.3
InfiniteVGGT	Metric-scale	0.491	11.2	0.646	0.3	0.969	0.0
RetrieveVGGT	Metric-scale	0.491	11.0	0.643	0.3	0.969	0.0
ZipMap-Stream	Metric-scale	0.573	0.0	0.648	0.6	0.925	0.0
Spann3R	Metric-scale	0.820	0.0	0.850	0.0	0.959	0.0
Point3R	Metric-scale	0.069	95.6	0.081	95.8	0.169	80.5
Ours (Core)	Metric-scale	0.063	96.5	0.087	95.7	0.179	77.7
Ours-VPC-M	Metric-scale	0.063	96.5	0.087	95.8	0.179	77.5
Ours-VPC-A	Metric-scale	0.063	96.6	0.087	95.8	0.179	77.8

Table 3: Camera pose estimation after Sim(3) alignment on 94 ScanNet scenes, 14 Sintel sequences, and 8 TUM-Dynamic sequences.

Method	ScanNet (Static)			Sintel			TUM-Dynamic		
	ATE RMSE ↓	RPE trans ↓	RPE rot ↓	ATE RMSE ↓	RPE trans ↓	RPE rot ↓	ATE RMSE ↓	RPE trans ↓	RPE rot ↓
CUT3R	0.0960	0.0180	0.4830	0.2020	0.0560	0.5290	0.0460	0.0120	0.3750
TTT3R	0.0640	0.0170	0.4590	0.2050	0.0740	0.5930	0.0290	0.0110	0.3270
StreamVGGT	0.1607	0.0515	3.2820	0.2589	0.1296	1.6673	0.0604	0.0303	2.7908
STream3R	0.1191	0.0407	2.3151	0.3721	0.0923	1.1694	0.0294	0.0125	0.3098
InfiniteVGGT	0.1618	0.0515	3.3262	0.2589	0.1296	1.6673	0.0607	0.0305	2.7962
RetrieveVGGT	0.1624	0.0521	3.2814	0.2599	0.1297	1.6721	0.0603	0.0305	2.7915
ZipMap-Stream	0.3340	0.1010	1.7810	0.5030	0.2380	1.1570	0.1670	0.0700	1.5070
Spann3R	0.0760	0.0190	0.5090	3.2490	1.2050	2.5020	0.0720	0.0360	2.7670
Point3R	0.1120	0.0310	1.1630	0.4360	0.1360	1.5450	0.0640	0.0280	0.6610
Ours (Core)	0.0809	0.0322	1.4235	0.4243	0.1279	2.0687	0.0404	0.0260	0.7231
Ours-VPC-M	0.0817	0.0165	1.0528	0.2832	0.0574	1.1154	0.0307	0.0059	0.4843
Ours-VPC-A	0.0794	0.0162	1.1103	0.2539	0.0563	1.1529	0.0300	0.0057	0.4964

Metrics. We report accuracy (Acc), completeness (Comp), normal consistency (NC), and average forwarding speed (Avg. FPS), preferring lower Acc and Comp and higher NC and FPS. After Sim(3) alignment, pose metrics are absolute trajectory error RMSE, mean relative translation error, and mean relative rotation-angle error in degrees; relative errors use adjacent-frame pairs. Depth metrics are absolute relative error (AbsRel) and the percentage of pixels satisfying $\delta < 1.25$, under direct metric-scale evaluation and per-sequence scale-and-shift alignment.

Implementation details. Point3R-family variants use released Point3R weights without fine-tuning. Unless otherwise stated, our model uses $K = 8$ slots per bucket, a spherical hash with $(B_\theta, B_\phi, B_r) = (16, 8, 32)$ bins, radius range $[\rho_{\min}, \rho_{\max}] = [0.05, 50]$, and uniform log-radius binning, local readout radius $d = 1$, merge threshold $\tau_{\text{merge}} = 0.90$, confidence-drop quantile $q = 0.25$, $K_{\text{anchor}} = 128$ global anchors, and a readout budget of $K_{\text{read}} = 640$ memory tokens. Point3R-family runtime results use the same fused RoPE3D implementation, so FPS differences reflect memory organization and readout rather than the PyTorch RoPE3D fallback. As verified in Appendix A.7, the fused kernel changes runtime while leaving reconstruction metrics nearly unchanged. Runtime and OOM comparisons use a single NVIDIA RTX 4090 (24 GB). FPS excludes data loading and metric computation. The decoder and memory updates are causal, although the evaluation harness encodes each input sequence before recurrent decoding. OOM denotes out-of-memory failure. Ours (Core) denotes core Slot3R. Ours-VPC-M adds the auxiliary viewpoint-conditioned pose readout from Sec. 3.5 with motion gating; Ours-VPC-A uses equal interpolation endpoints, yielding agreement-only conditioning, and refreshes its auxiliary bank every frame.

4.2 Point-Cloud Reconstruction

Table 1 shows that Ours reduces Point3R's Acc by 57.1%–63.1% on 7Scenes and 64.0%–72.0% on NeuralRGBD, while improving NC throughout. The VPC variants retain comparable quality, and all three variants complete every setting; STream3R fails at 500 frames, whereas Spann3R is faster but markedly less accurate. Figure 3 shows the qualitative gains over Point3R.

4.3 Depth Estimation

Under per-sequence alignment, Ours improves Point3R's AbsRel on all three datasets while matching or improving $\delta < 1.25$ (Table 2). The VPC variants remain comparable. Metric-scale results improve on ScanNet but not Bonn or KITTI, indicating better relative depth consistency without a consistent gain in absolute-scale calibration.

4.4 Camera Pose Estimation

Ours improves Point3R's ATE on all three benchmarks but not its RPE consistently (Table 3). Ours-VPC-A reduces Point3R's ATE by 29.1%–53.1% and translation RPE by 47.8%–79.7%, while Ours-VPC-M gives comparable gains. Although CUT3R or TTT3R remains stronger on several metrics, these results support viewpoint-conditioned pose readout.

4.5 Ablation Study

Table 4: Component ablation on nine 200-frame NeuralRGBD sequences ($k = 2$). Ours uses $q = 0.25$ and a 640-token readout; Final tokens is the average sequence-end memory size.

Component	Acc ↓	Comp ↓	NC ↑	FPS ↑	Final tokens ↓
Point3R	0.086	0.032	0.635	17.943	836.0
+ Set-associative K-way memory	0.044	**0.017**	0.676	16.346	1165.2
+ Sparse local readout (640)	0.042	**0.017**	0.670	**19.411**	1162.3
+ Filtering and anchors (Ours)	**0.039**	0.018	**0.680**	19.283	**694.2**

Table 5: Spatial-prior ablation on nine 200-frame NeuralRGBD sequences ($k = 2$). The current-frame alternative adds a preliminary prediction pass; Final tokens follows Table 4.

Method	Acc ↓	Comp ↓	NC ↑	FPS ↑	Final tokens ↓
Sparse640 ($q = 0.25$)	**0.039**	**0.018**	**0.680**	**19.28**	**694.2**
+ Predicted 3D position prior	**0.039**	**0.018**	0.676	12.29	695.4

Table 4 shows that K-way memory improves reconstruction, sparse readout restores throughput, and filtering with anchors reduces storage while preserving NC. Table 5 finds comparable reconstruction but much lower FPS with a current-frame prior, favoring Point3R's previous-frame prior.

5 Conclusion

We introduced Slot3R, a training-free set-associative spatial memory that separates spatial addressing from state identity and bounds per-frame decoder access while keeping the pretrained Point3R backbone frozen. Across the main 300–500-frame evaluations, core Ours reduces Point3R's point-cloud Acc by 57.1%–72.0%, improves normal consistency, and provides 5.6%–14.3% higher average throughput; for depth, it improves per-sequence AbsRel on all three benchmarks and remains broadly comparable overall, although metric-scale results are mixed. Ours-VPC-M and Ours-VPC-A maintain comparable point-cloud and depth performance while substantially improving camera pose, with Ours-VPC-A reducing Point3R's Sim(3)-aligned ATE by 29.1%–53.1% and translation RPE by 47.8%–79.7% across the three pose benchmarks. Persistent storage still grows with scene coverage and depends on fixed memory choices; future work should study adaptive, globally budgeted memory across backbones, while our results broadly support separating location from identity and storage from per-frame access in long-lived 3D perception.

AI USE STATEMENT

In this work, we used generative AI tools, including ChatGPT and Codex by OpenAI and Claude Code by Anthropic, to polish writing, provide feedback on the research methodology and experimental presentation, and support the interpretation and communication of experimental results. We additionally used these tools to draft and edit manuscript text for clarity, organize and check LaTeX source and tables, edit and execute code under author supervision, and check internal consistency. The tools did not participate in research ideation, originate the research question, core method, or hypotheses, or generate scientific figures; nor were they used to generate datasets or experimental measurements. All AI-assisted text, code, tables, analyses, and suggestions were reviewed and, where applicable, verified by the authors against the implementation, experimental outputs, and cited sources. The authors take responsibility for the final content of this work, including all claims and artifacts produced with AI assistance.

REFERENCES

Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 72–79, 2009.

Dejan Azinović, Ricardo Martin-Brualla, Dan B. Goldman, Matthias Nießner, and Justus Thies. Neural RGB-D surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6290–6301, 2022.

Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *Proceedings of the European Conference on Computer Vision*, 2012.

Leyang Chen, Junyi Wu, Zhiteng Li, and Yulun Zhang. GHOST: Geometry-hierarchical online streaming token eviction for efficient 3d reconstruction. *arXiv preprint arXiv:2605.15852*, 2026a.

Lin-Zhuo Chen, Jian Gao, Yihang Chen, Ka Leong Cheng, Yipengjing Sun, Liangxiao Hu, Nan Xue, Xing Zhu, Yujun Shen, Yao Yao, and Yinghao Xu. Geometric context transformer for streaming 3d reconstruction. *arXiv preprint arXiv:2604.14141*, 2026b.

Tianrun Chen, Yuanqi Hu, Yidong Han, Hanjie Xu, Deyi Ji, Qi Zhu, Chunan Yu, Xin Zhang, Cheng Chen, Chaotao Ding, Ying Zang, Xuanfu Li, Jin Ma, and Lanyun Zhu. HD-VGGT: High-resolution visual geometry transformer. *arXiv preprint arXiv:2603.27222*, 2026c.

Xingyu Chen, Yue Chen, Yuliang Xiu, Andreas Geiger, and Anpei Chen. TTT3R: 3d reconstruction as test-time training. *arXiv preprint arXiv:2509.26645*, 2025a.

Zhuoguang Chen, Minghui Qin, Tianyuan Yuan, Zhe Liu, and Hang Zhao. LONG3R: Long sequence streaming 3d reconstruction. *arXiv preprint arXiv:2507.18255*, 2025b.

Chong Cheng, Peilin Tao, Nanjie Yao, Guanzhi Ding, Xianda Chen, Yuansen Du, Xiaoyang Guo, Wei Yin, Weiqiang Ren, Qian Zhang, Zhengqing Chen, and Hao Wang. HorizonStream: Long-horizon attention for streaming 3d reconstruction. *arXiv preprint arXiv:2605.23889*, 2026.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Kai Deng, Zexin Ti, Jiawei Xu, Jian Yang, and Jin Xie. VGGT-Long: Chunk it, loop it, align it, pushing VGGT's limits on kilometer-scale long RGB sequences. *arXiv preprint arXiv:2507.16443*, 2026a.

Tianchen Deng, Zhenxiang Xiong, Nailin Wang, Fangjinhua Wang, Jiuming Liu, Jianfei Yang, and Hesheng Wang. Mamba-VGGT: Persistent long-sequence video geometry grounded transformer via external sliding window mamba memory. *arXiv preprint arXiv:2605.17478*, 2026b.

Jiacheng Dong, Huan Li, Sicheng Zhou, Wenhao Hu, Weili Xu, and Yan Wang. MeMix: Writing less, remembering more for streaming 3d reconstruction. *arXiv preprint arXiv:2603.15330*, 2026.

Keyu Fang, Changchun Zhou, Yuzhe Fu, Hai Li, and Yiran Chen. IncVGGT: Incremental VGGT for memory-bounded long-range 3d reconstruction. In *Proceedings of the International Conference on Learning Representations*, 2026.

Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Haian Jin, Rundi Wu, Tianyuan Zhang, Ruiqi Gao, Jonathan T. Barron, Noah Snavely, and Aleksander Holynski. ZipMap: Linear-time stateful 3d reconstruction with test-time training. *arXiv preprint arXiv:2603.04385*, 2026.

Seonghyun Jin and Jong Chul Ye. FILT3R: Latent state adaptive kalman filter for streaming 3d reconstruction. *arXiv preprint arXiv:2603.18493*, 2026.

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

Feifei Li, Qi Song, Chi Zhang, and Rui Huang. Ray-aware pointer memory with adaptive updates for streaming 3d reconstruction. *arXiv preprint arXiv:2605.05749*, 2026.

Zizun Li, Jianjun Zhou, Yifan Wang, Haoyu Guo, Wenzheng Chang, Yang Zhou, Haoyi Zhu, Junyi Chen, Chunhua Shen, and Tong He. WinT3R: Window-based streaming reconstruction with camera token pool. *arXiv preprint arXiv:2509.05296*, 2025.

Shing Ho J. Lin, Wenzhao Zheng, Dong Zhuo, Yuqi Wu, Jie Zhou, and Jiwen Lu. SM4RT: Learning structured motion geometry for 4d reconstruction. *arXiv preprint arXiv:2607.22534*, 2026.

Xuanyi Liu, Deyi Ji, Chunan Yu, Qi Zhu, Xuanfu Li, Jin Ma, Tianrun Chen, and Lanyun Zhu. StreamCacheVGGT: Streaming visual geometry transformers with robust scoring and hybrid cache compression. *arXiv preprint arXiv:2604.15237*, 2026.

Emanuele Palazzolo, Jens Behley, Philipp Lottes, Philippe Giguère, and Cyrill Stachniss. ReFusion: 3d reconstruction in dynamic environments for RGB-D cameras exploiting residuals. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

Jinsoo Park, Donggyu Choi, Ahyun Seo, Minsu Cho, and Jeany Son. Diversity-aware view partitioning for scalable VGGT. *arXiv preprint arXiv:2607.01885*, 2026.

Kejun Ren, Lei Jin, Tianxin Huang, Lianming Xu, and Li Wang. Rethinking the state update gate for long-sequence recurrent 3d reconstruction. *arXiv preprint arXiv:2605.16981*, 2026.

Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4104–4113, 2016.

Johannes L. Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision*, pp. 501–518, 2016.

You Shen, Zhipeng Zhang, Yansong Qu, Xiawu Zheng, Jiayi Ji, Shengchuan Zhang, and Liujuan Cao. FastVGGT: Training-free acceleration of visual geometry transformer. *arXiv preprint arXiv:2509.02560*, 2025.

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*, 2013.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics*, 25(3):835–846, 2006.

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

Hengyi Wang and Lourdes Agapito. 3d reconstruction with spatial memory. *arXiv preprint arXiv:2408.16061*, 2024.

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*, 2025a.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. *arXiv preprint arXiv:2501.12387*, 2025b.

Runze Wang, Yuxuan Song, Youcheng Cai, and Ligang Liu. STAC: Plug-and-play spatio-temporal aware cache compression for streaming 3d reconstruction. *arXiv preprint arXiv:2603.20284*, 2026.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Scalable permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025c.

Yuqi Wu, Wenzhao Zheng, Jie Zhou, and Jiwen Lu. Point3R: Streaming 3d reconstruction with explicit spatial pointer memory. *arXiv preprint arXiv:2507.02863*, 2025.

Tao Xie, Peishan Yang, Yudong Jin, Yingfeng Cai, Wei Yin, Weiqiang Ren, Qian Zhang, Wei Hua, Sida Peng, Xiaoyang Guo, and Xiaowei Zhou. Scal3R: Scalable test-time training for large-scale 3d reconstruction. *arXiv preprint arXiv:2604.08542*, 2026.

Zhisong Xu and Takeshi Oishi. FrameVGGT: Frame evidence rolling memory for streaming VGGT. *arXiv preprint arXiv:2603.07690*, 2026.

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. *arXiv preprint arXiv:2501.13928*, 2025.

Shuai Yuan, Yantai Yang, Xiaotian Yang, Xupeng Zhang, Zhonghao Zhao, Lingming Zhang, and Zhipeng Zhang. InfiniteVGGT: Visual geometry grounded transformer for endless streams. *arXiv preprint arXiv:2601.02281*, 2026.

Chuhan Zhang, Guillaume Le Moing, Skanda Koppula, Ignacio Rocco, Liliane Momeni, Junyu Xie, Shuyang Sun, Rahul Sukthankar, Joëlle K. Barral, Raia Hadsell, Zoubin Ghahramani, Andrew Zisserman, Junlin Zhang, and Mehdi S. M. Sajjadi. Efficiently reconstructing dynamic scenes one D4RT at a time. *arXiv preprint arXiv:2512.08924*, 2025a.

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

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

Zichen Zou, Xiaosong Jia, Zuxuan Wu, and Yu-Gang Jiang. RetrieveVGGT: Training-free long context streaming 3d reconstruction via query-key similarity retrieval. *arXiv preprint arXiv:2605.09644*, 2026.

A Additional Ablations and Diagnostics

This appendix reports diagnostic experiments that support the design choices in the main paper. The results isolate bucket capacity, readout budget, hash parameterization, sequence length sensitivity, and the fused RoPE3D implementation used for fair runtime comparison.

A.1 Viewpoint-Guided Variant Details

This section provides the implementation details of the auxiliary viewpoint-guided pose conditioning introduced in Sec. 3.5. Figure 4 summarizes the auxiliary pathway and the two variant-specific conditioning strategies.

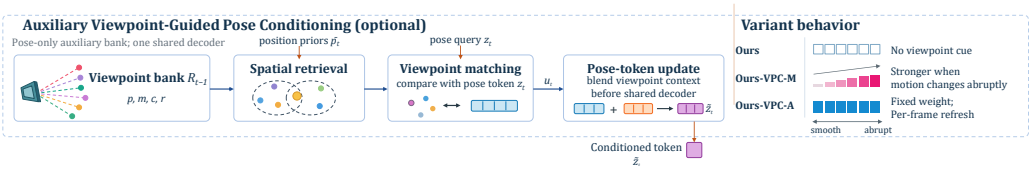


Figure 4: Auxiliary viewpoint-guided pose conditioning. Spatial priors retrieve nearby viewpoint-bank entries, whose features are pooled using the pose query to condition its input to the shared decoder. Ours-VPC-M uses motion- and agreement-gated mixing, whereas Ours-VPC-A uses agreement-only mixing with per-frame bank refresh. The auxiliary bank leaves the structure and update rule of the core Slot3R memory unchanged, and neither variant requires a second decoder forward.

The viewpoint bank $\mathcal{R}_t$ is maintained separately from the core Slot3R memory $\mathcal{M}_t$. Each entry stores a 3D position, a memory feature, a confidence value, and the viewing direction associated with the observation:

$$\mathcal{R}_t = \{(p_k^r, m_k^r, c_k^r, r_k)\}\,.$$

We adapt the direction-conditioned association and retain-or-replace update of Li et al. (2026) for this auxiliary bank. The main Slot3R memory remains on the set-associative K-way update described in Sec. 3.3 and supplies the sparse reconstruction readout of Sec. 3.4. Thus, the viewpoint bank enters through pose-token conditioning without changing the structure or update rule of the core memory. Because pose and image tokens interact in the shared decoder, this conditioning can still influence dense predictions and subsequent memory writes. Ours-VPC-M updates the auxiliary bank every four frames, whereas Ours-VPC-A refreshes it every frame.

Viewpoint-bank retrieval and pooling. Before processing frame t, we use the spatial position priors $\{\bar{p}_{t,i}\}$ from Sec. 3.4 to retrieve relevant entries from $\mathcal{R}_{t-1}$. We uniformly sample at most 192 position priors and rank the viewpoint-bank entries by their minimum Euclidean distance to these sampled positions. We retain the nearest 128 entries, denoted by $\mathcal{K}_t$.

Given the current pose-query token $\mathbf{z}_t$, we compute the cosine similarity between the query and each retrieved feature:

$$s_{t,k} = \mathrm{sim}\,(\mathbf{z}_t, m_k^r)\,,$$

and convert the similarities to attention weights using temperature τ:

$$\alpha_{t,k} = \frac{\exp(s_{t,k}/\tau)}{\sum_{\ell\in\mathcal{K}_t}\exp(s_{t,\ell}/\tau)}.$$

The viewpoint-conditioned feature is then:

$$\mathbf{u}_t = \|\mathbf{z}_t\|_2 \frac{\sum_{k\in\mathcal{K}_t}\alpha_{t,k}m_k^r}{\left\|\sum_{k\in\mathcal{K}_t}\alpha_{t,k}m_k^r\right\|_2}.$$

The normalization preserves the magnitude of the original pose-query token.

To prevent weakly related viewpoint-bank entries from perturbing the pose query, we compute an agreement factor from the maximum cosine similarity. We first rescale the maximum similarity from $[-1, 1]$ to $[0, 1]$, clip it to this interval for numerical stability, and raise it to the fourth power:

$$a_t = \left[\mathrm{clip}\left(\frac{1 + \max_{k \in \mathcal{K}_t} s_{t,k}}{2}, 0, 1\right)\right]^4.$$

We use $\tau = 0.10$ in all reported viewpoint-guided experiments.

Motion gate for Ours-VPC-M. Ours-VPC-M additionally modulates the auxiliary context according to the change in recent rotational motion. Let $\mathbf{R}_{t-3}$, $\mathbf{R}_{t-2}$, and $\mathbf{R}_{t-1}$ be the latest three estimated camera rotations available before processing frame t. We form the two relative rotations:

$$\Delta\mathbf{R}_{t-2} = \mathbf{R}_{t-3}^\top \mathbf{R}_{t-2}, \qquad \Delta\mathbf{R}_{t-1} = \mathbf{R}_{t-2}^\top \mathbf{R}_{t-1}.$$

The rotational change j_t is the angular difference between these two relative rotations. We linearly map j_t from j_{low} to j_{high} and clip the result to $[0, 1]$ to obtain the motion gate g_t. We use $j_{\mathrm{low}} = 0.010$ and $j_{\mathrm{high}} = 0.035$ radians.

The final pose-query mixing is:

$$\widetilde{\mathbf{z}}_t = (1 - w_t)\mathbf{z}_t + w_t \mathbf{u}_t.$$

For Ours-VPC-M:

$$w_t^{\mathrm{M}} = a_t \lambda_{\mathrm{j}} g_t, \qquad \lambda_{\mathrm{j}} = 0.025.$$

The auxiliary context therefore receives negligible weight under smooth rotational motion and progressively more influence as the change between consecutive relative rotations increases.

Agreement-only conditioning for Ours-VPC-A. Ours-VPC-A uses the same auxiliary bank, spatial retrieval, cosine attention, and agreement factor, but removes the motion-dependent weighting. Its mixing coefficient is:

$$w_t^{\mathrm{A}} = a_t \lambda_{\mathrm{c}}, \qquad \lambda_{\mathrm{c}} = 0.15.$$

Hence, motion magnitude does not directly affect the auxiliary residual in Ours-VPC-A; only the agreement between the pose query and retrieved viewpoint features controls its frame-dependent strength. This corresponds to setting the smooth- and high-jerk endpoints of the underlying interpolation to the same value, making the interpolation constant.

In both variants, the resulting $\widetilde{\mathbf{z}}_t$ replaces the original pose-query token before the shared decoder. The decoder memory context still comes from the core Slot3R sparse readout, and no second decoder forward is introduced.

A.2 Extended-Sequence Point-Cloud Reconstruction

Table 6 evaluates sequences of 600–1000 frames with $k = 1$, rather than the $k = 2$ sampling used in the main comparison. This stress test examines whether reconstruction quality and execution remain stable beyond the sequence lengths reported in Table 1.

All three Slot3R configurations complete the 600–1000-frame settings on both datasets. On NeuralRGBD, Ours averages 18.91 FPS; its peak GPU memory at 600, 700, 800, 900, and 1000 frames is 17.6074, 19.8434, 17.3587, 18.6159, and 20.0095 GB, respectively. The VPC variants sustain 16–17 FPS and stable NC. Point3R and InfiniteVGGT fail beyond 700 frames, while the recurrent baselines complete the streams with larger errors.

Table 6: Extended-sequence point-cloud reconstruction on 7Scenes and NeuralRGBD with k-frame sampling $k = 1$. "OOM" indicates out-of-memory failure. Pink, Orange, and Yellow cells mark the best, second-best, and third-best results within each dataset–length–metric block and within each dataset for Avg. FPS; ties occupy the corresponding ranks. † RetrieveVGGT exhausts CPU memory because its full-history KV repository grows with the input sequence.

Method	Acc ↓					Comp ↓					NC ↑					Avg. FPS ↑
	600	700	800	900	1000	600	700	800	900	1000	600	700	800	900	1000	
7Scenes																
CUT3R	0.2117	0.2200	0.2381	0.2658	0.2674	0.1263	0.1286	0.1279	0.1236	0.1010	0.5620	0.5602	0.5547	0.5502	0.5333	17.37
TTT3R	0.0901	0.1106	0.1338	0.1588	0.1638	0.0477	0.0616	0.0670	0.0728	0.0728	0.6217	0.6157	0.6075	0.6018	0.6013	17.04
StreamVGGT	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
InfiniteVGGT	0.0533	0.0545	OOM	OOM	OOM	0.0356	0.0378	OOM	OOM	OOM	0.6815	0.6772	OOM	OOM	OOM	3.44
RetrieveVGGT†	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
ZipMap-Stream	0.1321	0.1236	0.1281	0.1285	0.1233	0.2225	0.2333	0.2345	0.2470	0.2288	0.5671	0.5629	0.5602	0.5543	0.5567	17.82
Spann3R	1.0309	0.9774	0.9379	0.9294	0.9545	1.9878	1.9876	1.9716	1.9669	1.9844	0.5372	0.5246	0.5316	0.5312	0.5226	23.85
Point3R	0.1039	0.0891	OOM	OOM	OOM	0.0680	0.0432	OOM	OOM	OOM	0.6042	0.6023	OOM	OOM	OOM	17.54
Ours (Core)	0.0336	0.0351	0.0348	0.0369	0.0362	0.0231	0.0241	0.0247	0.0251	0.0245	0.6663	0.6700	0.6703	0.6739	0.6737	19.18
Ours-VPC-M	0.0340	0.0346	0.0355	0.0362	0.0364	0.0220	0.0231	0.0239	0.0241	0.0237	0.6627	0.6672	0.6655	0.6679	0.6687	17.52
Ours-VPC-A	0.0333	0.0349	0.0352	0.0362	0.0362	0.0228	0.0240	0.0248	0.0254	0.0243	0.6684	0.6704	0.6701	0.6727	0.6726	16.03
NeuralRGBD																
CUT3R	0.3547	0.3961	0.4217	0.4454	0.4627	0.2166	0.2327	0.2536	0.2590	0.2641	0.5861	0.5884	0.5776	0.5708	0.5782	17.76
TTT3R	0.1626	0.1959	0.2463	0.2793	0.2703	0.0621	0.0861	0.1071	0.1263	0.1338	0.6733	0.6523	0.6380	0.6204	0.6181	17.45
StreamVGGT	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
InfiniteVGGT	0.0592	0.0688	OOM	OOM	OOM	0.0364	0.0436	OOM	OOM	OOM	0.7750	0.7725	OOM	OOM	OOM	3.44
RetrieveVGGT†	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
ZipMap-Stream	0.1780	0.1846	0.2002	0.2150	0.2259	0.2697	0.3160	0.3635	0.3984	0.4293	0.5873	0.5822	0.5773	0.5649	0.5649	17.76
Spann3R	0.6594	0.5771	0.4579	0.4555	0.4837	2.0183	2.0187	1.9562	1.9908	2.0561	0.5307	0.5246	0.5151	0.5089	0.5025	23.99
Point3R	0.1650	0.1885	OOM	OOM	OOM	0.0831	0.1004	OOM	OOM	OOM	0.6195	0.6215	OOM	OOM	OOM	18.19
Ours (Core)	0.0406	0.0505	0.0588	0.0604	0.0618	0.0147	0.0176	0.0208	0.0244	0.0243	0.6567	0.6631	0.6684	0.6644	0.6643	18.91
Ours-VPC-M	0.0411	0.0503	0.0588	0.0605	0.0615	0.0146	0.0173	0.0206	0.0239	0.0217	0.6574	0.6616	0.6652	0.6614	0.6645	17.24
Ours-VPC-A	0.0405	0.0495	0.0586	0.0609	0.0605	0.0144	0.0172	0.0209	0.0245	0.0216	0.6575	0.6637	0.6693	0.6646	0.6657	16.07

A.3 Confidence-Aware Memory Update

Candidate filtering and bucket admission. For each $(p_i, m_i) \in \mathcal{C}_t$, we compute its confidence c_i by averaging the prediction confidence over the corresponding image patch. We retain candidates at or above a frame-wise quantile threshold:

$$\mathcal{W}_t = \{(p_i, m_i, c_i) \mid (p_i, m_i) \in \mathcal{C}_t, \quad c_i \geq \mathrm{Quantile}_q(\{c_k\}_{k=1}^{N_t})\}. \tag{7}$$

Here q is the drop quantile, and $\mathcal{W}_t$ contains the candidates retained by this frame-level filter. The address $b_i = h(p_i)$ identifies the destination bucket $\mathcal{B}_{b_i}$. A candidate whose destination bucket does not yet exist is admitted directly. For an existing bucket, admission depends on feature similarity and the confidence of the stored slots.

For a bucket, we identify its lowest-confidence slot $j_{\min}$ and the slot j^* whose feature is most similar to the candidate. Let $\mathrm{sim}(a, b) = a^\top b/(\|a\|_2\|b\|_2)$ denote cosine similarity. We then write:

$$j_{\min} = \arg\min_{1\leq j\leq n_{b_i}} c_{b_i,j}, \qquad j^* = \arg\max_{1\leq j\leq n_{b_i}} \mathrm{sim}(m_i, m_{b_i,j}), \qquad s_i = \mathrm{sim}(m_i, m_{b_i,j^*}). \tag{8}$$

Here j^* is the selected slot and s_i is its similarity to the candidate. A similar candidate with $s_i \geq \tau_{\mathrm{merge}}$ is admitted only if $c_i \geq c_{b_i,j^*}$, whereas a dissimilar candidate is admitted if $c_i \geq c_{b_i,j_{\min}}$; all other candidates are rejected. The former condition prevents a weaker observation from modifying a more reliable matched state, while the latter ensures that a distinct candidate is at least as reliable as the weakest state that it may replace.

Matched-slot fusion. For a single admitted candidate with $s_i \geq \tau_{\mathrm{merge}}$, confidence-weighted fusion is:

$$\begin{aligned} \beta_i &= \frac{c_i}{c_i + c_{b_i,j^*} + \epsilon}, \qquad p_{b_i,j^*} \leftarrow (1-\beta_i)p_{b_i,j^*} + \beta_i p_i, \\ m_{b_i,j^*} &\leftarrow (1-\beta_i)m_{b_i,j^*} + \beta_i m_i, \qquad c_{b_i,j^*} \leftarrow (1-\beta_i)c_{b_i,j^*} + \beta_i c_i. \end{aligned} \tag{9}$$

The weight β_i gives more reliable evidence greater influence on the updated position, feature, and confidence. The implementation evaluates all candidates in a frame against the same pre-update memory snapshot. If multiple candidates target the same slot, it first computes their confidence-weighted mean position and feature and their mean confidence, then applies Eq. (9) once using these aggregate quantities. The single-candidate rule above is the corresponding singleton case.

Distinct-state storage. If no matched slot is available, an admitted candidate is stored as a distinct state. Within each frame, candidates assigned to the same bucket can occupy slots that were free

in the pre-update snapshot, up to capacity K. If the snapshot bucket is full, the implementation applies at most one replacement candidate for that bucket. After grouped fusion, this candidate replaces the then-lowest-confidence slot; remaining colliding candidates are rejected. The vectorized implementation resolves excess candidates in its bucket-grouping order rather than assuming a stable input order. Each candidate compares against at most K snapshot slots.

Algorithm 1 summarizes this snapshot-based batched update.

Algorithm 1 Snapshot-based confidence-aware memory update

Require: Previous memory $\mathcal{M}_{t-1}$, retained write candidates $\mathcal{W}_t = \{(p_i, m_i, c_i)\}$, hash function $h(\cdot)$, bucket capacity K, merge threshold τ_{merge}.
Ensure: Updated memory $\mathcal{M}_t$.

Snapshot $\mathcal{M}^0 \leftarrow \mathcal{M}_{t-1}$
Initialize merge groups $\{\mathcal{G}_{b,j}\}$ and distinct groups $\{\mathcal{D}_b\}$
for all $(p_i, m_i, c_i) \in \mathcal{W}_t$ **in parallel do**
    $b_i \leftarrow h(p_i)$; read $n^0_{b_i}$ and slots from $\mathcal{M}^0$
    if $n^0_{b_i} = 0$ **then**
        Append i to $\mathcal{D}_{b_i}$
    else
        Compute $j_{\min}$, j^*, and s_i from $\mathcal{M}^0$ using Eq. (8)
        if $s_i \geq \tau_{\text{merge}}$ and $c_i \geq c^0_{b_i,j^*}$ **then**
            Append i to $\mathcal{G}_{b_i,j^*}$
        else if $s_i < \tau_{\text{merge}}$ and $c_i \geq c^0_{b_i,j_{\min}}$ **then**
            Append i to $\mathcal{D}_{b_i}$
        else
            Drop i
        end if
    end if
end for
for all nonempty merge groups $\mathcal{G}_{b,j}$ **do**
    Compute confidence-weighted $\bar{p}_{b,j}$, $\bar{m}_{b,j}$, and mean confidence $\bar{c}_{b,j}$
    Fuse $(\bar{p}_{b,j}, \bar{m}_{b,j}, \bar{c}_{b,j})$ once into snapshot slot (b, j) using Eq. (9)
end for
for all nonempty distinct groups $\mathcal{D}_b$ **do**
    if $n^0_b < K$ **then**
        Insert up to $K - n^0_b$ candidates selected by the vectorized bucket grouping into snapshot-free slots
    else
        Select one candidate from $\mathcal{D}_b$ by the same grouping order
        Replace the lowest-confidence slot of bucket b after grouped fusion with the selected candidate
    end if
end for
return $\mathcal{M}_t$

Table 7 favors Drop-then-Compare, particularly at 400–500 frames. Global post-write pruning degrades more sharply because it reduces spatial coverage.

Drop-quantile sensitivity. Tables 8 and 9 show opposing task trends. Stronger filtering generally benefits point-cloud accuracy and memory: at $q = 0.25$, NC is highest, memory is 23.5% smaller, and FPS is 7.1% higher than at $q = 0$. Depth favors weaker filtering, and $q = 0.25$ outperforms $q = 0.35$ on nine of twelve metrics. We therefore use $q = 0.25$ as a cross-task compromise between reconstruction, memory, and depth.

Table 7: Ablation of the ordering and scope of confidence-guided memory filtering on NeuralRGBD with $k = 2$ and $n = 9$ sequences. All variants retain the same K-way memory, confidence-aware fusion, and 640-token sparse readout. C–D and D–C denote Compare-then-Drop and Drop-then-Compare, respectively. Avg. FPS is computed over the matched 200–500-frame runs. This diagnostic sweep was run separately from the main comparison in Table 1. OOM denotes failure on at least one sequence.

Filtering policy	Acc ↓					Comp ↓					NC ↑					Avg. FPS ↑
	200	300	400	500	1000	200	300	400	500	1000	200	300	400	500	1000	
C–D, bucket 25%	0.0440	0.0450	0.0690	0.0686	OOM	0.0180	0.0150	0.0227	0.0204	OOM	0.6745	0.6620	0.6692	0.6673	OOM	**19.52**
C–D, bucket 12.5%	0.0390	0.0460	0.0672	0.0708	OOM	0.0180	0.0160	0.0227	0.0217	OOM	0.6810	0.6675	0.6733	0.6695	OOM	18.99
C–D, global 10%	0.0420	0.0620	0.1109	0.0861	OOM	0.0180	0.0190	0.0362	0.0250	OOM	0.6670	0.6490	0.6545	0.6554	OOM	19.26
C–D, global 15%	0.0530	0.1430	0.1753	0.1242	OOM	0.0200	0.0290	0.0660	0.0362	OOM	0.6625	0.6435	0.6446	0.6469	OOM	19.33
No dropping	0.0390	0.0450	0.0617	0.0650	OOM	0.0180	0.0160	0.0214	0.0203	OOM	0.6795	0.6635	0.6737	0.6736	OOM	18.91
D–C, 25% (Ours)	**0.0389**	**0.0408**	**0.0571**	**0.0589**	**0.0631**	**0.0178**	0.0151	**0.0207**	0.0236	**0.0239**	0.6801	0.6664	**0.6743**	0.6729	**0.6758**	19.02

Table 8: A separate sensitivity sweep of the pre-write drop quantile q on 500-frame NeuralRGBD reconstruction with $n = 9$ sequences. All variants within this sweep use the same K-way memory and 640-token sparse readout.

q	Acc ↓	Comp ↓	NC ↑	FPS ↑	Final tokens ↓
0.00	0.0649	0.0227	0.6713	17.949	1367.3
0.05	0.0637	0.0228	0.6678	19.140	1291.4
0.10	0.0621	**0.0211**	0.6713	19.230	1210.9
0.15	0.0624	0.0220	0.6698	19.141	1163.4
0.20	0.0620	0.0220	0.6702	19.253	1101.4
0.25	0.0589	0.0236	**0.6729**	19.216	1046.2
0.30	0.0597	0.0228	0.6708	19.209	996.2
0.35	**0.0588**	0.0231	0.6673	**19.359**	**937.1**

Table 9: Depth sensitivity to the pre-write drop quantile q on ScanNet, Bonn, and KITTI under metric-scale and per-sequence alignment. These results are from a diagnostic sweep separate from the main depth evaluation in Table 2. $\delta < 1.25$ is reported as a percentage.

Alignment	q	ScanNet		Bonn		KITTI	
		AbsRel ↓	$\delta < 1.25$ ↑	AbsRel ↓	$\delta < 1.25$ ↑	AbsRel ↓	$\delta < 1.25$ ↑
Metric-scale	0.00	0.06219	96.54	**0.08453**	**96.08**	0.18206	77.19
Metric-scale	0.05	0.06225	96.53	0.08534	95.98	0.18179	77.07
Metric-scale	0.10	**0.06179**	**96.58**	0.08514	96.04	0.18216	77.33
Metric-scale	0.15	0.06251	96.45	0.08521	95.97	0.18135	77.11
Metric-scale	0.20	0.06230	96.43	0.08547	96.01	0.17915	**77.72**
Metric-scale	**0.25**	0.06317	96.43	0.08652	95.86	0.17966	77.37
Metric-scale	0.30	0.06290	96.44	0.08727	95.78	0.17871	77.58
Metric-scale	0.35	0.06349	96.24	0.08730	95.83	**0.17833**	77.47
Per-sequence	0.00	**0.04455**	**97.84**	0.06379	96.38	**0.08708**	93.39
Per-sequence	0.05	0.04495	97.80	0.06364	96.43	0.08793	93.35
Per-sequence	0.10	0.04465	97.83	0.06350	96.40	0.08738	93.42
Per-sequence	0.15	0.04533	97.74	**0.06297**	**96.44**	0.08806	93.28
Per-sequence	0.20	0.04554	97.77	0.06298	96.43	0.08832	93.23
Per-sequence	**0.25**	0.04622	97.72	0.06324	96.37	0.08789	**93.44**
Per-sequence	0.30	0.04585	97.80	0.06360	96.39	0.08772	93.30
Per-sequence	0.35	0.04675	97.76	0.06355	96.36	0.08815	93.31

A.4 Bucket Capacity

The bucket capacity K denotes the maximum number of distinct states retained at one spatial address. Tables 10 and 11 vary only K under two 7Scenes settings. In Table 10, increasing K from 1 to 2 reduces Acc and Comp by 66.7% and 66.5%, respectively, showing that a single state is insufficient when local ambiguity is strong. The trend is smoother when sampling every 20th frame ($k = 20$), where Point3R is already competitive; nevertheless, $K = 8$ improves its Acc and Comp by 9.0% and 12.7% while its NC differs by only 0.0009. Across both settings, $K = 1$ is weakest and gains saturate by $K = 8$. With $k = 20$, $K = 16$ uses 67.1% more final tokens than $K = 8$ without improving reconstruction. Although $K = 16$ gives the best reconstruction metrics in Table 10, it uses 63.8% more final tokens than $K = 8$ and is slower. We therefore use $K = 8$ as the default balance between quality, local-state capacity, and memory cost.

Table 10: Bucket-capacity ablation on 7Scenes with $k = 2$ and 200 sampled frames, where K is the maximum number of slots per spatial bucket. Avg. final token count is averaged across evaluated sequences.

Method	K	Acc ↓	Comp ↓	NC ↑	FPS ↑	Avg. final tokens ↓
Point3R	–	0.047003	0.018037	0.578354	17.746	918.0
Ours (Core)	1	0.089329	0.053222	0.569152	19.911	**219.2**
Ours (Core)	2	0.029775	0.017823	0.580439	**19.935**	250.9
Ours (Core)	4	0.028163	0.016960	0.583035	19.816	440.4
Ours (Core)	**8**	0.026758	0.017059	0.584917	19.405	754.3
Ours (Core)	16	**0.026092**	**0.016093**	**0.585791**	18.958	1235.2

Table 11: Bucket-capacity ablation on 7Scenes when sampling every 20th frame ($k = 20$). Avg. final token count is averaged across evaluated sequences.

Method	K	Acc ↓	Comp ↓	NC ↑	FPS ↑	Avg. final tokens ↓
Point3R	–	0.034096	0.024413	**0.649663**	13.618	1406.9
Ours (Core)	1	0.056078	0.039312	0.636682	16.914	**168.3**
Ours (Core)	2	0.042358	0.029619	0.641545	16.199	271.4
Ours (Core)	4	0.034110	0.023624	0.646727	**17.047**	482.4
Ours (Core)	**8**	**0.031013**	**0.021317**	0.648767	16.459	847.0
Ours (Core)	16	0.031309	0.021690	0.648118	16.513	1415.4

A.5 Readout Budget Sensitivity

The readout budget controls the number of memory tokens exposed to the decoder, not the size of the stored memory. Tables 12 and 13 show that performance saturates at moderate budgets. On NeuralRGBD point-cloud reconstruction, increasing the budget from 128 to 512 improves Acc from 0.043 to 0.039 and NC from 0.653 to 0.680, indicating that very small readouts omit useful spatial evidence. Beyond 512 tokens, Acc remains within 0.038–0.039 and NC within 0.677–0.681, with no monotonic gain from exposing more tokens. Depth accuracy similarly plateaus around 640–768 tokens: 768 is numerically best, but differs from 640 by only 0.00003 AbsRel and 0.00030 in $\delta < 1.25$. We therefore use $K_{\text{read}} = 640$ as a balanced budget across reconstruction and depth.

Table 12: Sparse-readout budget sensitivity on NeuralRGBD point-cloud reconstruction with $k = 2$ and 200 sampled frames. This diagnostic sweep was run separately from the component ablation in Table 4.

Sparse budget	Acc ↓	Comp ↓	NC ↑	FPS ↑
128	0.043	**0.017**	0.653	20.000
256	0.041	**0.017**	0.665	20.068
384	0.040	**0.017**	0.677	**20.312**
512	0.039	0.018	0.680	19.756
640	**0.038**	0.018	0.677	19.114
768	**0.038**	**0.017**	0.678	18.925
912	**0.038**	**0.017**	**0.681**	19.259
1024	0.039	0.018	0.680	19.362

Table 13: Sparse-readout budget sensitivity for depth estimation.

Sparse budget	AbsRel ↓	$\delta < 1.25$ ↑
128	0.06508	0.96185
256	0.06427	0.96323
384	0.06453	0.96245
512	0.06405	0.96303
640	0.06326	0.96399
768	**0.06323**	**0.96429**
912	0.06392	0.96199
1024	0.06390	0.96175

A.6 Spherical Hash versus Voxel Hash

The hash parameterization determines which observations share a bucket and which buckets are treated as neighbors during readout. We compare the proposed azimuth–elevation–log-radius grid with a voxel-style Cartesian quantization under the same sparse-512 setting on seven diagnostic sequences. As shown in Table 14, the spherical hash reduces Acc from 0.1409 to 0.0491 and Comp from 0.0564 to 0.0280, while increasing NC from 0.6076 to 0.7451. This result indicates that the addressing geometry, rather than hash lookup alone, materially affects which local states are grouped and retrieved. We therefore retain the spherical parameterization in the main method, while treating alternative grids as an implementation choice rather than a separate contribution.

Table 14: Hash parameterization ablation for sparse-512 readout.

Hash	n	Acc ↓	Comp ↓	NC ↑	Acc med. ↓
Spherical	7	0.0491	0.0280	0.7451	0.0318
Voxel	7	0.1409	0.0564	0.6076	0.0991

A.7 Fused RoPE3D Runtime Control

Runtime comparisons are sensitive to the RoPE3D implementation. Table 15 compares Point3R before and after enabling the fused RoPE3D kernel under the NeuralRGBD diagnostic protocol. The fused kernel raises Point3R throughput from 4.547 to 14.750 FPS while leaving reconstruction metrics and memory essentially unchanged. Under the same fused implementation, the 640-token sparse readout raises throughput from 16.658 to 18.269 FPS relative to full-memory readout, while NC remains 0.753. This bounded readout uses up to 128 uniformly sampled global anchors and fills the remaining 640-token budget with up to 512 additional local tokens after deduplication. These controls separate acceleration due to fused RoPE3D from that due to the local-and-global bounded readout. We therefore use fused RoPE3D for all Point3R-family runtime comparisons and do not count it as part of our method.

Table 15: Effect of fused RoPE3D and sparse readout on NeuralRGBD diagnostic runs with approximately 50-frame sequences. "Ours (Core) w/o sparse readout" uses K-way memory and pre-write filtering with $q = 0.25$, but exposes the full stored memory to the decoder; Ours (Core) instead applies a bounded readout of at most 640 tokens, comprising up to 128 uniformly sampled global anchors and up to 512 additional local tokens after deduplication.

Method	Acc ↓	Comp ↓	NC ↑	FPS ↑	Avg. stored tokens ↓
Point3R w/o fused RoPE3D	0.084	0.030	0.753	4.547	1571.0
Point3R w/ fused RoPE3D	0.083	0.030	0.752	14.750	1572.6
Ours (Core) w/o sparse readout	**0.061**	**0.027**	**0.753**	16.658	**1204.1**
Ours (Core)	0.063	0.029	**0.753**	**18.269**	1220.1