Solv3R: 3D Looped Transformer as a Multi-view Solver

Jeong-gi Kwak[1] Andrea Tagliasacchi[2,3,4] Sho Kagami[5] Yuki Ono[6] Kwang Moo Yi[1]

[1]University of British Columbia [2]Wayve Labs [3]Simon Fraser University
[4]University of Toronto [5]Sony Semiconductor Solutions Corporation [6]Sony Corporation

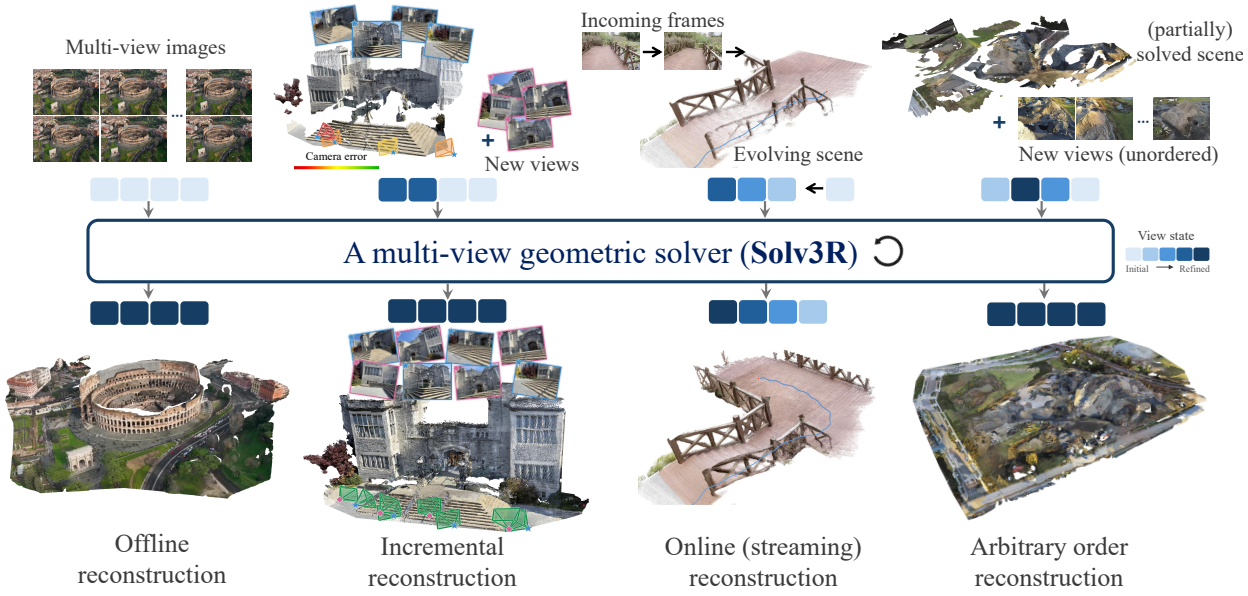


Figure 1. **Teaser.** We present Solv3R, a looped transformer that learns to solve multi-view geometry from its current state. Diverse reconstruction settings, *e.g.*, offline, incremental, online, and out-of-order reconstruction, can all be handled by our single learned solver.

Abstract

Classical structure-from-motion reconstruction proceeds iteratively, refining camera poses and scene geometry from a current estimate. This allows maps to easily expand without recomputing already computed views, or geometry estimates to improve as more observations are provided. Most geometric foundation models, however, predict entire geometry in a single forward pass or update a recurrent state as frames arrive sequentially, making this expansion and refinement non-trivial. We introduce Solv3R, a looped transformer that learns to continue solving multi-view geometry from its current state. Importantly, its update operator is independent of the iteration count, allowing previously refined and newly introduced views to be jointly updated regardless of how "solved" they are. We train the operator to preserve accurate reconstructions under continued iteration and correct inaccurate states, enabling refinement of fixed image collections and integration of new observations. Representations of solved views can also be cached once and reused as geometric context across iterations, enabling compact and efficient key-value caching. We analyze the solver dynamics and compare it with the state-of-the-art, demonstrating that a single learned solver supports offline reconstruction, incremental reconstruction, relocalization, and streaming without task-specific training or modules. The code and models will be made publicly available at https://ubc-vision.github.io/solv3r/.

1. Introduction

Recent geometric foundation models (GFMs) [47, 51, 88, 89, 93, 95, 101] have driven a shift in multi-view 3D

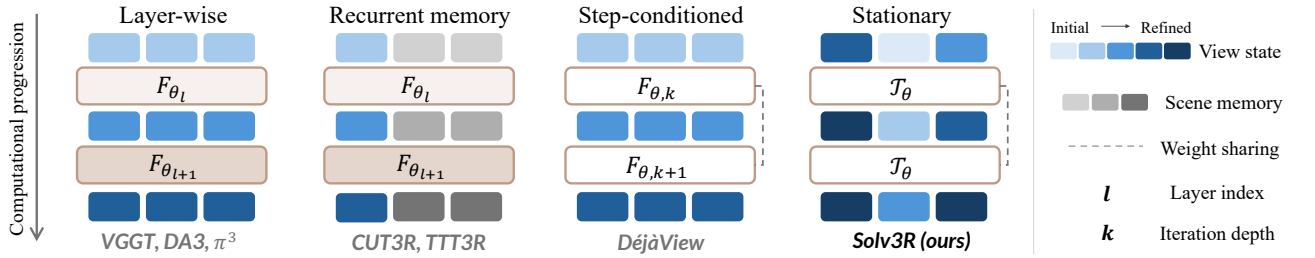


Figure 2. Comparison of geometric update mechanisms. Conventional GFMs [51, 88, 95] use successive layer-specific transformations, tying their intermediate representations to layer depth. Recurrent-memory models [21, 90] process incoming frames while updating a persistent scene memory across observations. Although DéjàView [12] shares transformer weights, its updates remain conditioned on the refinement step. Solv3R applies the same stationary update operator regardless of the iteration count, allowing previously refined and newly introduced views to be jointly updated.

reconstruction toward predicting scene geometry directly from multi-view images. Unlike classical Structure-from-Motion (SfM), which relies on correspondence matching and *iterative* geometric optimization [69], these models estimate geometry in a *feed-forward* manner. They typically employ successive transformer blocks to progressively build geometry-aware representations and predict camera poses, depths, and point maps.

But large-scale reconstruction often extends beyond a single forward pass. The image collection may exceed a transformer's input capacity, or new views may arrive after the initial reconstruction, such as subsequent video frames or additional images and videos of the same scene. Incorporating these views with a feed-forward GFM typically requires processing the combined image set again [88, 95]. Streaming models [21, 45, 86, 90, 117] avoid this by updating a state frame by frame, but they are limited to views that arrive *sequentially*. In contrast, classic SfM incorporates new observations into an existing reconstruction and uses *iterative* optimization to refine its estimates [69]. Hence, a learned reconstruction model should likewise be able to *continue* solving from its current state, refining existing estimates and integrating new views as they arrive. This requires a reconstruction "state" that remains usable for further geometric computation beyond the initial forward pass.

But conventional feed-forward GFMs like VGGT [88] process image representations through successive transformer layers, each applying a *different* learned transformation (Fig. 2). Their intermediate states (i.e. tokens) are consequently tied to particular *stages* of computation (i.e. layer depth). DéjàView [12] moves toward iterative reconstruction by repeatedly applying a shared transformer block. However, its architecture remains conditioned on the refinement step, allowing the same block to perform *different* transformations at different stages. In contrast, a classical *stationary* iterative solver such as Jacobi [10], applies the same update at every iteration, and *does not* need to know which iteration it is operating on. It can thus continue from any state, including one whose variables have received different numbers of iterations. A key observation is that sharing weights alone *does not* give DéjàView this ability.

We thus posit that a GFM *should be a learning-based multi-view solver*. Given more computation, a solver should continue refining a reconstruction until convergence, and once converged, the solution should not change with further iterations. A solver of this type should handle offline reconstruction, adding views, and streaming with one unifying architecture (see Fig. 1).

To this end, we introduce Solv3R, an architecture that reformulates multi-view reconstruction as a reusable learning-based geometric solver. Solv3R employs a stationary (*i.e.* timestep invariant) learned geometric update operator that acts on the current multi-view state. We train this operator so that a solved reconstruction stays unchanged under further iterations, and so that a reconstruction with errors, for example one with newly added views, converges to the solution (Fig. 3).

With these solver properties, Solv3R *unifies* different reconstruction settings through a single learned geometric solving process. Our solver achieves competitive accuracy in conventional multi-view reconstruction, while allowing existing estimates to be jointly revised as new observations arrive. This reuse extends beyond reconstruction states to attention context: a fixed reconstructed map can provide a single KV cache reused across solver calls. Together, these capabilities support incremental reconstruction, map-based relocalization, and streaming reconstruction without task-specific learned modules, achieving competitive streaming performance even without streaming-specific training.

Through extensive analyses of solver dynamics and experiments across reconstruction scenarios, we demonstrate how reusable geometric computation enables flexible and efficient multi-view reconstruction.

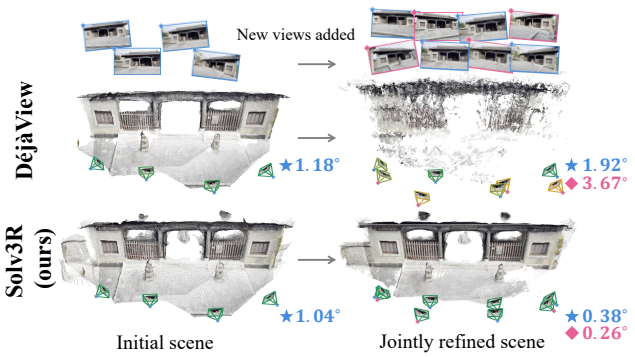


Figure 3. Continuing geometric solving with new observations. When new views are introduced, Solv3R continues solving from its existing state, jointly refining both previously reconstructed and newly added views. DéjàView struggles to maintain consistent reconstruction. Numbers indicate camera rotation errors in degrees.

2. Related work

Geometric foundation models (GFMs) have advanced feed-forward 3D reconstruction, progressing from pairwise pointmap regression and matching [47, 93] to joint prediction of camera poses and dense geometry from multiple views [15, 28, 51, 75, 88, 95, 101]. Recent methods further improve multi-view geometric prediction and scalability [19, 20, 30–32, 63, 87, 89, 99, 114]. Related advances extend geometric prediction to monocular inputs [35, 91, 92, 102], incorporate additional geometric priors [38, 42, 56, 58, 65], or produce renderable Gaussian representations [22, 100, 105, 106].

Learned iterative refinement and solvers. Learned iterative refinement has been widely applied to optical flow, stereo, multi-view depth, and visual SLAM [8, 41, 53, 77, 78, 85, 112]. Related work on equilibrium and recurrent models investigates fixed-point behavior, stability, and extended computation [1, 7, 9, 29]. Inspired by recent advances in looped transformers for language models [5, 6, 24, 34, 39, 61, 103, 115, 116], DéjàView [12] introduces shared recurrent computation for multi-view reconstruction. We extend this direction to continued geometric solving across fixed and evolving view sets, including views at different refinement stages.

Continuous and streaming 3D reconstruction. Continuous reconstruction methods maintain and update geometric information as new observations arrive. Some approaches employ persistent scene representations or test-time adaptation [21, 86, 90, 98, 113], while others preserve geometric context through causal attention and caching [18, 45, 108, 117]. Window-based methods further integrate local reconstructions through registration and optimization [26, 50, 54, 55, 59, 62, 109, 111]. Rather than introducing a separate learned reconstruction process for streaming, Solv3R applies the same geometric solver to both offline and online reconstruction.

3. Method

Our key idea is to make the shared block of a looped transformer act as the "iteration" of a classical solver, whose update logic does not depend on how many iterations have already run. We review looped transformers for 3D reconstruction in Sec. 3.1, and use a simple classical solver in Sec. 3.2 to define the properties that we want from such a solver. We then replace the classical iteration with a learned operator in Sec. 3.3, and discuss its properties in Sec. 3.4.

3.1. Looped transformers for 3D reconstruction

DéjàView [12] formulates multi-view reconstruction as iterative refinement of a latent state. Given input images $\mathcal{I}=\{I_i\}_{i=1}^{N}$, an image encoder initializes a multi-view state $\mathbf{z}^0$, which is repeatedly updated by a shared transformer block with frame-wise and global attention. The refined state is decoded into depth and ray maps, from which scene geometry and camera poses are recovered. Unlike conventional feed-forward transformers with independently parameterized layers, DéjàView *shares* the transformer block across iterations, but conditions each update on the step:

$$\mathbf{z}^{k+1} = F_\theta(\mathbf{z}^k; \tau_k), \tag{1}$$

where τ_k is the conditioning on the iteration index k, which lets the shared block apply a *different* transformation at each iteration. Because of this structure, DéjàView loses reconstruction accuracy when iterated beyond its trained range, as we demonstrate in Section 4. More fundamentally, each state is tied to the iteration index at which it was computed. One application of F_θ thus cannot jointly update views that have received different numbers of iterations, because no single τ_k matches all of them. Thus, sharing weights alone *does not* give the block an update that is independent of the iteration index.

3.2. An iterative solver as a running example

To make the properties we want concrete, consider a classical constraint satisfaction problem, solving a linear system $A\mathbf{x}=\mathbf{b}$ with the Jacobi method [10]. The rows of matrix A and corresponding observations $\mathbf{b}$ define the constraints, and the state $\mathbf{x}$ holds the current estimate of the unknowns. Starting from an initial state $\mathbf{x}^0$, each iteration applies the same update:

$$\mathbf{x}^{k+1} = \Phi(\mathbf{x}^k; A, \mathbf{b}), \tag{2}$$

We call one iteration, that is, one application of Φ, a solver call. The update Φ does not depend on the iteration index k, which makes the Jacobi *stationary* [10]. When the method converges, three *properties* follow:

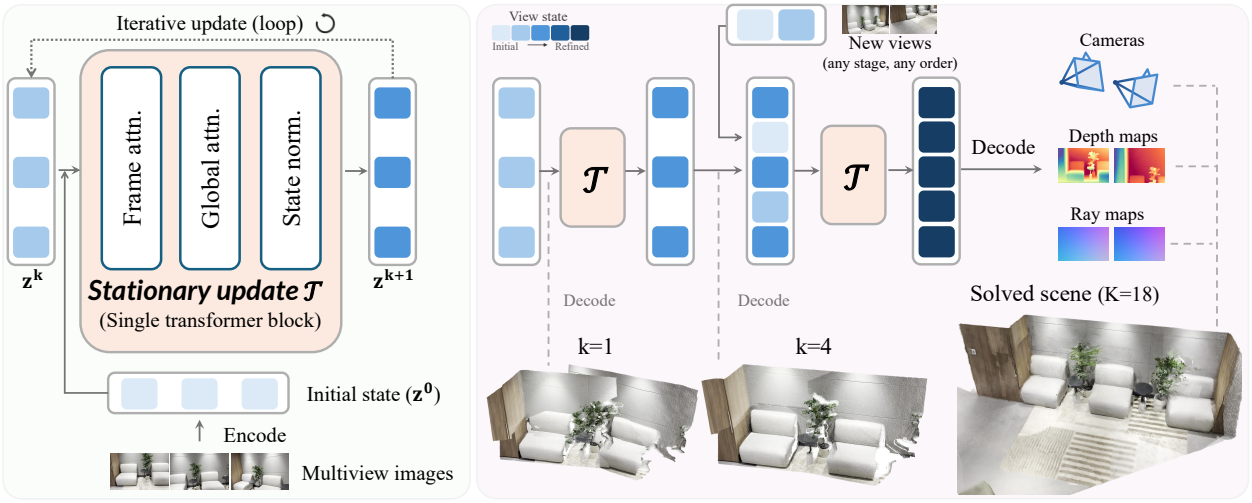


Figure 4. **Overview.** Left: A stationary transformer block repeatedly updates the multi-view state through frame-wise and global attention, initial-state injection, and state normalization. Right: The same operator continues refining the reconstruction across solver calls, allowing new views to be introduced at any stage and jointly updated with existing views. Camera poses, depth maps, and ray maps can be decoded from the current state after any solver call.

(P1) *Stability.* The accurate solution $\mathbf{x}^*$, which satisfies $A\mathbf{x}^*{=}\mathbf{b}$, is a fixed point with $\Phi(\mathbf{x}^*; A, \mathbf{b}){=}\mathbf{x}^*$, so further solver calls leave it unchanged.
(P2) *Correction.* Consider a perturbed state, that is, a solution moved off by, for example, added noise. From such a state, the same solver calls converge back to $\mathbf{x}^*$.
(P3) *Continuation.* When new unknowns and constraints are added, the solver can start the old unknowns from their solved values, and the new ones from an initial guess. It then updates all of them together toward the solution of the enlarged system, often in fewer solver calls than a restart from $\mathbf{x}^0$. The state holds values that have received different numbers of solver calls, and Φ still updates them in one solver call. It can do so because it does not depend on how a state was reached.

Multi-view reconstruction has the same structure. The input images define the constraints, the per-view states hold the current estimate, and the cameras and depths decoded from the state are the current estimate of the solution. A solved state is one whose decoded cameras and depths are accurate, and which further solver calls change little. A learned multi-view solver should thus leave a solved reconstruction unchanged under further solver calls (P1), correct a perturbed reconstruction (P2), and continue from a solved reconstruction when views are added (P3). Meanwhile, the update of DéjàView in (1) depends on τ_k, so it *does not* have these properties by construction.

3.3. A stationary geometric solver

We replace the update Φ of (2) with a learned operator $\mathcal{T}_\theta$, and the constraints $(A, \mathbf{b})$ with the encoded input images $\mathbf{z}^0{=}E(\mathcal{I})$, as illustrated in Figure 4:

$$\mathbf{z}^{k+1} = \mathcal{T}_\theta(\mathbf{z}^k; \mathbf{z}^0), \tag{3}$$

where the state also starts from the *initial* encoding $\mathbf{z}^0$. Like Φ, the operator is *stationary* (i.e. time independent). It needs neither the iteration index nor the number of solver calls that each view has received. The state may thus combine views with different numbers of solver calls, $\mathbf{z}^k{=}(\mathbf{z}_1^{[h_1]}, \ldots, \mathbf{z}_N^{[h_N]})$, where view i has received h_i calls. We inject $\mathbf{z}^0$ at every solver call for the same reason that Φ reads A and $\mathbf{b}$ at every call. The state tracks the current estimate, while $\mathbf{z}^0$ carries the constraints from the images. Attention infers how the views constrain each other. Without $\mathbf{z}^0$, each call would see only the current state, and errors in the state could accumulate over calls. We instantiate $\mathcal{T}_\theta$ as a *single* transformer block that is completely *shared* across solver calls, with frame-wise and global attention.

We fine-tune this block from the pretrained DéjàView [12], and make its update stationary with two changes. First, DéjàView scales the channels of each sub-block with gates computed from the conditioning τ_k on the iteration index. We replace these gates with learned *constant* gates, which apply the same channel-wise scaling at every solver call. Second, to keep the magnitude of the state from drifting over many solver calls, we rescale the updated state to a fixed root-mean-square (RMS) norm. We use a single scale factor for the whole state, which preserves the relative magnitudes across views. Stationarity alone does not make $\mathcal{T}_\theta$ converge. We thus train it to satisfy P1–P3, as detailed in Sec. 3.4.

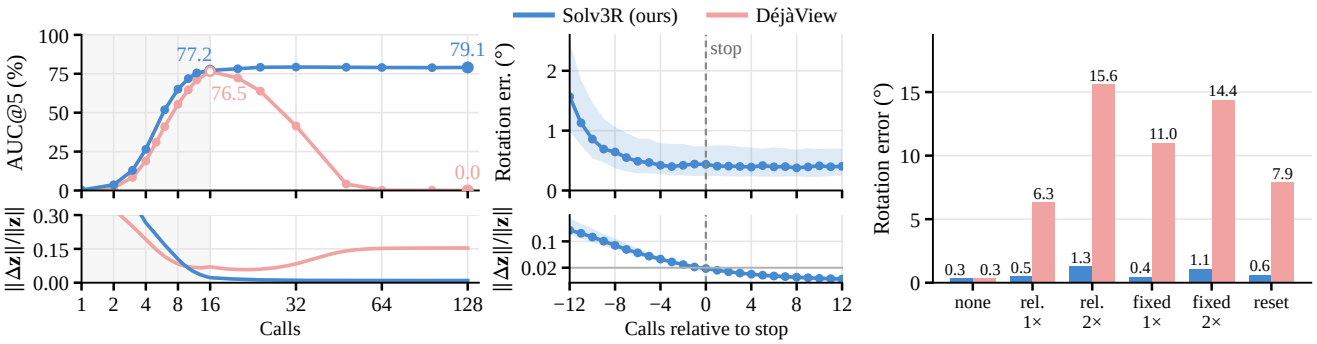


Figure 5. **Stable and corrective solver dynamics.** Left: pose accuracy and relative state update norm over solver calls. DéjàView is trained with up to 16 steps (shaded). Middle: camera rotation error and relative state update aligned to the solver call at which each window meets the stopping criterion of (5), showing median and interquartile range. Right: rotation error at call 16 after perturbations at call 8 using relative (rel.) or fixed-scale (fixed) Gaussian noise, or state resets (reset).

3.4. Learning solver dynamics

We train $\mathcal{T}_\theta$ to satisfy P1–P3. When the input images do not change, a solved state $\mathbf{z}^*$ should change little under further solver calls, $\mathcal{T}_\theta(\mathbf{z}^*;\mathbf{z}^0)\approx\mathbf{z}^*$. From a perturbed state, the same operator should bring the reconstruction back to the solution. We thus train it on states that have received different numbers of solver calls, as well as on perturbed states.

Training states. To keep solved states unchanged under further solver calls (P1), we train the operator on heterogeneous states.

In the main training stage, starting from $\mathbf{z}^0$, we apply a random number of solver calls without gradients, followed by several differentiable calls with supervision at the final call. This exposes the operator to states at different refinement stages, encouraging stable updates under continued computation. To train the operator to correct perturbed states (P2), we occasionally perturb the state before the differentiable calls—we either add Gaussian noise to the whole state or reset a subset of views to $\mathbf{z}^0$. Gaussian noise mimics deviations from a solution, while partial resets simulate newly introduced views alongside refined ones (P3). Similar in spirit to denoising-based learning [72, 83] and Diffusion Forcing [17], these perturbations expose the operator to a broader range of states, encouraging corrective refinement beyond its usual trajectories.

Geometric supervision. We supervise geometric predictions after varying numbers of solver calls, rather than after a fixed number. At the end of each sampled rollout, we decode camera poses, ray maps, and depths from the current state and apply ground-truth geometric supervision where available. We do not supervise the states of earlier calls; instead, varying the supervised call trains the operator to produce accurate geometry at any call between the 24th and the 48th.

To preserve the priors of the pretrained DéjàView, we also distill its predictions after 16 solver calls into the solver.

The overall training objective is

$$\mathcal{L} = \mathcal{L}_{\text{geo}} + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}}, \tag{4}$$

where $\mathcal{L}_{\text{geo}}$ is the loss against the ground truth and $\mathcal{L}_{\text{dist}}$ is the DéjàView distillation loss. We provide further details on the loss terms and training procedure in the supplementary material.

4. Solver analysis

We first evaluate whether Solv3R exhibits the three solver properties defined in Section 3.2, comparing it with DéjàView [12].

Stability (P1) and correction (P2) – Figure 5. We examine whether Solv3R preserves accurate reconstructions under continued solver calls and recovers from perturbed states. To measure solver stabilization, we define the relative state update between consecutive calls as

$$r^k = \frac{\|\mathbf{z}^{k+1} - \mathbf{z}^k\|_2}{\|\mathbf{z}^k\|_2}. \tag{5}$$

As shown in Figure 5 (left), Solv3R maintains reconstruction accuracy as its relative state updates diminish, whereas DéjàView loses accuracy under continued computation. We use this measure for adaptive inference, stopping refinement when $r^k < \gamma$. We empirically select γ=0.02 to balance reconstruction accuracy and computational cost. Figure 5 (middle) shows the relative state updates and camera rotation errors aligned at the stopping call. Camera errors have largely stabilized by this point and remain low under further refinement. In our convergence analysis over 148 multi-view reconstruction windows, 144 meet the stopping criterion within 64 calls, of

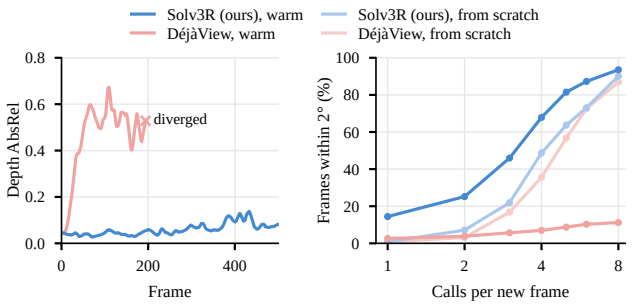


Figure 6. **Warm-start convergence.** Left: depth AbsRel along a KITTI [33] sequence, warm-starting each new frame from the previous solution with four solver calls. Right: percentage of new frames with rotation error below 2° versus solver calls in a 24-view window, comparing warm-starting with restarting from scratch. Solv3R remains stable under continued warm-starting and achieves accurate camera estimates with fewer calls, whereas DéjàView diverges.

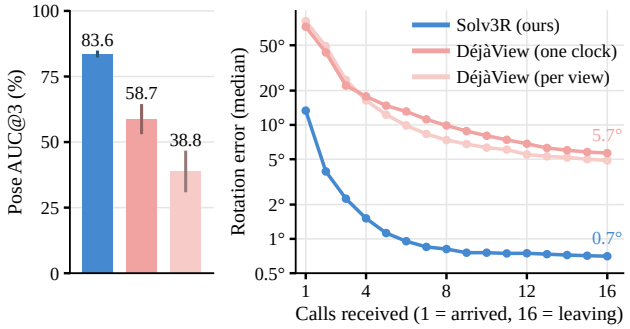


Figure 7. **Mixed-progress solving.** Left: pose accuracy when each view is initialized from a randomly sampled reconstruction stage. Right: rotation error as each view progresses through a pipelined reconstruction window, where views enter and leave after each solver call. Solv3R maintains accurate reconstruction across views with varying reconstruction stages, whereas DéjàView is sensitive to their computational progress.

which 143 (99.3%) have camera rotation errors below 5°. These 144 windows stop after 13 to 32 calls, 17.5 on average, and harder windows take more calls. Further analyses in the supplementary material demonstrate reliable stopping across additional datasets and examine sensitivity to the stopping threshold. To test corrective behavior, we perturb the state after eight solver calls by adding Gaussian noise or resetting views to their initial states, then continue refinement for eight more calls. As shown in Figure 5 (right), Solv3R recovers accurate camera poses from every perturbation, whereas DéjàView does not.

Warm-start convergence (P3) – Figure 6. When a new frame arrives, a reusable solver should retain the states of previously reconstructed views rather than restarting the entire reconstruction. In a sliding-window setting, we preserve the current states of overlapping views while initializing newly introduced views from their image encodings. As shown in Figure 6, Solv3R maintains stable reconstruction accuracy over repeated warm starts, whereas DéjàView diverges. Additionally, Solv3R reaches accurate camera estimates in fewer solver calls than restarting from scratch, demonstrating that previously computed states can be effectively reused.

Mixed-progress solving (P3) – Figure 7. A reusable multi-view solver should jointly update views at different refinement stages without requiring synchronized progress. We evaluate this capability through experiments in Figure 7. In Figure 7 (left), we initialize each view from randomly selected reconstruction stages (0–16 calls). Despite having input in varying stages, Solv3R produces nearly identical camera rotations across independent trials, with a mean difference of only 0.08°. In contrast, DéjàView remains sensitive to initialization as simply using a looped transformer does not result in solver-like properties.

Enhanced throughput via pipelining – Figure 7. This mixed-progress solving capacity allows us to enhance our throughput. In Figure 7 (right) we further test a *pipelined* setup, where the sliding window advances after every solver call rather than waiting for convergence. In other words, each view in the window has an increasing number of calls, with the most recent view having a single call, and the oldest view in the window having the most calls. Solv3R progressively refines camera estimates properly despite this non-conventional setup. It is important to point out that this means that, once the first, *e.g.*, 16 calls are made, the delay between estimates for each frame is now a *single call*—frames are already being processed while previous frames are being estimated. This effectively enhances the throughput of our solver by the window size (note that the latency of estimates remains the same). For example, compared to DéjàView, which must re-solve the window for every new frame, we share the same latency when we use 16 calls, but our solver processes 16× more frames per second (10× end to end).

Reusable geometric context. The properties of Solv3R also lead to efficient *caching*. For efficient inference, key–value (KV) caching allows previously reconstructed views to serve as fixed attention context without repeated computation. In conventional stacked GFMs, these representations are layer-specific and must be cached separately for each transformer layer. Although DéjàView shares transformer weights across iterations, its step-dependent updates *limit* KV cache reuse across refinement stages. By contrast, Solv3R reuses a single KV cache computed from fixed views across all solver calls, regardless of the query's refinement stage. This extends reuse beyond model parameters to

Method	Efficiency		Camera Pose										Geometry									
			DTU		ETH3D		7-Scenes		ScanNet++		nuScenes		DTU		ETH3D		7-Scenes		ScanNet++		nuScenes	
	Params	TFLOPs	@3	@30	@3	@30	@3	@30	@3	@30	@3	@30	L2	IR	L2	IR	L2	IR	L2	IR	L2	IR
MASt3R [47] (ECCV'24)	689M (5.9×)	500.0	21.6	81.7	35.2	57.6	9.3	70.6	15.5	43.9	9.4	62.7	.011	94.9	.340	29.1	.076	41.7	.251	14.5	.360	11.4
MASt3R-SfM [27] (3DV'25)	690M (5.9×)	1150.1	40.2	91.4	52.8	85.0	16.4	80.3	38.7	85.0	9.1	70.9	.009	96.9	.095	54.1	.051	64.7	.042	69.7	.311	18.4
VGGT [88] (CVPR'25)	1.26B (10.8×)	190.0	96.5	99.8	35.4	82.8	11.1	79.1	15.1	74.7	39.2	83.8	.010	95.8	.053	52.6	.042	70.8	.034	68.4	.081	42.3
VGGT-Ω [89] (CVPR'26)	1.14B (9.7×)	99.8	77.4	98.1	64.1	95.4	21.3	87.0	29.9	87.3	42.4	85.5	.009	97.2	.024	78.6	.039	65.3	.032	70.6	.055	62.3
π^3 [95] (ICLR'26)	959M (8.2×)	153.8	70.2	97.3	43.2	84.9	11.3	81.3	76.5	97.3	24.8	82.4	.009	97.3	.034	66.8	.032	77.8	.014	94.3	.078	51.0
MapAnything [42] (3DV'26)	1.23B (10.5×)	148.4	18.1	88.8	37.1	73.8	8.2	74.6	70.6	96.7	37.8	82.8	.014	95.2	.227	40.6	.044	67.9	.019	89.2	.089	51.9
DA3-L [51] (ICLR'26)	411M (3.5×)	71.4	79.4	98.4	42.0	75.6	11.4	79.7	7.0	71.8	10.9	74.6	.010	97.1	.210	49.9	.039	69.7	.051	47.8	.141	27.0
DA3-G [51] (ICLR'26)	1.32B (11.3×)	178.7	86.4	99.1	63.2	83.2	15.4	81.3	24.3	80.1	39.8	82.3	.010	97.1	.137	64.7	.037	71.0	.041	57.7	.080	42.0
DéjàView [12] (NeurIPS'26)	117M	75.9	83.3	98.8	66.0	95.4	13.9	81.7	79.5	98.0	43.4	85.3	.009	97.1	.026	78.3	.035	74.2	.015	93.3	.067	58.5
Solv3R (ours), fixed	117M	76.5	84.5	99.0	65.4	94.8	13.8	81.8	80.7	98.2	41.5	85.3	.010	96.9	.029	74.3	.035	74.4	.016	92.5	.059	59.9
Solv3R (ours), adaptive	117M	91.3	85.5	99.1	70.4	96.3	14.2	81.9	81.2	98.2	41.5	85.3	.009	97.1	.026	77.9	.034	75.8	.015	92.6	.059	60.4

Table 1. **Standard (offline) multi-view reconstruction benchmarks.** Quantitative comparison with state-of-the-art methods [12, 27, 42, 47, 51, 88, 89, 95] across five benchmark datasets [16, 40, 70, 71, 107]. We report camera pose AUC@3/@30 (↑), geometry relative L2 (↓), and inlier ratio (IR, ↑). TFLOPs are measured for 24 views. Solv3R (fixed) uses 16 solver calls, while Solv3R (adaptive) stops based on the relative state update criterion in Eq. (5) with $\gamma = 0.02$.

the results of geometric computation, reducing KV cache storage by 16× compared with stage-specific caching in DéjàView. We demonstrate this capability applied to map-based relocalization in Section 5.3.

5. Experiments

We initialize from the pretrained looped transformer [12] and fine-tune using AdamW [57] with a learning rate of 2×10^{-5} and an exponential moving average (EMA) of the weights. Each training iteration samples 4–64 views and performs 16–40 solver calls without gradients, followed by 8 differentiable calls supervised at the final step. Before the differentiable calls, we apply feature noise or reset selected views with probability 0.15, and set λ_{dist}=0.5. We train for approximately one week on 8×H100 GPUs. Further implementation details, dataset splits, training configurations, and evaluation protocols are provided in the supplementary material.

5.1. Offline reconstruction

We first compare our method with state-of-the-art multi-view reconstruction methods on standard benchmarks. As shown in Table 1, Solv3R achieves competitive camera pose and geometry accuracy with a compact model. Importantly, Solv3R retains accuracy comparable to DéjàView at the same 16-call budget while exhibiting the stable and corrective solver dynamics demonstrated in Section 4. These properties enable continued refinement and reuse of previously computed states beyond a predefined iteration schedule. The stable solver dynamics also enable adaptive inference using (5), with γ=0.02 and a maximum of 64 solver calls. With adaptive stopping, we can improve nearly all reconstruction metrics over our fixed-budget inference with only 3–4 *additional* solver calls on average.

5.2. Incremental reconstruction

We next evaluate whether Solv3R can extend and revise an existing reconstruction as new observations arrive. Using evaluation windows from DL3DV [52] and ScanNet++ [107], we first solve four views for eight calls, introduce twelve additional views, and continue joint refinement from the existing states. As shown in Figure 8, Solv3R reconstructs the new views while improving the camera estimates of the original views, whereas DéjàView fails to maintain a consistent reconstruction. Compared with the no-join control using the same number of solver calls, this improvement shows that new observations contribute to correcting the existing solution rather than merely benefiting from further computation. In Figure 9, we examine revisability across different refinement stages and the efficiency of *reusing* previous computation. Adding new views at different refinement stages, even after the initial reconstruction meets the stopping criterion, improves original-view accuracy in most evaluation windows. When incrementally expanding the reconstruction from 8 to 64 views, Solv3R achieves accuracy comparable to re-solving from scratch with fewer solver calls and lower total runtime. These results show that existing reconstructions can be revised as new observations arrive while reusing previous computation, rather than repeatedly solving from scratch.

5.3. Map-based relocalization

We evaluate map-based relocalization on 7-Scenes [71] and Cambridge [43], using maps reconstructed from the training frames of each scene. For each query, we freeze the solved

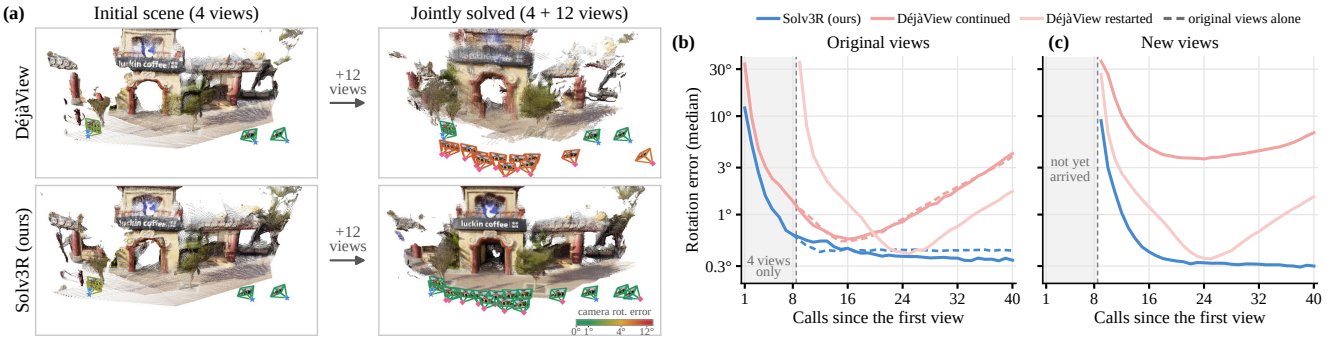


Figure 8. **Revising an existing reconstruction with new views.** (a) Four views are solved for eight calls before twelve new views are introduced; camera colors indicate rotation error. (b,c) Median rotation error of the original and new views during continued refinement over 96 windows. Dashed curves in (b) indicate the no-join controls, where the original four views continue refinement for the same number of solver calls without introducing new views. Solv3R incorporates new views while improving existing camera estimates, whereas DéjàView fails to maintain a consistent reconstruction.

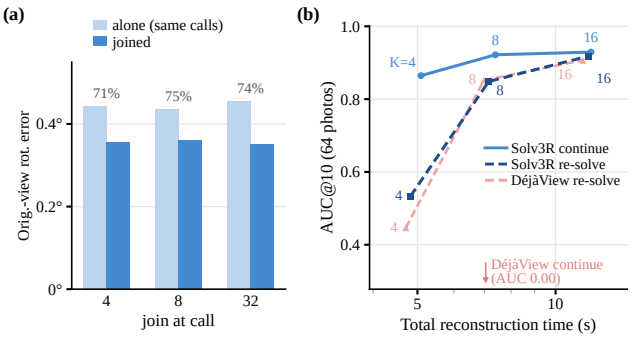


Figure 9. **Revisability and reuse.** (a) Original-view rotation error when new views join at different refinement stages, compared with the no-join control; percentages indicate the fraction of improved windows. (b) Final pose accuracy versus total reconstruction time as the view set expands from 8 to 64. Solv3R improves existing estimates across different join times and achieves comparable accuracy to re-solving with less computation.

Method	Cache (MB/view)↓	Time (ms)↓	7-Scenes Med.↓	7-Scenes 5cm/5°↑	Cambridge Med.↓	Cambridge 25cm/2°↑
VGGT	97.6	186	6.35 / 1.35	45.5	183.2 / 1.77	16.4
VGGT-Ω	78.5	158	8.87 / 1.68	32.8	187.7 / 1.18	14.9
π^3	89.7	161	4.11 / 0.99	66.9	85.9 / 0.95	14.7
DA3-L	30.4	62	6.90 / 1.16	36.7	147.9 / 1.86	12.7
DéjàView (all step)	45.8	88	11.30 / 1.31	28.3	131.0 / 1.07	0.7
DéjàView (single)	2.86	50	123.9 / 29.9	0.0	3615 / 51.5	0.0
Solv3R	2.86	51	3.58 / 0.95	72.8	63.5 / 0.74	24.9

Table 2. **Map-based relocalization.** Evaluation on indoor 7-Scenes [71] and outdoor Cambridge Landmarks [43] . Median translation/rotation errors (cm/°), localization recall (%), cache size per view (MB), and query latency (ms).

map representations and refine only the query. We select the 10 nearest map views of each query with an off-the-shelf image retrieval model [2]. The query attends to the cached states of these views and of map views that cover the scene. All methods attend to the same map views for each query. This setting tests whether previously solved geometry can serve as reusable context for subsequent inference. Conventional GFMs require layer-specific KV caches to preserve geometric context. Although DéjàView shares a single transformer block across iterations, reusing only the KV cache of its solved map fails to relocalize queries, requiring separate caches for each refinement stage. In contrast, Solv3R successfully reuses a single solved-map KV cache across all solver calls. As shown in Table 2, Solv3R achieves the highest localization recall on both benchmarks with lower cache storage than layer-wise or stage-specific caching alternatives. Figure 10 shows the queries of one scene, where DéjàView with a single cache scatters them while Solv3R places them where they belong. These results demonstrate that Solv3R produces geometric representations that remain reusable across solver calls, enabling subsequent inference without stage-specific map computation.

5.4. Streaming reconstruction

We evaluate Solv3R against state-of-the-art streaming reconstruction methods across benchmarks covering camera pose, video depth, and 3D reconstruction. We do not introduce streaming-specific training or a separate optimization backend. Instead, we leverage the solver properties established in Section 4 to maintain and update an ongoing reconstruction as new frames arrive: An initial set of anchor views establishes a shared reference frame, while incoming views are warm-started and jointly refined with recent observations. Previously solved views provide frozen, KV-cached geometric context, but can be reactivated for joint refinement when new observations reveal inconsistencies. This allows the reconstruction to incorporate new information and correct earlier estimates without restarting

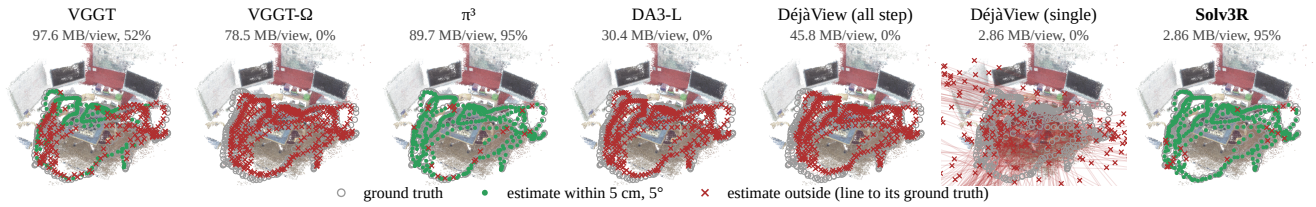


Figure 10. **Relocalization against a frozen map.** Green dots are query estimates within 5 cm and 5° of the ground truth (gray rings), and red crosses are the others, linked to their ground truth. Above each panel are the cache size per map view and the fraction of queries within 5 cm and 5°. With the smallest cache, Solv3R localizes as many queries as the most accurate baseline.

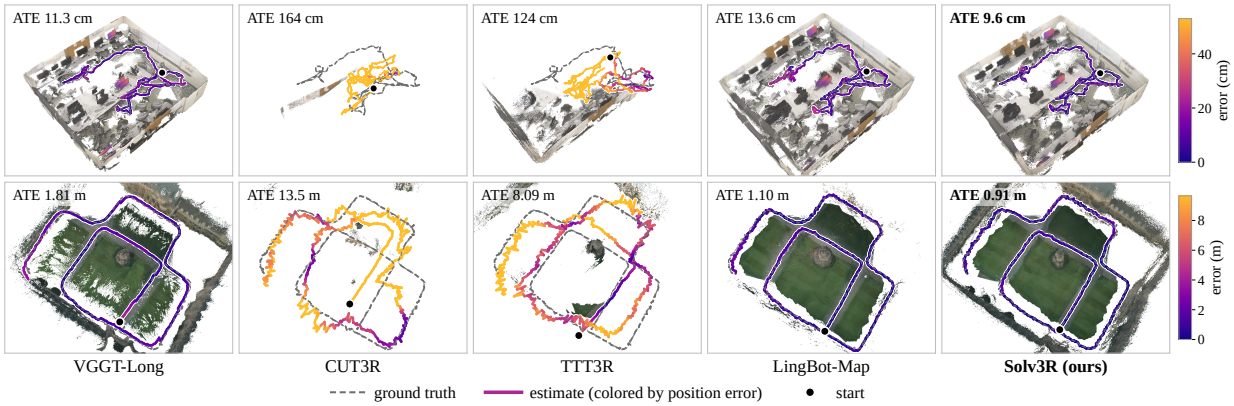


Figure 11. **Streaming reconstruction.** Qualitative comparison with window-based reconstruction [26], recurrent-state methods [21, 90], and geometric-context approaches [18] on long sequences, with camera trajectories colored by position error.

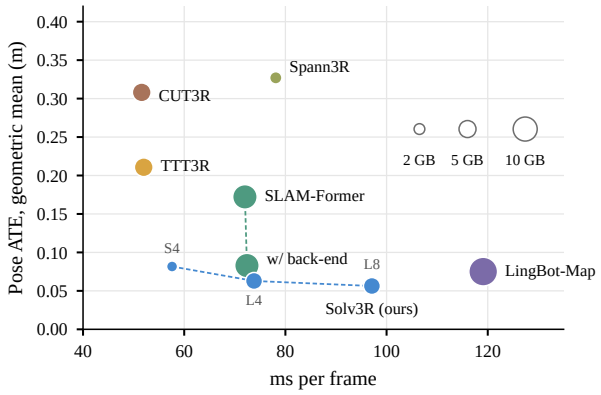


Figure 12. **Streaming accuracy vs. efficiency.** Comparison of online reconstruction methods in terms of geometric-mean pose ATE and per-frame runtime; bubble size indicates peak GPU memory. For Solv3R, S/L denotes small/large memory configurations, and numbers indicate solver calls per frame (L8 is the default).

or chaining separate reconstructions. Details of window management, memory selection, and cache refresh are provided in the supplementary material. Although incoming frames receive further refinement while remaining in the active window, we report predictions at frame arrival for fair comparison with existing streaming methods.

As shown in Table 3, Solv3R achieves strong results across camera pose, video depth, and 3D reconstruction, with competitive runtime and low memory consumption. Figure 11 further illustrates accurate camera trajectories over long sequences, while Figure 12 shows the accuracy–efficiency trade-off. Further ablations in the supplementary material show that retaining previously solved states and periodically reactivating frozen views are important for maintaining reconstruction accuracy over long sequences. These results demonstrate that the same learned solver can achieve competitive streaming reconstruction by continuing and revising existing solutions, without streaming-specific training or a separate optimization backend.

5.5. Ablation studies

We examine the contributions of individual components to Solv3R's solver dynamics in Table 4. For controlled ablations, we fine-tune each configuration for 3k steps from the same checkpoint [12]. While reconstruction accuracy remains comparable at 16 calls, removing individual components reveals distinct failure modes under continued computation and state reuse. Notably, step-conditioned gates are sensitive to the planned refinement horizon P, limiting their flexibility in practical reconstruction settings that require

Method	Efficiency		Camera Pose			Video Depth						3D Reconstruction			
	Time	Mem.	Sintel	TUM-dyn.	ScanNet	Sintel		Bonn		KITTI		7-Scenes		NRGBD	
	(ms)↓	(GB)↓	ATE↓	ATE↓	ATE↓	AbsRel↓	δ ↑	AbsRel↓	δ ↑	AbsRel↓	δ ↑	Cham.↓	NC↑	Cham.↓	NC↑
w/ Separate back-end															
DROID-SLAM† [78] (NeurIPS'21)	36	5.7	.164	.065	.137	–	–	–	–	–	–	–	–	–	–
MASt3R-SLAM [62] (CVPR'25)	73	7.2	.273	.102	.139	–	–	–	–	–	–	–	–	–	–
SLAM-Former [109] (ECCV'26)	72	10.4	.147	.040	.096	–	–	–	–	–	–	–	–	–	–
VGGT-Long‡ [26] (ICRA'26)	584	19.9	.172	.187	.090	.302	.681	.068	.957	.120	.867	.022	.592	.009	.765
Online reconstruction															
CUT3R [90] (CVPR'25)	52	6.1	.204	.170	.843	.431	.471	.084	.939	.130	.866	.069	.566	.085	.616
TTT3R [21] (ICLR'26)	52	6.1	.200	.112	.416	.406	.487	.076	.953	.118	.893	.026	.582	.038	.634
Spann3R [86] (3DV'25)	78	2.6	.311	.147	.767	.649	.325	.091	.930	.343	.442	.038	.566	.046	.615
Point3R [98] (NeurIPS'25)	192	3.5	.388	.171	.694	.478	.452	.082	.948	.104	.915	.034	.578	.054	.621
LingBot-Map [18] (ECCV'26)	119	14.3	.149	.032	.089	.322	.648	.067	.959	.070	.955	.021	.586	.016	.633
Solv3R	95	5.0	.157	.015	.077	.322	.668	.064	.956	.078	.947	.018	.591	.018	.640

† GT camera intrinsics. ‡ Offline loop closure. "–" indicates dense metrics not evaluated for keyframe-based methods.

Table 3. **Streaming reconstruction benchmarks.** Comparison with state-of-the-art streaming reconstruction methods across camera pose, video depth, and 3D reconstruction. We report per-frame runtime (ms) and peak GPU memory (GB), together with camera trajectory ATE (m), depth AbsRel and $\delta_{1.25}$, and reconstruction Chamfer distance (m) and normal consistency (NC).

continued solving without a predefined iteration budget. Input injection, state normalization, and random-depth pre-calls are all critical for long-horizon stability, as removing any one causes accuracy to collapse at 1024 calls. In contrast, removing perturbation training largely preserves long-horizon accuracy but degrades joining and cached-context inference, showing that stability alone does not guarantee corrective refinement. With sufficient training, Solv3R also recovers the initial drop in fixed-budget accuracy while retaining stable and corrective solver dynamics. Together, these results demonstrate that accurate fixed-budget predictions are insufficient for reusable geometric solving, which requires both stable continuation and corrective updates.

6. Conclusion

In this paper, we introduced Solv3R, which reformulates multi-view 3D reconstruction as an ongoing geometric solving process. By combining a stationary update operator with learned stable and corrective dynamics, Solv3R can *continue* refining existing estimates, *integrate* new observations, and *reuse* previously computed geometric representations. Our experiments demonstrate that these properties enable a single learned solver to handle diverse reconstruction settings, including offline and incremental reconstruction, map-based relocalization, and streaming, without task-specific learned modules or a separate optimization backend. These results suggest a path toward more open-ended 3D reconstruction, where geometric estimates can be continuously refined and revised as observations change. More broadly, we believe this perspective could extend to vision-language models, robotics, and world modeling, where maintaining and refining evolving states is a fundamental challenge.

Method	AUC@5↑			Stability↓	Join↓	Cache↓
	16	128	1024	(°)	(°)	(°)
DéjàView (released)	76.5	0.0	0.0	8.37	1.80	55.5
Fine-tuned for 3k steps						
DéjàView w/ our training	73.5	19.3	0.0	3.94	1.31	1.05
Solv3R	72.5	70.8	65.4	1.26	0.62	0.76
w/ step-conditioned (P=16)	71.5	52.1	23.5	1.96	0.81	0.95
w/ step-conditioned, (P=1024)	61.9	43.3	52.2	4.33	2.53	2.58
w/o input injection	71.6	33.1	0.0	5.98	0.64	0.74
w/o state normalization	74.3	70.5	0.0	0.94	0.59	0.72
w/o random-depth pre-calls	73.1	3.4	0.0	7.30	4.37	4.15
w/o perturbation	71.0	69.1	61.7	3.32	4.24	3.51
Solv3R (final)	77.2	79.1	78.8	0.22	0.54	0.72

Table 4. **Ablation study.** All ablated variants are fine-tuned for 3k steps from the same pretrained initialization. The released DéjàView and the final Solv3R models are included as references. We report pose accuracy at different solver calls, along with rotation errors after perturbation (Stability), new-view insertion (Join), and inference with frozen KV context (Cache). For step-conditioned gates, P is the number of calls their schedule is planned for.

Acknowledgments

This work was supported in part by the Sony Faculty Innovation Award; the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant and Alliance Advantage Grant; the Digital Research Alliance of Canada; and Advanced Research Computing at the University of British Columbia.

References

[1] Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J. Zico Kolter, and Roger B. Grosse. Path Independent Equilibrium Models Can Better Exploit Test-Time Computation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 8, 17

[3] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural RGB-D Surface Reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 18

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016. 16

[5] Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed Recursive Transformers: Effective Parameter Sharing with Layer-wise LoRA. In *International Conference on Learning Representations (ICLR)*, 2025. 3

[6] Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron C. Courville, and Se-Young Yun. Mixture-of-Recursions: Learning Dynamic Recursive Depths for Adaptive Token-Level Computation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 3

[7] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep Equilibrium Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 3

[8] Shaojie Bai, Zhengyang Geng, Yash Savani, and J. Zico Kolter. Deep Equilibrium Optical Flow Estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[9] Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end Algorithm Synthesis with Recurrent Networks: Extrapolation without Overthinking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[10] Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 1994. 2, 3

[11] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 16

[12] Alessandro Burzio, Tobias Fischer, Sven Elflein, Qunjie Zhou, Riccardo de Lutio, Jiawei Ren, Jiahui Huang, Shengyu Huang, Marc Pollefeys, Laura Leal-Taixé, Zan Gojcic, and Haithem Turki. Déjà View: Looping Transformers for Multi-View 3D Reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026. 2, 3, 4, 5, 7, 9, 16, 17, 21

[13] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A Naturalistic Open Source Movie for Optical Flow Evaluation. In *European Conference on Computer Vision (ECCV)*, 2012. 18

[14] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. *arXiv preprint arXiv:2001.10773*, 2020. 16

[15] Yohann Cabon, Lucas Stoffl, Leonid Antsfeld, Gabriela Csurka, Boris Chidlovskii, Jerome Revaud, and Vincent Leroy. MUSt3R: Multi-view Network for Stereo 3D Reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[16] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 7, 17, 18

[17] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion Forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 5

[18] Lin-Zhuo Chen, Jian Gao, Shangzhan Zhang, Yihang Chen, Ka Leong Cheng, Yipengjing Sun, Liangxiao Hu, Nan Xue, Xing Zhu, Yujun Shen, Yao Yao, and Yinghao Xu. LingBot-Map: Geometric Context Transformer for Streaming 3D Reconstruction. *arXiv preprint arXiv:2604.14141*, 2026. 3, 9, 10, 20, 21

[19] 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*, 2026. 3

[20] Weirong Chen, Chuanxia Zheng, Ganlin Zhang, Andrea Vedaldi, and Daniel Cremers. NOVA3R: Non-pixel-aligned Visual Transformer for Amodal 3D Reconstruction. In *International Conference on Learning Representations (ICLR)*, 2026. 3

[21] Xingyu Chen, YUE CHEN, Yuliang Xiu, Andreas Geiger, and Anpei Chen. TTT3R: 3D Reconstruction as Test-Time Training. In *International Conference on Learning Representations (ICLR)*, 2026. 2, 3, 9, 10, 19, 20, 21

[22] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3D Gaussian Splatting from Sparse Multi-View Images. In *European Conference on Computer Vision (ECCV)*, 2024. 3

[23] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niessner. ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 16, 18

[24] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal Transformers.

In *International Conference on Learning Representations (ICLR)*, 2019. 3

[25] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 16

[26] 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. In *International Conference on Robotics and Automation (ICRA)*, 2026. 3, 9, 10, 21

[27] Bardienus Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. MASt3R-SfM: a Fully-Integrated Solution for Unconstrained Structure-from-Motion. In *International Conference on 3D Vision (3DV)*, 2025. 7

[28] Sven Elflein, Qunjie Zhou, and Laura Leal-Taixé. Light3R-SfM: Towards Feed-forward Structure-from-Motion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[29] Can Ufuk Ertenli, Emre Akbas, and Ramazan Gokberk Cinbis. Streaming Multiscale Deep Equilibrium Models. In *European Conference on Computer Vision (ECCV)*, 2022. 3

[30] Xianze Fang, Jingnan Gao, Zhe Wang, Zhuo Chen, Xingyu Ren, Jiangjing Lyu, Qiao-Mu Ren, Zhonglei Yang, Xiaokang Yang, Yichao Yan, and Chengfei Lv. Dens3R: A Foundation Model for 3D Geometry Prediction. In *International Conference on Learning Representations (ICLR)*, 2026. 3

[31] Jingnan Gao, Zhe Wang, Xianze Fang, Xingyu Ren, Zhuo Chen, Shengqi Liu, Yuhao Cheng, Jiangjing Lyu, Xiaokang Yang, and Yichao Yan. MoRE: 3D Visual Geometry Reconstruction Meets Mixture-of-Experts. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[32] Zihui Gao, Ke Liu, Jiawang Bian, Donny Y. Chen, Duochao Shi, Guosheng Lin, Hao Chen, and Chunhua Shen. Scalable Adaptation of 3D Geometric Foundation Models via Weak Supervision from Internet Video. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026. 3

[33] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 6, 18

[34] Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 3

[35] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3D v2: A Versatile Monocular Geometric Foundation Model for Zero-shot Metric Depth and Surface Normal Estimation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2024. 3

[36] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning Multi-View Stereopsis. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 16

[37] Varun Jampani, Kevis-Kokitsi Maninis, Andreas Engelhardt, Arjun Karpur, Karen Truong, Kyle Sargent, Stefan Popov, André Araujo, Ricardo Martin-Brualla, Kaushal Patel, et al. NAVI: Category-Agnostic Image Collections with High-Quality 3D Shape and Pose Annotations. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. 16

[38] Wonbong Jang, Philippe Weinzaepfel, Vincent Leroy, Lourdes Agapito, and Jerome Revaud. Pow3R: Empowering Unconstrained 3D Reconstruction with Camera and Scene Priors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[39] Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. LoopFormer: Elastic-Depth Looped Transformers for Latent Reasoning via Shortcut Modulation. In *International Conference on Learning Representations (ICLR)*, 2026. 3

[40] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanaes. Large Scale Multi-view Stereopsis Evaluation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 7, 17, 18

[41] Jianfei Jiang, Liyong Wang, Haochen Yu, Tianyu Hu, Jiansheng Chen, and Huimin Ma. RRT-MVS: Recurrent Regularization Transformer for Multi-View Stereo. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2025. 3

[42] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal Feed-Forward Metric 3D Reconstruction. In *International Conference on 3D Vision (3DV)*, 2026. 3, 7

[43] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *International Conference on Computer Vision (ICCV)*, 2015. 7, 8, 17

[44] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (TOG)*, 36(4), 2017. 18

[45] 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 (ICLR)*, 2026. 2, 3

[46] Justin Lazarow, David Griffiths, Gefen Kohavi, Francisco Crespo, and Afshin Dehghan. Cubify Anything: Scaling Indoor 3D Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 16

[47] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding Image Matching in 3D with MASt3R. In *European Conference on Computer Vision (ECCV)*, 2024. 1, 3, 7

[48] Yixuan Li, Lihan Jiang, Linning Xu, Yuanbo Xiangli, Zhenzhi Wang, Dahua Lin, and Bo Dai. MatrixCity: A Large-scale City Dataset for City-scale Neural Rendering and Beyond. In *International Conference on Computer Vision (ICCV)*, 2023. 16

[49] Zhengqi Li and Noah Snavely. MegaDepth: Learning Single-View Depth Prediction from Internet Photos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 16

[50] 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. In *International Conference on Learning Representations (ICLR)*, 2026. 3

[51] Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the Visual Space from Any Views. In *International Conference on Learning Representations (ICLR)*, 2026. 1, 2, 3, 7, 21

[52] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedrich Benes, and Aniket Bera. DL3DV-10K: A Large-Scale Scene Dataset for Deep Learning-based 3D Vision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 7, 16, 17, 21

[53] Lahav Lipson, Zachary Teed, and Jia Deng. RAFT-Stereo: Multilevel Recurrent Field Transforms for Stereo Matching. In *International Conference on 3D Vision (3DV)*, 2021. 3

[54] Mingkai Liu, Hao Zhao, and Xingxing Zuo. SURE-Map: Self-Correcting Streaming Geometric Foundation Models. *arXiv preprint arXiv:2609.15795*, 2026. 3

[55] Yuzheng Liu, Siyan Dong, Shuzhe Wang, Yingda Yin, Yanchao Yang, Qingnan Fan, and Baoquan Chen. SLAM3R: Real-Time Dense Scene Reconstruction from Monocular RGB Videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[56] Yifan Liu, Zhiyuan Min, Zhenwei Wang, Junta Wu, Tengfei Wang, Yixuan Yuan, Yawei Luo, and Chunchao Guo. WorldMirror: Universal 3D World Reconstruction with Any-Prior Prompting. In *International Conference on Machine Learning (ICML)*, 2026. 3

[57] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*, 2019. 7, 16

[58] Yuanxun Lu, Jingyang Zhang, Tian Fang, Jean-Daniel Nahmias, Yanghai Tsin, Long Quan, Xun Cao, Yao Yao, and Shiwei Li. Matrix3D: Large Photogrammetry Model All-in-One. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[59] 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)*, 2025. 3

[60] Lukas Mehl, Jenny Schmalfuss, Azin Jahedi, Yaroslava Nalivayko, and Andrés Bruhn. Spring: A High-Resolution High-Detail Dataset and Benchmark for Scene Flow, Optical Flow and Stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 16

[61] Amirkeivan Mohtashami, Matteo Pagliardini, and Martin Jaggi. CoTFormer: A Chain of Thought Driven Architecture with Budget-Adaptive Computation Cost at Inference. In *International Conference on Learning Representations (ICLR)*, 2025. 3

[62] Riku Murai, Eric Dexheimer, and Andrew J. Davison. MASt3R-SLAM: Real-Time Dense SLAM with 3D Reconstruction Priors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3, 10

[63] Tuan Duc Ngo, Jiahui Huang, Seoung Wug Oh, Kevin Blackburn-Matzen, Evangelos Kalogerakis, Chuang Gan, and Joon-Young Lee. DAGE: Dual-Stream Architecture for Efficient and Fine-Grained Geometry Estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 3

[64] Emanuele Palazzolo, Jens Behley, Philipp Lottes, Philippe Giguère, and Cyrill Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. In *International Conference on Intelligent Robots and Systems (IROS)*, 2019. 18

[65] Haosong Peng, Hao Li, Yalun Dai, Yushi Lan, Yihang Luo, Tianyu Qi, Zhengshen Zhang, Yufeng Zhan, Junfei Zhang, Wenchao Xu, and Ziwei Liu. OmniVGGT: Omni-Modality Driven Visual Geometry Grounded Transformer. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 3

[66] Alexander Raistrick, Lahav Lipson, Zeyu Ma, Lingjie Mei, Mingzhe Wang, Yiming Zuo, Karhan Kayan, Hongyu Wen, Beining Han, Yihan Wang, Alejandro Newell, Hei Law, Ankit Goyal, Kaiyu Yang, and Jia Deng. Infinite Photorealistic Worlds Using Procedural Generation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 16

[67] 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 *International Conference on Computer Vision (ICCV)*, 2021. 16

[68] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding. In *International Conference on Computer Vision (ICCV)*, 2021. 16

[69] Johannes L. Schonberger and Jan-Michael Frahm. Structure-From-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[70] Thomas Schops, Johannes L. Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A Multi-View Stereo Benchmark With High-Resolution Images and Multi-Camera Videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 7, 17, 18

[71] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 7, 8, 17, 18

[72] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 5

[73] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *International Conference on Intelligent Robots and Systems (IROS)*, 2012. 18

[74] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 16

[75] Zhenggang Tang, Yuchen Fan, Dilin Wang, Hongyu Xu, Rakesh Ranjan, Alexander Schwing, and Zhicheng Yan. MV-DUSt3R+: Single-Stage Scene Reconstruction from Sparse Views In 2 Seconds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[76] Yifu Tao, Miguel Ángel Muñoz-Bañón, Lintong Zhang, Jiahao Wang, Lanke Frank Tarimo Fu, and Maurice Fallon. The Oxford Spires Dataset: Benchmarking Large-Scale LiDAR-Visual Localisation, Reconstruction and Radiance Field Methods. *The International Journal of Robotics Research*, 45(6):839–857, 2026. 16

[77] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *European Conference on Computer Vision (ECCV)*, 2020. 3

[78] 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)*, 2021. 3, 10

[79] Fabio Tosi, Yiyi Liao, Carolin Schmitt, and Andreas Geiger. SMD-Nets: Stereo Mixture Density Networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 16

[80] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly-Throughs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 21

[81] Shinji Umeyama. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1991. 18

[82] Basile Van Hoorick, Rundi Wu, Ege Ozguroglu, Kyle Sargent, Ruoshi Liu, Pavel Tokmakov, Achal Dave, Changxi Zheng, and Carl Vondrick. Generative Camera Dolly: Extreme Monocular Dynamic Novel View Synthesis. In *European Conference on Computer Vision (ECCV)*, 2024. 16

[83] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 2011. 5

[84] Oleg Voynov, Gleb Bobrovskikh, Pavel Karpyshev, Saveliy Galochkin, Andrei-Timotei Ardelean, Arseniy Bozhenko, Ekaterina Karmanova, Pavel Kopanev, Yaroslav Labutin-Rymsho, Ruslan Rakhimov, et al. Multi-Sensor Large-Scale Dataset for Multi-View 3D Reconstruction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 16

[85] Fangjinhua Wang, Silvano Galliani, Christoph Vogel, and Marc Pollefeys. IterMVS: Iterative Probability Estimation for Efficient Multi-View Stereo. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[86] Hengyi Wang and Lourdes Agapito. 3D Reconstruction with Spatial Memory. In *International Conference on 3D Vision (3DV)*, 2025. 2, 3, 10

[87] Hengyi Wang and Lourdes Agapito. AMB3R: Accurate Feed-forward Metric-scale 3D Reconstruction with Backend. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 3

[88] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual Geometry Grounded Transformer. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 2, 3, 7, 20, 21

[89] Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 1, 3, 7, 21

[90] Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A. Efros, and Angjoo Kanazawa. Continuous 3D Perception Model with Persistent State. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 2, 3, 9, 10, 19, 20, 21

[91] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. MoGe: Unlocking Accurate Monocular Geometry Estimation for Open-Domain Images with Optimal Training Supervision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[92] Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. MoGe-2: Accurate Monocular Geometry with Metric Scale and Sharp Details. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 3

[93] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D Vision Made Easy. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 3

[94] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. TartanAir: A Dataset to Push the Limits of Visual SLAM. In *International Conference on Intelligent Robots and Systems (IROS)*, 2020. 16

[95] 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)*, 2026. 1, 2, 3, 7, 21

[96] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen

Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. 16

[97] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, Dahua Lin, and Ziwei Liu. OmniObject3D: Large-Vocabulary 3D Object Dataset for Realistic Perception, Reconstruction and Generation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 16

[98] Yuqi Wu, Wenzhao Zheng, Jie Zhou, and Jiwen Lu. Point3R: Streaming 3D Reconstruction with Explicit Spatial Pointer Memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 3, 10

[99] Guangkai Xu, Hua Geng, Huanyi Zheng, Songyi Yin, Yanlong Sun, Hao Chen, and Chunhua Shen. Unlocking the Power of Critical Factors for 3D Visual Geometry Estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 3

[100] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting Gaussian Splatting and Depth. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[101] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 3

[102] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything V2. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 3

[103] Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped Transformers are Better at Learning Learning Algorithms. In *International Conference on Learning Representations (ICLR)*, 2024. 3

[104] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A Large-Scale Dataset for Generalized Multi-View Stereo Networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 16

[105] Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No Pose, No Problem: Surprisingly Simple 3D Gaussian Splats from Sparse Unposed Images. In *International Conference on Learning Representations (ICLR)*, 2025. 3

[106] Botao Ye, Boqi Chen, Haofei Xu, Daniel Barath, and Marc Pollefeys. YoNoSplat: You Only Need One Model for Feedforward 3D Gaussian Splatting. In *International Conference on Learning Representations (ICLR)*, 2026. 3

[107] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes. In *International Conference on Computer Vision (ICCV)*, 2023. 7, 16, 17, 18, 21

[108] 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. 3

[109] 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. 3, 10, 20

[110] Amir R. Zamir, Alexander Sax, William Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling Task Transfer Learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 16

[111] 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 *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 3

[112] Ganlin Zhang, Weirong Chen, Daniel Cremers, and Xi Wang. BA-T: An Iterative Transformer for Two-View Bundle Adjustment. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026. 3

[113] Junyi Zhang, Charles Herrmann, Junhwa Hur, Chen Sun, Ming-Hsuan Yang, Forrester Cole, Trevor Darrell, and Deqing Sun. LoGeR: Long-Context Geometric Reconstruction with Hybrid Memory. In *European Conference on Computer Vision (ECCV)*, 2026. 3

[114] Shangzhan Zhang, Jianyuan Wang, Yinghao Xu, Nan Xue, Christian Rupprecht, Xiaowei Zhou, Yujun Shen, and Gordon Wetzstein. FLARE: Feed-forward Geometry, Appearance and Camera Estimation from Uncalibrated Sparse Views. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3

[115] Xiaojing Zhang, Haifeng Wu, Gang He, Jiyang Shen, Bochen Lyu, and Zhanxing Zhu. MoDr: Mixture-of-Depth-Recurrent Transformers for Test-Time Reasoning. In *International Conference on Learning Representations (ICLR)*, 2026. 3

[116] Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling Latent Reasoning via Looped Language Models. *arXiv preprint arXiv:2510.25741*, 2025. 3

[117] 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. 2, 3, 20

A. Implementation details

We provide details of the architecture, training, and streaming configuration of Solv3R.

Architecture. We build on the pretrained DéjàView [12], retaining its shared transformer block with frame-wise and global attention. We replace the step-conditioned gates with learnable constant gates, which provide the same channel-wise scaling in each sub-block across all solver calls. We initialize these gates by averaging the corresponding gate values over the 16 refinement steps of DéjàView.

To keep the state grounded in the original observations and stable under continued computation, we inject the initial state at every solver call and normalize the updated state:

$$\mathbf{u} = \mathcal{U}_\theta(\mathbf{z}^k + W\bar{\mathbf{z}}^0), \qquad \mathbf{z}^{k+1} = \frac{\rho\,\mathbf{u}}{\|\mathbf{u}\|_{\mathrm{RMS}}}. \tag{6}$$

Here, $\mathbf{z}^k$ is the state after k solver calls, $\mathcal{U}_\theta$ denotes the shared block, $\bar{\mathbf{z}}^0$ is the layer-normalized [4] initial state without affine parameters, and W is a zero-initialized linear projection. The target RMS ρ is the RMS of the final DéjàView state, measured once before training ($\rho \approx 0.85$), and stays fixed throughout training and inference. We scale the entire state by a single factor, which preserves relative feature magnitudes across views and tokens. The resulting model has 117M parameters. It takes images whose longer side is 504 pixels.

Training data. To cover diverse scenes, we train on a broad mixture of indoor, object-centric, outdoor, driving, and synthetic datasets, summarized in Table 5.

Type	Dataset	Weight (%)
Indoor	ScanNet++ [107]	10.8
	ScanNet [23]	7.4
	CA-1M [46]	3.3
	Taskonomy [110]	3.3
Object	CO3D [67]	7.4
	NAVI [37]	3.3
	Skoltech3D [84]	2.5
	OmniObject3D [97]	2.5
	Objaverse renders [25]	2.5
Outdoor	DL3DV [52]	8.2
	BlendedMVS [104]	4.1
	MegaDepth [49]	3.3
	Oxford Spires [76]	1.6
	Mip-NeRF 360* [11]	0.8
Driving	Waymo* [74]	4.1
	Argoverse 2 [96]	3.3
	ParallelDomain-4D [82]	3.3
	Virtual KITTI 2 [14]	1.6
Synthetic	Infinigen [66]	9.1
	TartanAir [94]	5.8
	Hypersim [68]	5.8
	MVS-Synth [36]	1.6
	Spring [60]	1.6
	MatrixCity [48]	1.6
	UnrealStereo4K [79]	1.2

Table 5. **Training data.** Sampling weights (%) in the second training stage. *Supervised only by the distillation loss.

Training objective. The geometric loss $\mathcal{L}_{\mathrm{geo}}$ supervises the ray maps, cameras, and depths decoded at the final call. For ray maps, we apply an L1 loss to ray directions and scale-aligned ray origins. For cameras, we use geodesic losses on the relative rotations and translation directions. For depth, we use a scale-aligned log-L1 loss with weight 0.5. On datasets with dense and accurate depth, we add a multi-scale depth-gradient loss [49] with weight 0.5. These terms align each prediction to the ground truth with its own scale. They do not enforce a common scale across views or between depths and camera translations, so we add two consistency terms with weight 1.5 each. The first penalizes the deviation of each view's depth scale from the window mean. The second supervises the camera centers and ray origins after rescaling them by the window's depth scale. The distillation loss $\mathcal{L}_{\mathrm{dist}}$ applies the ray, log-depth, and camera losses against the 16-step predictions of DéjàView, with $\lambda_{\mathrm{dist}} = 0.5$. For datasets whose ground truth is sparse or noisy, marked with * in Table 5, we use $\mathcal{L}_{\mathrm{dist}}$ only.

Training procedure. To cover different window sizes, we sample each window from one of three size classes: 4–16, 24–32, or 48–64 views, with probabilities 0.4, 0.3, and 0.3. Starting from $\mathbf{z}^0$, we apply a number of solver calls without gradients, drawn uniformly from 16 to 40. We then apply eight differentiable calls and supervise the final call. With probability 0.15, we perturb the state before the differentiable calls. We either add Gaussian noise with a standard deviation of $\sigma\,\|\mathbf{z}\|_{\mathrm{RMS}}$, where σ is drawn uniformly from $[0, 0.5]$, or reset a random subset of up to half of the views to $\mathbf{z}^0$. The noise exposes the solver to globally inaccurate states. The reset mixes refined and unrefined views within one window, as occurs when views are inserted into an existing reconstruction.

Optimization. We train in two stages. The first stage gradually increases the window size and the number of solver calls. It runs for about 70k iterations with a batch of 8 windows. The second stage trains with the full procedure above for about 20k iterations with a batch of 32 windows. To cover rolled cameras, we rotate each view by 90° in either direction with probability 0.15 in its last 2k iterations. We fine-tune only the shared block, the gates, and the projection W, and freeze the image encoder and the prediction heads of DéjàView. We use AdamW [57] with a weight decay of 0.01 and learning rates of 2×10^{-5} for the shared block and the projection, and 2×10^{-4} for the gates. We evaluate an exponential moving average of the weights with a decay of 0.999.

Streaming reconstruction. Streaming uses the final model without any streaming-specific training. For each incoming frame, the solver operates on a window of anchors, memory frames, recent frames, and the new frame. The first a frames serve as anchors that fix the coordinate frame and are initially solved jointly for 16 calls. The r most recent frames form the active window, where each new frame is initialized from its image encoding plus the solved offset of the previous frame. The solver then jointly updates the new and recent frames for m calls, retaining their previously computed states (**continuation**). Anchors and memory frames remain frozen during normal solver calls, providing fixed attention context. Since their representations remain unchanged, their key–value (KV) pairs are computed once and reused across solver calls (**stability**). When a frame leaves the recent window, it is retained in memory if its index is a multiple of s, and discarded otherwise. The memory holds the latest M such frames. To detect inconsistencies with newly observed frames, we periodically check whether the frozen states would change under further refinement. Every four incoming frames, one additional solver call evaluates the relative state update $\|\mathbf{z}^{k+1} - \mathbf{z}^k\|_2 / \|\mathbf{z}^k\|_2$ on the frozen views. If this exceeds $\gamma = 0.02$, the frozen views rejoin joint refinement with the incoming frame, continuing from their existing states rather than restarting from scratch (**correction**). Their KV cache is then rebuilt from the updated states; otherwise, the cached representations are retained. The default configuration (L8) uses $a = 8$, $r = 6$, $s = 16$, $M = 64$, and $m = 8$.

B. Evaluation protocols

We detail the evaluation protocols for the solver-property analyses and the offline, incremental, map-based relocalization, and streaming experiments.

Solver properties. The stability analysis of the main paper uses 148 eight-view windows, 66 from DTU [40], 37 from ETH3D [70], and 45 from ScanNet++ [107] scenes held out from our fine-tuning. We measure rotation errors relative to the first view of each window. To test recovery, we perturb the state of 36 of these windows, 12 from each dataset, at call 8 and report the rotation error at call 16. Relative noise (rel.) is Gaussian noise with a standard deviation of one or two times the RMS of the state of each model. Fixed-scale noise (fixed) uses the mean state RMS of Solv3R and DéjàView instead, so that both models receive the same absolute noise. A reset returns every view except the first to its initial state. The other solver-property experiments use 64-frame sequences from the DL3DV [52] validation split and the held-out ScanNet++ scenes.

Warm start simulates a video that arrives one frame at a time. The window holds the first frame, which fixes the coordinate frame, and the 23 most recent frames. When a new frame arrives, the frames that stay in the window keep their solved states. The new frame starts from its initial state plus the solved offset of the previous frame, as in streaming (Sec. A). Over 16 sequences from each dataset, we count the solver calls until the rotation error of the new frame falls below 2°, and compare them with solving the window from scratch.

Pipelined reconstruction slides a window of the first frame and the 16 most recent frames over a sequence. It advances the window after every solver call rather than after convergence. After each call, the oldest frame other than the first leaves the window, and the next frame enters from its initial state. Apart from the first frame, each call thus updates frames that have received between 0 and 15 previous calls. We evaluate the pose of each frame when it leaves the window, after its 16th call.

Random computational histories test whether the joint update depends on how each view reached its current state. We record the states of a window over its first 16 calls from scratch, and build a new state by taking each view from a randomly chosen call of this trajectory. We then run 16 joint calls from this state, and report the mean difference of the camera rotations across 20 such draws for each of 96 windows of 16 views.

Offline reconstruction. We follow the evaluation protocol and splits of DéjàView [12] on DTU [40], ETH3D [70], 7-Scenes [71], ScanNet++ [107], and nuScenes [16]. We report the area under the curve (AUC) of relative pose errors at 3° and 30°, and the relative L2 error and inlier ratio of the world-frame point maps. For adaptive inference, we stop when the relative update falls below $\gamma = 0.02$, with at least 4 and at most 64 solver calls. Without adaptive stopping, Solv3R runs 16 solver calls, the budget of DéjàView.

Incremental reconstruction. We use 20 scenes, 10 from the DL3DV validation split and 10 from the held-out ScanNet++ scenes. Starting from 8 views in capture order, we add 8 views at a time up to 64 views, and report the pose AUC@10 over all view pairs after each insertion.

Map-based relocalization. For each scene, we solve the training frames once as a map and cache the key-value (KV) pairs of every map frame. Each test frame is a query that we solve for 16 calls against cached map frames only. The attended frames are its 10 nearest map frames retrieved by NetVLAD [2], the reference view of the map, and 64 evenly spaced map frames that cover the scene. We recover the absolute query pose from the ground-truth poses of the retrieved frames and their predicted poses in the map. We report the median translation and rotation errors and the recall within 5 cm and 5° for 7-Scenes [71] and within 25 cm and 2° for Cambridge [43]. All methods attend to the same map frames. We time each query on one H100 GPU, from encoding the query to reading out its pose, including copy-

ing the cached key-value pairs of its retrieved frames from host memory.

Streaming reconstruction. We evaluate on Sintel [13], TUM dynamics [73], ScanNet [23], Bonn [64], KITTI [33], 7-Scenes [71], and NRGBD [3]. We use 1000 frames with a stride of 3 for ScanNet, 500 frames for Bonn and KITTI, and 200 frames with a stride of 2 for 7-Scenes and NRGBD. We evaluate each frame with its prediction at arrival, before any later refinement. The first eight frames are an exception, since Solv3R solves them jointly as anchors. We report the absolute trajectory error (ATE) after Sim(3) alignment [81], the depth AbsRel and $\delta_{1.25}$ after per-sequence scale alignment, and the Chamfer distance and normal consistency of the reconstructed point clouds.

C. Additional analyses

We further examine the stopping behavior of the solver, its sensitivity to initialization, and the reuse of solved states in streaming and incremental reconstruction. We also inspect attention across solver calls and discuss a failure case.

C.1. Stopping and initialization

Stopping behavior. As state updates diminish, geometric error decreases and levels off by the stopping point, remaining low under further refinement (Fig. 5). Adaptive stopping matches the median accuracy of continuing to 64 calls. Of the 148 windows of the stability analysis, 144 reach $\gamma = 0.02$ within 64 calls, after 17.5 calls on average (median 17, interquartile range 15–19). Over these windows, the median rotation error is $0.44°$ at the stopping call and $0.43°$ at call 64. These results indicate that our state-based stopping criterion identifies when further refinement yields little improvement in geometric accuracy. The other four windows are also accurate at call 64, with rotation errors below $3°$. DéjàView, which repeats its final step after 16 calls, does not reach the threshold in any window.

Distribution of stopping calls. The number of calls adapts to each window (Fig. 13). All 144 windows stop between 13 and 32 calls. Solv3R runs more calls on harder windows. The stopping call correlates with the rotation error at that call, with a Spearman correlation of 0.77. The median stopping call also differs across datasets, from 14 calls on ScanNet++ to 18.5 calls on DTU.

Sensitivity to the threshold. The threshold $\gamma = 0.02$ balances accuracy and cost (Tab. 6). A tighter threshold needs 1.6 to 2.7 times as many calls without consistent gains. A looser threshold saves about a third of the calls but loses 4 to 8 AUC points on DTU, ETH3D, and ScanNet++.

Other datasets. Stopping remains nearly as accurate as continuing to 128 calls on datasets outside the stability analysis (Tab. 7). We solve 107 more eight-view windows from

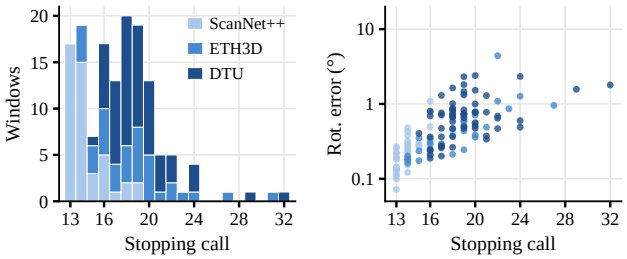


Figure 13. **Stopping calls.** Left: number of windows per stopping call, for the 144 of the 148 windows of the stability analysis that reach $\gamma = 0.02$, stacked by dataset. Right: rotation error of each window at its stopping call.

	DTU [40]		ETH3D [70]		7-Scenes [71]		ScanNet++ [107]		nuScenes [16]	
γ	AUC	Calls	AUC	Calls	AUC	Calls	AUC	Calls	AUC	Calls
0.01	85.9	36.1	69.8	54.9	14.2	35.8	81.5	39.8	41.8	31.1
0.02	85.5	19.7	70.4	20.5	14.2	22.6	81.2	17.3	41.5	18.2
0.05	81.5	13.0	62.0	13.5	13.9	12.9	77.0	11.8	40.9	11.9

Table 6. **Sensitivity to the stopping threshold γ.** Pose AUC@3 and the mean number of solver calls per scene.

Tanks and Temples [44], 7-Scenes [71], and TUM dynamics [73] with moving people. Solv3R reaches $\gamma = 0.02$ in 105 of these windows. On each dataset, its median rotation error at the stopping call is within $0.04°$ of that at call 128. No window stops with an error above $5°$. DéjàView reaches the threshold in none of the windows.

Different initial states. The solver reaches nearly the same camera rotations from different initial states. We solve each window again from two other initial states. The noisy start adds Gaussian noise to $\mathbf{z}^0$ with a standard deviation of half the RMS of each token. In the swapped start, each view except the first starts from the initial state of another view, which is a valid state of a different camera. After 128 calls, the median rotation difference between the start from $\mathbf{z}^0$ and either other start is at most $0.11°$ in every row of Tab. 7.

Initialization of new views. The solver converges from different initial states of a new view. We repeat the warm-start experiment of the main paper with three initializations of the new frame (Fig. 14). The new frame starts from its own initial state $\mathbf{z}^0$, from $\mathbf{z}^0$ plus the solved offset of the previous frame, or from the solved state of the previous frame. The offset gives the most accurate first call, but $\mathbf{z}^0$ reaches the same error within two to four calls. Even when the new frame starts from the solved state of the previous frame, which describes a different camera, the solver corrects it within about 10 calls. All three initializations converge to the error of solving the whole window from scratch.

The initialization matters more when each frame receives only a few calls, as in streaming. We run the default streaming window with 1 to 16 calls per frame (Fig. 15). We

				Error (°)		Stops	Start diff. (°)		DéjàView
Windows	#	Stopped	Calls	Stop	128	$> 5°$	Noise	Swap	at 128 (°)
Stability analysis	148	97%	17.5	0.44	0.42	1	0.09	0.11	96.0
Tanks and Temples	39	100%	17.7	0.29	0.27	0	0.07	0.09	86.6
7-Scenes	36	94%	16.2	1.56	1.57	0	0.06	0.06	82.9
TUM dynamics	32	100%	15.7	0.42	0.39	0	0.04	0.05	75.1

Table 7. **Stopping on other datasets.** Eight-view windows solved for 128 calls. Stopped is the share of windows that reach $\gamma = 0.02$, and Calls is their mean stopping call. Error is the median rotation error of the stopped windows at the stopping call and at call 128. Stops $> 5°$ counts the stopped windows with an error above $5°$ at the stopping call. Start diff. is the median rotation difference after 128 calls between the start from $\mathbf{z}^0$ and the noisy or swapped start. The last column is the median rotation error of DéjàView at call 128.

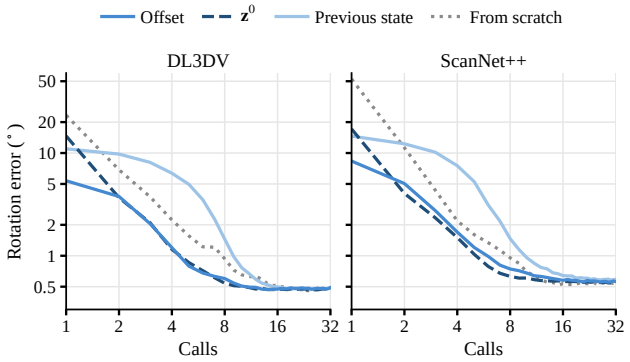


Figure 14. **Initialization of new views.** Median rotation error of a newly arrived frame over solver calls, for 656 frames per dataset, 41 from each of 16 sequences. The new frame starts from its initial state $\mathbf{z}^0$, from $\mathbf{z}^0$ plus the solved offset of the previous frame (Offset), or from the solved state of the previous frame (Previous state). The frames that remain in the window keep their solved states. "From scratch" solves the whole window from the initial states.

compare a cold start from $\mathbf{z}^0$ with a warm start from $\mathbf{z}^0$ plus the solved offset of the previous frame. With one call per frame, the warm start reduces the ATE about sixfold on TUM dynamics and 3.5-fold on 7-Scenes. From eight calls per frame, both initializations give nearly the same accuracy. We thus warm-start new frames in all streaming configurations.

C.2. Streaming memory and long sequences

Configurations. The number of calls per frame and the window size set the speed of streaming, and the window size also sets its GPU memory. We compare the default configuration (L8) with L4, which halves the calls per frame, and with S4, which also uses fewer anchors, recent frames, and memory frames (Tab. 8). L4 shortens the runtime

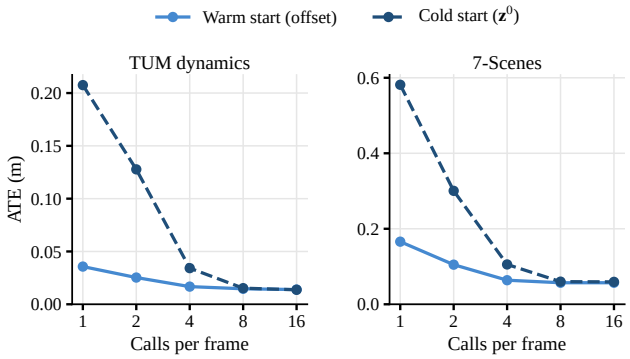


Figure 15. **Initialization in streaming.** ATE of on-arrival predictions with the L8 window for different numbers of calls per frame. Each new frame starts from its initial state $\mathbf{z}^0$ (cold start) or from $\mathbf{z}^0$ plus the solved offset of the previous frame (warm start).

by about a third while keeping its video depth nearly unchanged. S4 needs 40% less runtime and less than half the GPU memory of L8. Compared with CUT3R [90] and TTT3R [21] in the streaming benchmark of the main paper, S4 remains more accurate in camera pose and video depth, at a similar runtime and a third of their GPU memory.

On-arrival and refined predictions. The solver refines each frame until it leaves the recent window, six frames after its arrival. The main paper reports every frame at its arrival, for comparison with existing online methods. The last row of Table 8 reports the same run with each frame read when it leaves the recent window. These refined predictions are more accurate on most metrics, most clearly for video depth on Sintel and KITTI.

Reuse in streaming. On long sequences, keeping solved frames in the window makes Solv3R far more accurate than re-solving a short window (Tab. 9). The gain grows with the sequence length, to 7-fold on ScanNet and more than 10-fold on TUM dynamics. Keeping solved frames prevents the drift that re-solving accumulates over long sequences, which the 46 frames of a Sintel sequence leave little room for. Without reuse, the solver solves each new frame together with the seven previous frames from their initial states. Solv3R and DéjàView then reach nearly the same accuracy. The L8 window keeps the solved states of its anchors, memory frames, and recent frames. DéjàView cannot refine frames past its schedule in the same window. Its poses diverge after about 50 frames on TUM dynamics and ScanNet. Within the window, refreshing the frozen frames when new observations disagree with them matters most. Without the refresh, the ATE of Solv3R grows more than 10-fold on TUM dynamics and ScanNet.

Long sequences. Solv3R streams with bounded GPU memory, since its window holds a fixed number of frames.

					ATE (m)↓			Video depth AbsRel↓			Chamfer (m)↓		Time↓	GPU mem.↓
Config	a	r	M	m	Sintel	TUM	ScanNet	Sintel	Bonn	KITTI	7-Scenes	NRGBD	(ms)	(GB)
L8	8	6	64	8	.157	.015	.077	.322	.064	.078	.018	.018	95	5.0
L4	8	6	64	4	.163	.017	.091	.321	.064	.083	.037	.024	65	5.0
S4	4	4	16	4	.179	.017	.180	.328	.065	.088	.072	.045	51	2.1
L8, refined	8	6	64	8	.162	.014	.076	.302	.065	.072	.017	.017	95	5.0

Table 8. **Streaming configurations.** a, r, M, and m are the numbers of anchors, recent frames, memory frames, and calls per frame. Every configuration takes a memory frame every 16 frames. Runtime is the mean time per frame on an H100 GPU, including the image encoder and the prediction heads. As in the main paper, we measure it on the TUM dynamics sequences sitting_halfsphere, sitting_static, and walking_rpy. The last row reads each frame of the L8 run when it leaves the recent window instead of at its arrival.

Window	Method	Sintel (46)	TUM (878)	ScanNet (602)
No reuse	DéjàView	.124	.218	.560
	Solv3R	.118	.214	.574
L8	DéjàView	.591	div.	div.
	Solv3R	.157	.015	.077
	no refresh	.444	.160	1.05
	no recent window	.265	.014	.324

Table 9. **Reuse in streaming.** ATE (m) of on-arrival predictions. Without reuse, the solver solves each new frame with the seven previous frames from their initial states for 16 calls. We chain its pose through the first frame of the window. In the L8 window, DéjàView runs its step schedule for each frame and then repeats its final step. Its poses diverge (div.). Without the refresh, the frozen frames never rejoin the update. No recent window drops the six recent frames, so that each call updates only the new frame.

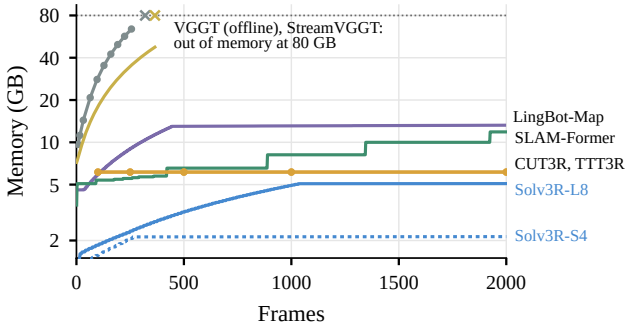


Figure 16. **Memory on a long sequence.** GPU memory over the 2000 frames of the long office sequence of TUM RGB-D, on a logarithmic axis. All methods are evaluated on a single H100 GPU. Crosses mark where a method runs out of the 80 GB of GPU memory (gray dotted line).

We track the peak GPU memory over a sequence of 2000 frames (Fig. 16). The GPU memory of Solv3R stops growing once it holds M memory frames, after about 1030 frames for L8 and 260 frames for S4. CUT3R [90], TTT3R [21], and LingBot-Map [18] also stay bounded. LingBot-Map needs more than twice the GPU memory of L8. The GPU memory of SLAM-Former [109] keeps growing with the sequence. Even on an H100 GPU with 80 GB, offline VGGT [88] runs out of memory at 320 frames and StreamVGGT [117] after 365 frames.

C.3. Attention across solver calls

Stable predictions under continued computation do not necessarily imply that the model preserves meaningful geometric correspondences internally. We therefore examine the global attention of the shared block across solver calls. A head finds the correspondence of a query patch when its strongest attention in an adjacent view falls within one patch of the ground-truth correspondence. At call 16, the same three heads are the most accurate in both models. They find the correspondence for 51 to 74% of the queries (Fig. 17). Solv3R keeps the accuracy of these heads after call 16, and their attention maps at call 64 remain close to those at call 16, with a similarity above 0.96. The heads of DéjàView instead lose their correspondences once its schedule ends, and their accuracy drops below 3% by call 64.

C.4. Incremental reuse and insertion order

Calls per insertion. Continuing from the current state reaches higher accuracy than re-solving from scratch at the same cost. With 8 calls per insertion, continuing reaches an AUC@10 of 92.2 after 64 views, while re-solving with 8 calls reaches 84.8. Re-solving needs 16 calls per insertion to reach 91.8. Beyond 8 calls, continuing improves only slightly, to 92.9 with 16 calls and 93.1 with 32 calls.

Insertion order. We feed the same 64 views in capture order, in reverse order, and in two random orders (Tab. 10). Continuing with half the calls stays within two points of re-solving in every order.

C.5. Failure case

The solver may occasionally stop at a suboptimal solution. In the stability analysis of the main paper, 143 of the 144 windows that meet the stopping criterion stop below 5° of rotation error. The remaining window is shown in Fig. 18. Its views form two groups with little overlap, and the solver places the second group at a wrong position and orientation.

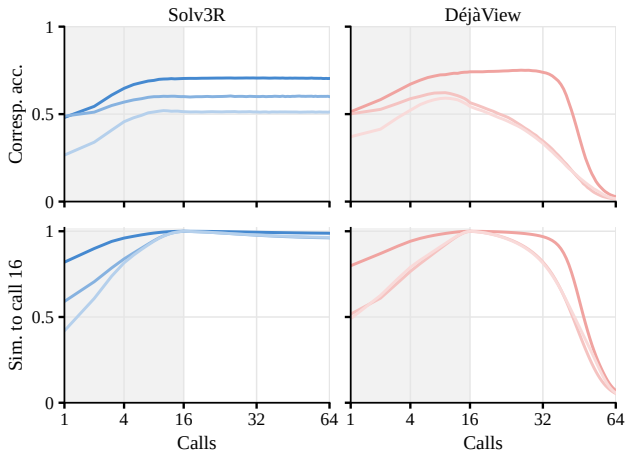


Figure 17. **Attention across solver calls.** We show the three global-attention heads, out of 12, that best find ground-truth correspondences at call 16. These are the same heads in Solv3R (left) and DéjàView (right). Top: fraction of query patches whose strongest attention in an adjacent view falls within one patch of the ground-truth correspondence. Bottom: similarity to the call-16 attention map, measured as the share of the call-16 attention mass on the 64 most attended keys of each call and normalized to 1 at call 16. The shaded range is the 16-step schedule of DéjàView. Averaged over 82 windows of ETH3D and ScanNet++.

	DL3DV [52]		ScanNet++ [107]	
Order	Continue	Re-solve	Continue	Re-solve
Capture	92.0	91.2	92.4	92.5
Reverse	89.9	91.6	92.4	92.4
Random 1	91.7	91.7	92.1	91.8
Random 2	92.0	92.1	92.3	92.3

Table 10. **Sensitivity to the insertion order.** AUC@10 after inserting 64 views, 8 at a time. Continue runs 8 calls per insertion from the current state, and re-solve runs 16 calls from scratch.

D. Additional results

We complement the main benchmarks with an evaluation on large aerial scenes and qualitative results under different view-arrival patterns, including offline and streaming reconstruction.

D.1. Large aerial scenes

Solv3R estimates the most accurate cameras on both large aerial scenes of Mill19 [80], Rubble and Building (Tab. 11). Every method reconstructs each scene offline from the same 96 evenly spaced images in a single window. We compare the predicted cameras with the reference poses of the dataset.

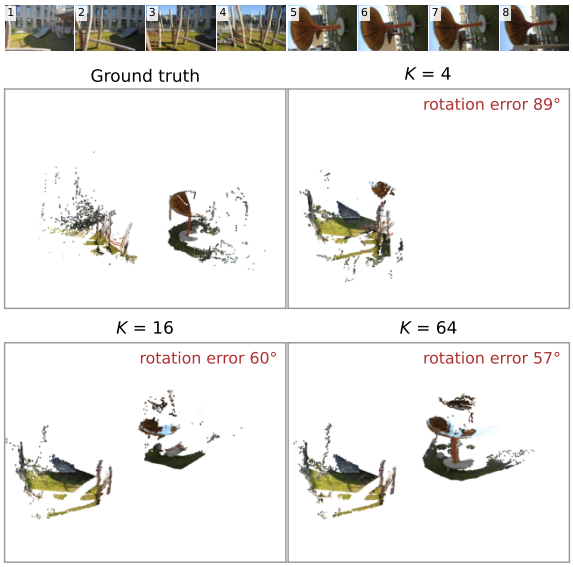


Figure 18. **Failure case.** In this window, views 5–8 share little content with views 1–4. Solv3R places them at a wrong position and orientation within 16 calls and keeps them there up to call 64.

	Rubble		Building	
Method	@3	@30	@3	@30
VGGT [88]	5.3	82.4	2.5	70.6
VGGT-Ω [89]	34.6	91.8	14.6	86.4
π^3 [95]	14.9	88.3	1.8	75.9
DA3-L [51]	3.8	74.0	1.1	55.8
DéjàView [12]	39.4	93.0	11.0	87.0
Solv3R	**48.9**	**94.1**	**22.5**	**89.6**

Table 11. **Large aerial scenes (Mill19).** Camera pose AUC@3 and AUC@30 from 96 images per scene.

D.2. Qualitative results

Arrival patterns. The same solver reconstructs a scene whether its views arrive all at once, in batches, one at a time, or in arbitrary order (Fig. 19). Views that join continue the running solve instead of restarting it. All four patterns end in nearly the same accurate reconstruction of a scene (Fig. 20).

Offline reconstruction. Figures 21 to 23 present three large indoor scenes, where every method sees all images at once. Solv3R reconstructs them accurately, while some existing methods [12, 51, 88, 89, 95] misplace or disconnect parts of them.

Streaming reconstruction. Figure 24 presents four more sequences streamed with the default configuration (L8), each frame shown with its prediction at arrival. Solv3R gives accurate trajectories, competitive with existing streaming methods [18, 21, 26, 90].

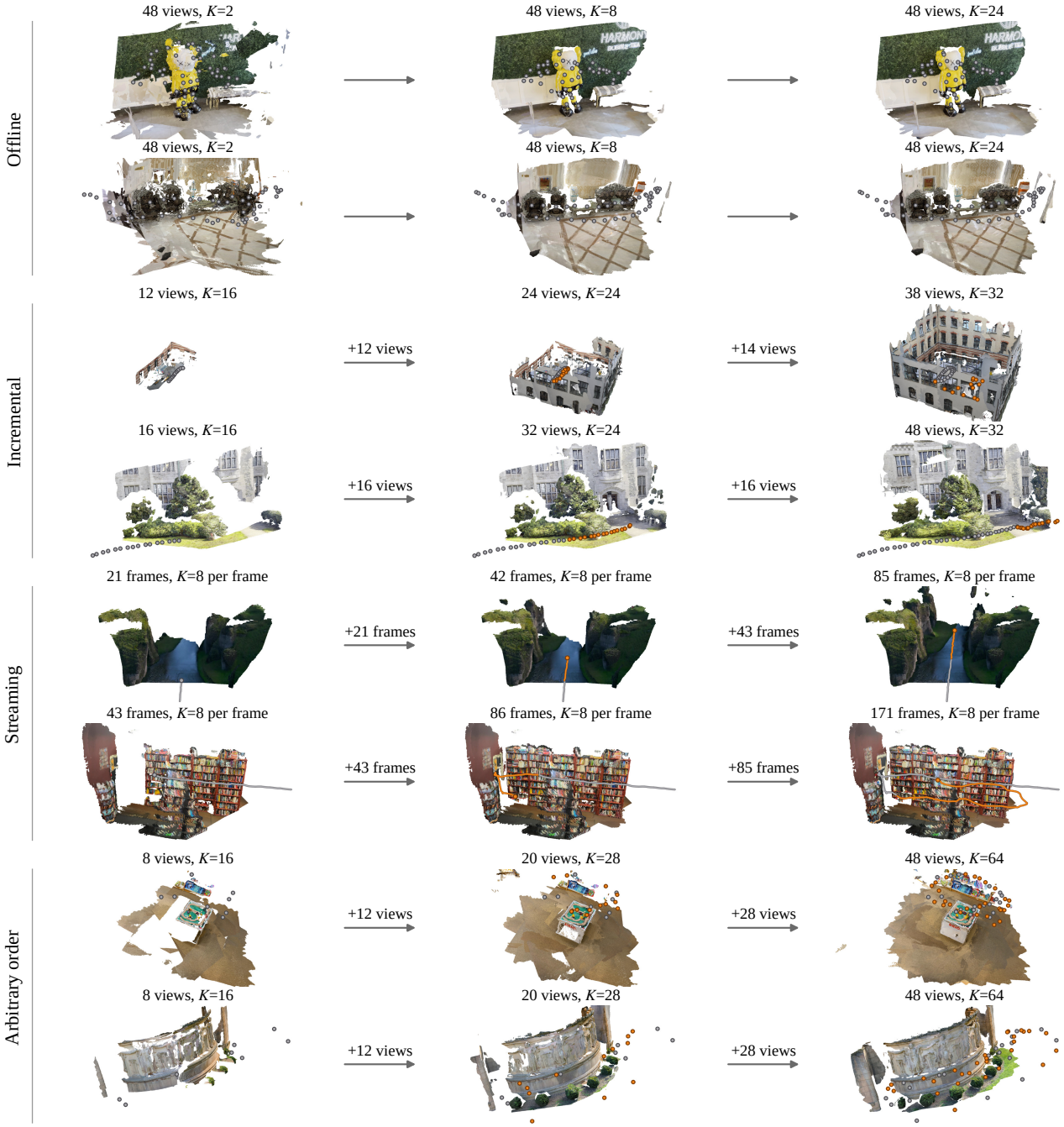


Figure 19. **Reconstructions that grow as views arrive.** Each row shows one scene at three stages of one arrival pattern. Offline solves all views jointly, incremental adds the capture in thirds, streaming adds video frames one at a time, and arbitrary order adds random groups of views. Views that join continue the running solve, and the views already in it keep their states. Above each panel are the number of views in the solve and the number K of solver calls run so far. Arrows give the views that joined, and orange marks their cameras.

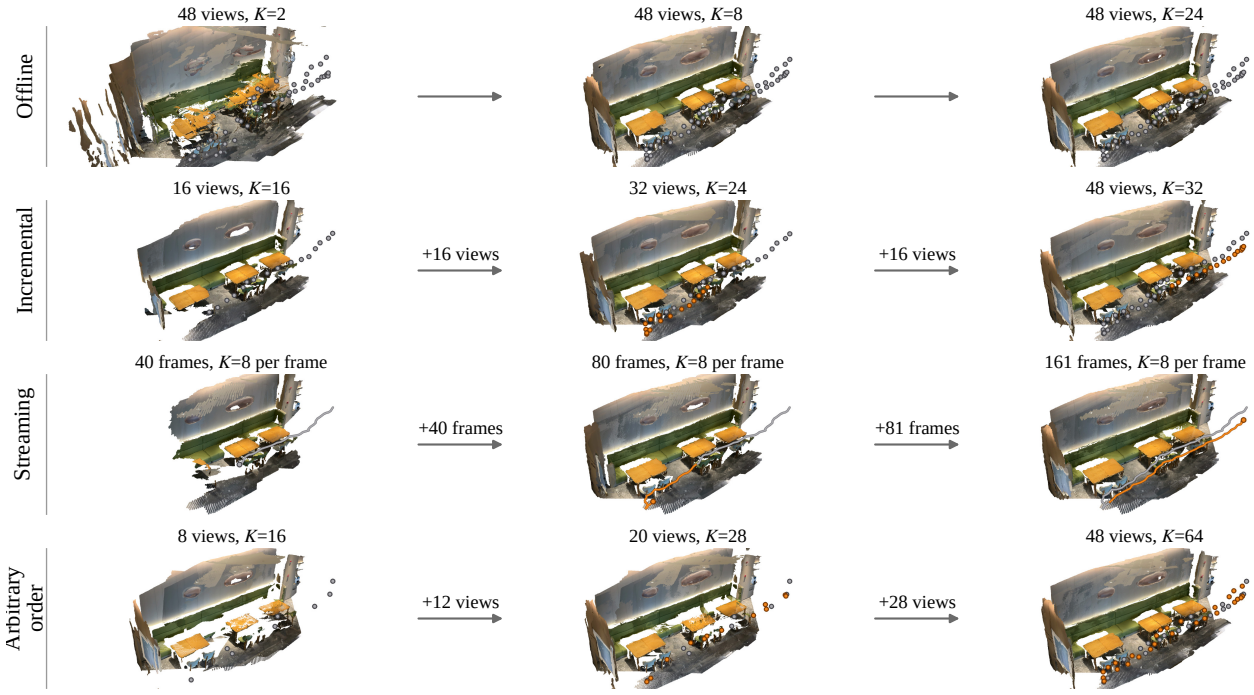


Figure 20. **One scene under the four arrival patterns.** Shown as in Fig. 19. All four patterns end in nearly the same reconstruction, although the views arrive in different orders and receive different numbers of calls.

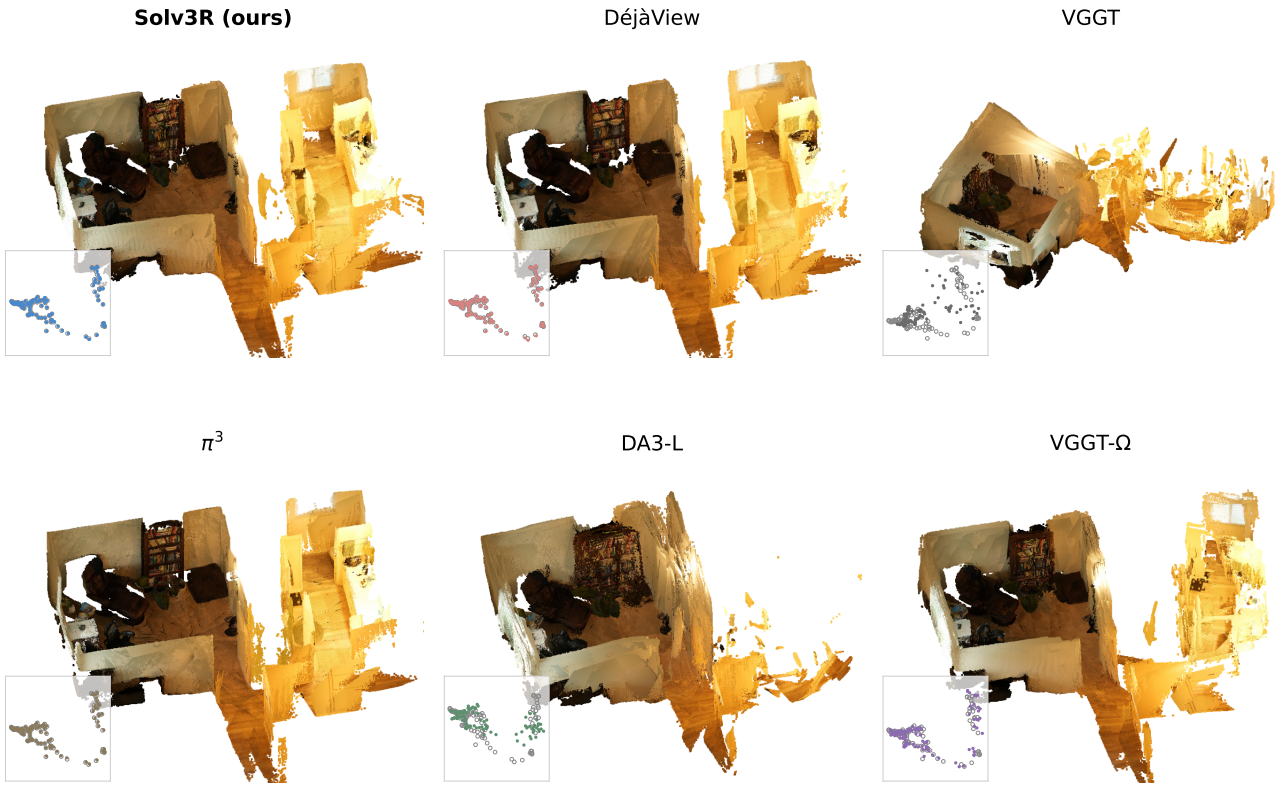


Figure 21. **Offline reconstruction comparison.** All methods see the same 100 images at once. VGGT, DA3-L, and VGGT-Ω misplace the second room, while Solv3R reconstructs both rooms in one piece.

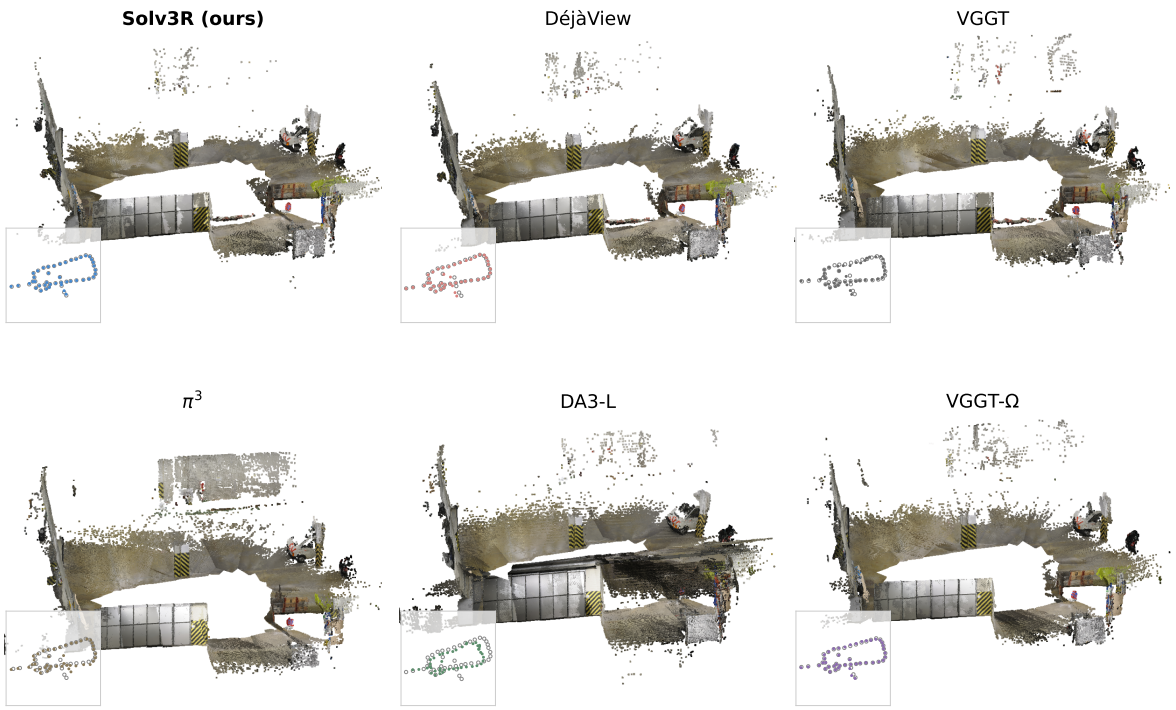


Figure 22. **Offline reconstruction comparison.** All methods see the same 44 images at once. π^3 and DA3-L misplace parts of the scene, while Solv3R keeps the walls and floor in place.

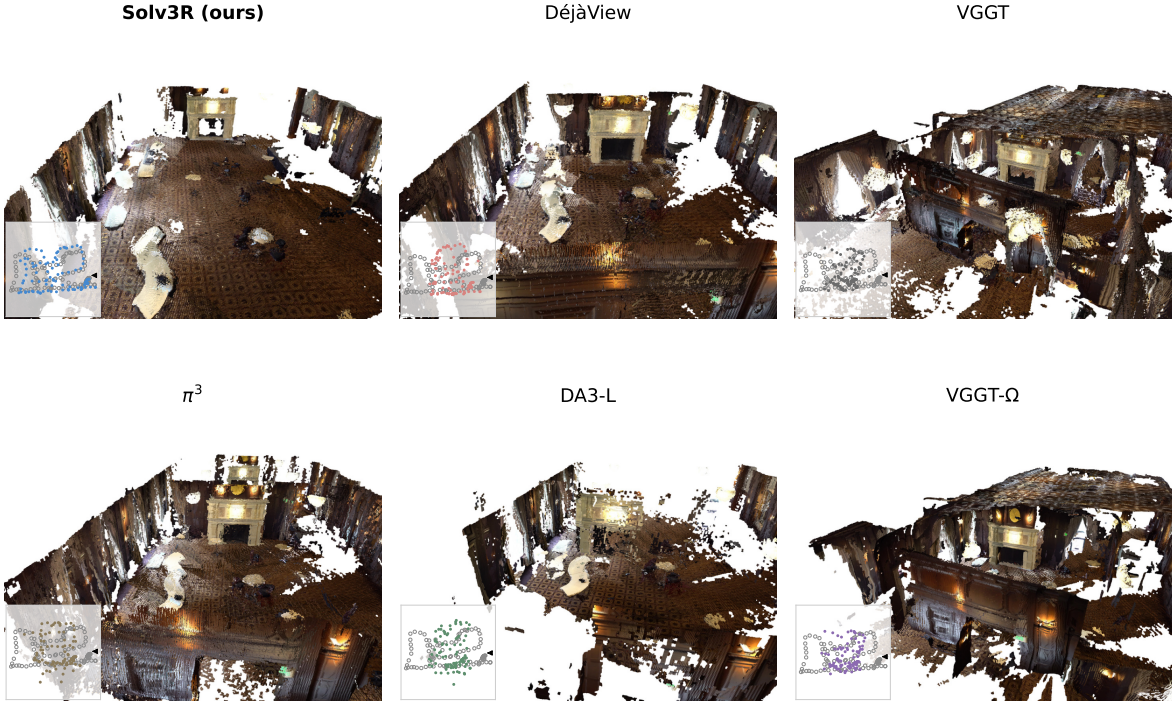


Figure 23. **Offline reconstruction comparison.** All methods see the same 100 images at once, and the ceiling is removed for display. Insets show the predicted cameras in color and the reference cameras in gray, seen from above. Solv3R places the cameras closest to the reference, while VGGT and VGGT-Ω fold the room.

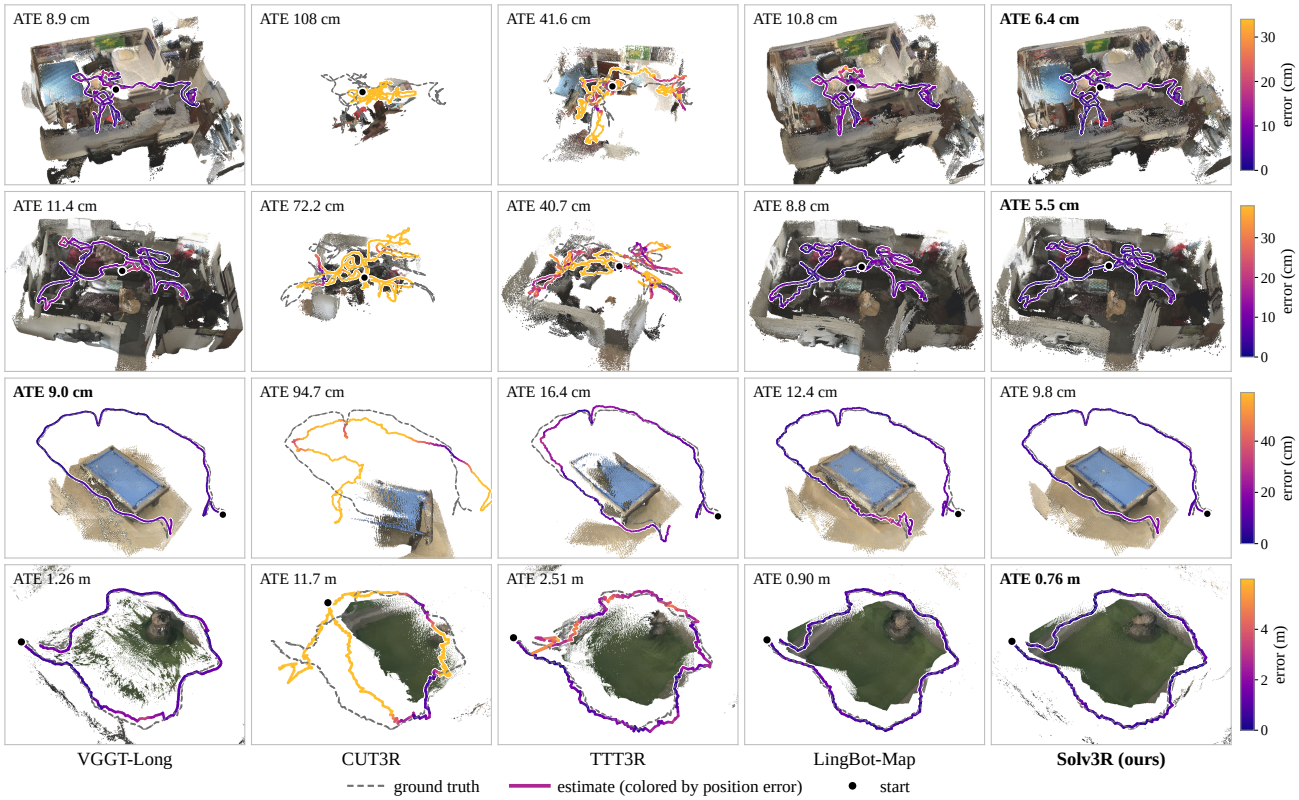


Figure 24. **Streaming reconstruction comparison.** The methods and settings are those of the streaming figure of the main paper. Camera trajectories are colored by position error.