MAP2SLAM: FROM STREAMING MAP PREDICTION TO REAL-TIME SLAM

Anonymous authors
Paper under double-blind review

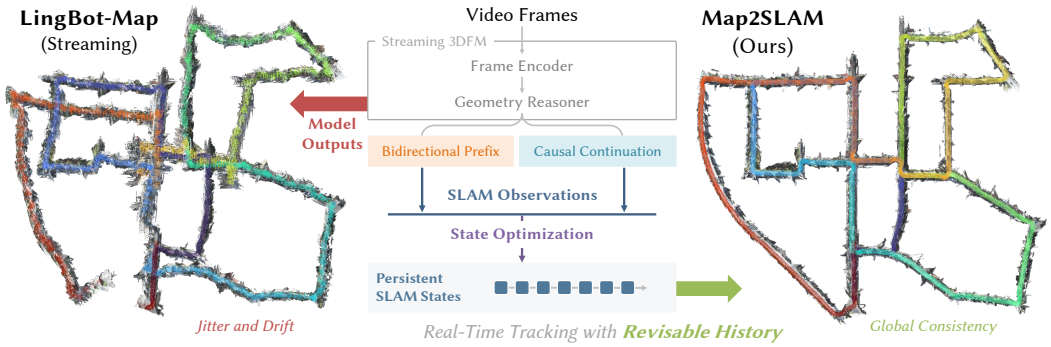


Figure 1: **From Streaming Map Prediction to SLAM.** Model outputs from LingBot-Map exhibit jitter and drift (left). Rather than treating these predictions as final estimates, Map2SLAM uses them as observations of persistent SLAM states (center). The Geometry Reasoner follows one prefix-causal attention architecture: selected history forms a fresh *Bidirectional Prefix* for re-observation, while *Causal Continuation* observes incoming frames. Both reuse features from a single Frame Encoder pass. These operations enable real-time tracking with a revisable history, yielding a globally consistent reconstruction (right).

ABSTRACT

Streaming 3D foundation models (3DFMs) estimate camera poses and dense geometry online, but they fall short of SLAM: their predictions are fixed once emitted, so loop closures cannot correct accumulated drift. With such models already providing strong geometric capability, a central question for SLAM becomes how to schedule their inference and how to keep its results as revisable state. We present Map2SLAM, which keeps LingBot-Map frozen, trains only a lightweight retrieval head, and builds SLAM on the model's inference interface, turning its predictions into observations of an external persistent SLAM state. The front-end tracks every incoming frame with the model's causal streaming inference and decides which frames are written to the persistent state as keyframes. The back-end asynchronously decides when and which past frames to re-observe, re-running the model's joint inference over them with fresh context; verified loop observations correct the trajectory online, and segment observations refine dense geometry after the sequence. All invocations share a single frame encoder pass per image, whose cached features also drive loop retrieval, so frequent re-invocation incurs no additional encoding cost. Map2SLAM achieves state-of-the-art performance on KITTI, VBR, and Oxford Spires, reducing mean ATE by over 40% compared with the best uncalibrated baselines, while maintaining real-time tracking at over 28 FPS at a resolution of 518×154 on an NVIDIA RTX 5090D GPU.

1 INTRODUCTION

Feed-forward 3D foundation models (3DFMs) infer camera poses and dense scene geometry directly from unposed images (Wang et al., 2024; 2025). Streaming 3DFMs extend this capability to online video, producing a geometric prediction for each incoming frame (Zhuo et al., 2026; Chen et al., 2026). However, these predictions are fixed once emitted: drift accumulates over long sequences,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

and even when the camera revisits a known place, earlier estimates cannot be corrected (Figure 1, left). Current streaming 3DFMs therefore cannot yet replace SLAM.

Prior work has combined 3DFMs with SLAM in several ways: aligning chunk-wise predictions (Deng et al., 2026; Lee et al., 2026b), using a 3DFM as the front-end of a classical SLAM pipeline (Murai et al., 2025; Maggio et al., 2025; Maggio & Carlone, 2026), or attaching loop closure to a streaming model (Cheng et al., 2026; Han et al., 2026). These methods invoke the model in a fixed pattern, over predefined chunks, frame pairs, or a single causal pass, rather than scheduling its inference. Given the strong geometric capability of current models, we argue that the key to building SLAM lies in how to schedule model inference and how to keep its results as revisable state. Just as LLM agents rely on a harness to schedule a fixed model, a 3DFM needs a similar mechanism to decide which frames enter its context, when to re-observe history, and which state each prediction updates.

Following this principle, we build Map2SLAM upon LingBot-Map (Chen et al., 2026) with a framework that schedules its inference and turns its predictions into persistent SLAM state, yielding accurate real-time SLAM. The prefix-causal structure of LingBot-Map supports two inference modes: *Bidirectional Prefix* jointly reasons over multiple frames, and *Causal Continuation* infers frame by frame autoregressively over long videos. Map2SLAM assigns the two modes to the two ends of the system: the front-end processes the live stream with Causal Continuation, and the back-end re-observes history with Bidirectional Prefix. The front-end decides which frames enter the context and are written to the state. To avoid the trajectory jitter caused by an ever-growing context, it tracks within bounded windows and treats each Causal Continuation step as a *Keyframe Probe*: based on the projective motion induced by the predicted geometry, an accepted frame is committed as a keyframe without repeated inference, and a rejected frame has its KV update rolled back.

Windowed tracking still accumulates drift over long sequences. The back-end revises the state asynchronously, deciding when and which past frames to re-observe. Whereas LingBot-Map uses Bidirectional Prefix only for initialization, the back-end re-invokes it with fresh context. *Loop Retrieval* proposes revisit candidates from cached features. Because appearance matches do not guarantee geometric consistency, *Pairwise Geometry* verifies each candidate with a two-view prefix, and verified constraints enter *Incremental Pose Graph Optimization* (iPGO), which corrects the trajectory online. The back-end also re-observes sequential and loop segments with multi-view prefixes, whose results feed *Dense Geometric Refinement* after the sequence ends. All invocations share a single *Frame Encoder* pass per image: both the downstream Geometry Reasoner and the retrieval head reuse its cached features, so frequent re-invocation incurs no additional encoding cost.

We evaluate Map2SLAM on KITTI (Geiger et al., 2012), VBR (Brizi et al., 2024), and Oxford Spires (Tao et al., 2026). Map2SLAM achieves state-of-the-art performance on KITTI, VBR, and Oxford Spires, reducing mean ATE by more than 40% relative to the best-performing uncalibrated baselines, while running in real time at over 28 FPS with an input resolution of 518×154 on an NVIDIA RTX 5090D GPU.

Our contributions are threefold:

- We propose a design principle for building SLAM on streaming 3DFMs: given their strong geometric capability, the key lies in scheduling model inference and managing the state formed by its results.
- We present Map2SLAM, which realizes this principle on a frozen LingBot-Map: a causal front-end writes the state through commit-or-rollback Keyframe Probes, and an asynchronous back-end revises it by re-invoking Bidirectional Prefix with fresh context, with all invocations sharing one Frame Encoder pass per image.
- Map2SLAM achieves the lowest mean ATE among uncalibrated methods on KITTI, VBR, and Oxford Spires, 42.7–48.3% below the second-best results, while tracking at over 28 FPS on an NVIDIA RTX 5090D GPU.

2 Related Work

Long-Sequence Visual SLAM Visual SLAM has evolved from feature-based systems (Klein & Murray, 2007; Mur-Artal et al., 2015), through direct methods (Engel et al., 2014; 2018), to learned

geometric frontends (Teed & Deng, 2021; Teed et al., 2023). Long-sequence SLAM, however, requires more than estimating motion as frames arrive. It must retain past estimates in its state so that a loop can introduce long-range constraints and revise the trajectory (Mur-Artal et al., 2015; Campos et al., 2021; Engel et al., 2014; Gao et al., 2018; Lipson et al., 2024).

3D Foundation Models Recent 3D foundation models (3DFMs) infer camera and scene geometry in a feed-forward pass over image collections (Wang et al., 2024; Zhang et al., 2025; Wang et al., 2025; 2026; Fang et al., 2026a; Gao et al., 2026). Streaming 3DFMs extend this inference to video frames as they arrive (Zhuo et al., 2026; Chen et al., 2026; Cheng et al., 2026; Han et al., 2026), enabling geometric prediction over increasingly long sequences. In particular, LingBot-Map (Chen et al., 2026) first processes an initial set of frames jointly to establish scene context, and then incrementally incorporates subsequent frames in an autoregressive manner.

Long-Sequence 3D Reconstruction Recent work extends 3DFMs to long sequences by composing local predictions into globally consistent reconstructions (Deng et al., 2026; Lee et al., 2026b; Zhang et al., 2026c; Ding et al., 2026; Zhang et al., 2026a; Xiong et al., 2026). Related efforts increasingly integrate 3DFM predictions with incremental estimation, loop closure, and global optimization (Murai et al., 2025; Maggio et al., 2025; Maggio & Carlone, 2026; Mandal et al., 2026; Jeon et al., 2026; Lee et al., 2026a; Fang et al., 2026b). Together, these developments are carrying long-sequence reconstruction beyond a purely offline setting and progressively closer to the system-level concerns of SLAM.

3 METHOD

Map2SLAM turns streaming geometric predictions into observations that update persistent trajectory and dense-geometry states. It builds on LingBot-Map (Chen et al., 2026), which first processes an initial frame set jointly with bidirectional attention for coordinate grounding and then handles later frames causally, one at a time, for online inference. We call these executions *Bidirectional Prefix* and *Causal Continuation*; together, they form a *prefix-causal architecture* (Dong et al., 2019). Map2SLAM uses Causal Continuation for live tracking and runs Bidirectional Prefix again on selected historical frames to produce fresh geometric observations for optimization.

3.1 SYSTEM OVERVIEW

In our Map2SLAM system, Front-end tracking and Back-end re-observation run asynchronously on separate GPUs. The *Frame Encoder* processes each image once: the Front-end consumes its geometric features immediately, while Back-end queries reuse cached outputs from the same pass. Within each tracking window, the Front-end maintains one active *Geometry Reasoner* context; each Back-end geometric query starts from a fresh context. Figure 2 summarizes this architecture.

3.2 FRONT-END TRACKING

Extending LingBot-Map's streaming context over long sequences leads to trajectory *jitter*. We therefore track in bounded windows and reinitialize the active *Geometry Reasoner* context at each boundary. This bounds the context and stabilizes local tracking, but introduces drift between windows.

To initialize the first window, we process its first N_p frames as a Bidirectional Prefix, establishing the initial tracking context. For each subsequent frame, one Causal Continuation step tentatively extends the active context to form a Keyframe Probe, yielding the pose, depth, and intrinsics used to measure projective motion. Let $k(i)$ denote the latest keyframe before frame I_i. We measure the projective motion induced by the probe as

$$m_i = \frac{1}{|\Omega_{k(i)}|} \sum_{\mathbf{u} \in \Omega_{k(i)}} \left\| \pi\big(K_i T_{i \leftarrow k(i)} \pi^{-1}\big(\mathbf{u}, D_{k(i)}(\mathbf{u}), K_{k(i)}\big)\big) - \mathbf{u} \right\|_2, \tag{1}$$

where $\mathbf{u}$ denotes a pixel coordinate in keyframe $k(i)$; T_i, K_i, and D_i are the predicted camera pose, intrinsics, and depth, respectively; π and π^{-1} denote perspective projection and back-projection; $\Omega_{k(i)}$ contains valid pixels of keyframe $k(i)$; and $T_{i \leftarrow k(i)} = T_i^{-1} T_{k(i)}$. If $m_i > \tau$, the front-end

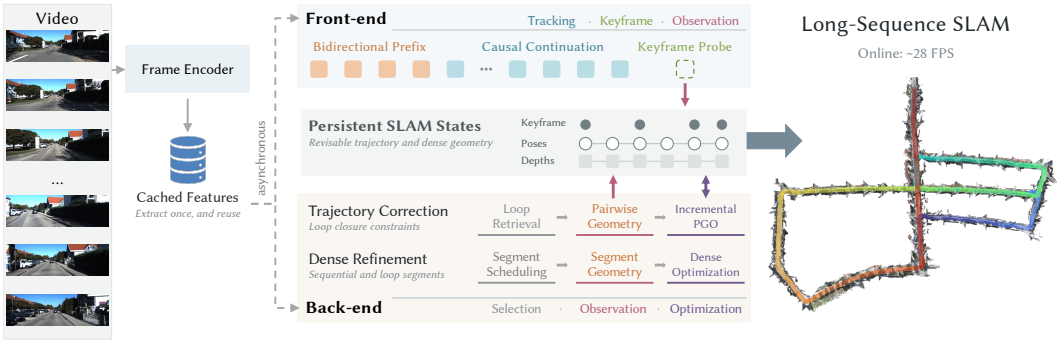


Figure 2: **System overview.** The Frame Encoder processes each image once, and cached outputs are reused throughout. The Front-end initializes each tracking window with a Bidirectional Prefix, tracks through Causal Continuation, and admits keyframes through Keyframe Probe. Each Back-end path performs selection, fresh-context re-observation, and optimization. Loop Retrieval, Pairwise Geometry, and iPGO correct the trajectory online; Segment Scheduling, Segment Geometry, and Dense Optimization refine keyframe poses and depths after the sequence. Final keyframe corrections propagate to non-keyframes.

accepts the candidate frame as a keyframe and commits its KV cache update; otherwise, it rejects the frame and rolls back the update.

When the context reaches its budget N_b, accepting a frame triggers context reinitialization rather than a commit. The front-end clears the KV cache and forms the successor Bidirectional Prefix from the accepted frame and the last $N_p - 1$ keyframes. This prefix has its own similarity gauge, which we align to the existing trajectory using the replayed keyframes: their depth ratios determine the scale s_w, while their poses determine the rotation R_w and translation t_w. Each local prediction $(\widetilde{R}_i, \widetilde{t}_i, \widetilde{D}_i)$ in the successor context is then mapped as

$$R_i = R_w \widetilde{R}_i, \qquad t_i = s_w R_w \widetilde{t}_i + t_w, \qquad D_i = s_w \widetilde{D}_i. \tag{2}$$

This alignment preserves cross-window continuity for subsequent causal outputs.

For each accepted keyframe, the Front-end records its pose relative to the preceding accepted keyframe as a sequential observation. For each non-keyframe, it retains the causal pose relative to the latest keyframe for final trajectory recovery. Online Trajectory Correction combines the sequential observations with verified loop constraints to correct accumulated long-range drift.

3.3 Back-end Optimization

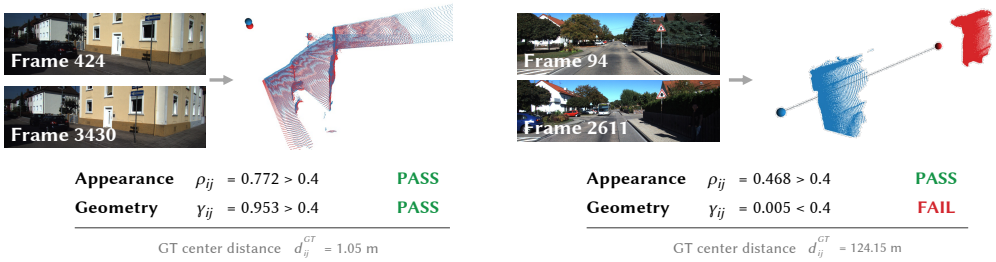
The Back-end comprises two paths with the same three stages: *selection*, *geometric re-observation*, and *optimization*. Online Trajectory Correction updates the trajectory while tracking continues; Dense Geometric Refinement jointly refines keyframe poses and depths after the sequence.

3.3.1 Online Trajectory Correction

Loop Retrieval Although LingBot-Map is trained to predict dense geometry, we find that intermediate tokens from its Frame Encoder remain effective for visual place recognition. We add a lightweight SALAD aggregation head (Izquierdo & Civera, 2024) to these cached tokens while keeping the encoder frozen. The head produces a global descriptor for each accepted keyframe, and descriptor similarity proposes historical loop candidates for Pairwise Geometry.

To control the number of redundant loop candidates, each descriptor becomes searchable only after $Q = 64$ subsequent keyframes have been accepted, and each query retains up to $K = 5$ loop candidates with SALAD similarity of at least $\tau_s = 0.40$. Once a pair is verified, we suppress further candidates between the accepted-keyframe neighborhoods of radius $r = 5$ around its two endpoints.

Pairwise Geometry Pairwise Geometry consists of *relative-geometry estimation* followed by *geometric consistency verification*. For each loop candidate (i, j), a fresh two-view Bidirectional Prefix



(a) Accepted after geometric verification. (b) Rejected by geometric verification.

Figure 3: **Loop Retrieval followed by Pairwise Geometry** ($\tau_\rho = \tau_\gamma = 0.4$). Both pairs pass appearance retrieval; fresh two-view geometric re-observation accepts (a) and rejects (b). Ground-truth camera-center distance is shown only for reference and is not used by the method.

over cached frame features jointly predicts the two camera poses, depth maps, and intrinsics. We then verify the predicted geometry through bidirectional depth consistency:

$$\gamma_{i\to j} = \frac{1}{|\Omega_i|}\sum_{\mathbf{p}\in\Omega_i} \mathbf{1}[e_{i\to j}(\mathbf{p}) < \epsilon], \qquad e_{i\to j}(\mathbf{p}) = \frac{\left|\hat{d}_{i\to j}(\mathbf{p}) - d_j(\pi_{i\to j}(\mathbf{p}))\right|}{d_j(\pi_{i\to j}(\mathbf{p}))}. \tag{3}$$

Here, Ω_i contains source pixels with valid depth, while $\pi_{i\to j}(\mathbf{p})$ and $\hat{d}_{i\to j}(\mathbf{p})$ denote the projected location and depth of $\mathbf{p}$ in view j, respectively. A projected depth is consistent when its relative error is below $\epsilon = 0.10$; invalid projections are counted as inconsistent. We use $\gamma_{ij} = \min\{\gamma_{i\to j}, \gamma_{j\to i}\}$ as the pairwise consistency score. If $\gamma_{ij} \geq \tau_g$, with $\tau_g = 0.40$, the candidate yields a verified loop constraint for iPGO; otherwise, it is discarded, as shown in Figure 3.

Incremental Pose Graph Optimization At time t, iPGO operates on the ordered accepted keyframes $\mathcal{K}_t = (k_0, \ldots, k_{N_t-1})$ and the loop candidates $\mathcal{C}_t$ proposed up to that time. Causal Tracking and Pairwise Geometry provide the sequential and verified-loop constraint sets

$$\mathcal{E}_{\text{seq},t} = \{(k_{n-1}, k_n)\}_{n=1}^{N_t-1}, \qquad \mathcal{E}^{\text{ver}}_{\text{loop},t} = \{(i,j) \in \mathcal{C}_t \mid \gamma_{ij} \geq \tau_g\}. \tag{4}$$

iPGO optimizes $\mathcal{S}_t = \{\mathbf{S}_i\}_{i\in\mathcal{K}_t}$, where $\mathbf{S}_i \in \mathrm{Sim}(3)$. Each constraint (i,j) supplies a relative measurement $\hat{\mathbf{S}}_{ij} \in \mathrm{Sim}(3)$. The $\mathrm{Sim}(3)$ representation allows joint correction of scale, rotation, and translation; fixing $\mathbf{S}_{k_0}$ to its initial value removes the global gauge freedom.

Because Pairwise Geometry verifies each candidate independently, verified loop constraints may still conflict with the current trajectory. We jointly evaluate them against the sequential constraints using GNC-TLS (Yang et al., 2020) in riSAM's incremental scheme (McGann et al., 2023), treating the sequential constraints as known inliers. Each verified loop receives a robust weight $w_{ij,t} \in [0,1]$, and the active set is $\mathcal{E}^{\text{act}}_{\text{loop},t} = \{(i,j) \in \mathcal{E}^{\text{ver}}_{\text{loop},t} \mid w_{ij,t} \geq 0.5\}$. As new constraints arrive, the selector updates these weights without changing the loop measurements, allowing loops to be activated, withdrawn, or reactivated.

The sequential and active loop constraints enter a stateful iSAM2 graph (Kaess et al., 2012), which estimates the trajectory by minimizing

$$\mathcal{S}_t^\star = \arg\min_{\mathcal{S}_t} \sum_{(i,j)\in\mathcal{E}_{\text{seq},t}\cup\mathcal{E}^{\text{act}}_{\text{loop},t}} \rho_{\text{H}}\left(d_{\mathrm{Sim}(3)}\left(\hat{\mathbf{S}}_{ij}, \mathbf{S}_{ij}(\mathcal{S}_t)\right)\right). \tag{5}$$

Here, $\mathbf{S}_{ij}(\mathcal{S}_t)$ is the relative transformation implied by the current keyframe states, $d_{\mathrm{Sim}(3)}$ compares it with the measurement $\hat{\mathbf{S}}_{ij}$, and ρ_{H} is the Huber loss. As keyframes and active-set changes arrive, iSAM2 adds new states and sequential constraints and adds or removes loop constraints, revising the trajectory without rebuilding the graph.

3.3.2 Dense Geometric Refinement

Segment Scheduling We first group accepted keyframes for historical re-observation into two types of segments. For *Sequential Segments*, we slide a window of $N_s = 24$ keyframes with an

overlap of $N_o = 6$ over the accepted-keyframe order. For a loop pair (i, j), $i < j$, we form a *Loop Segment* by concatenating $N_s/2$ keyframes ending at k_j with $N_s/2$ keyframes starting at k_i. This ordering places the verified revisit at one adjacent seam in a virtual capture sequence. Sequential Segments become available upon window completion and Loop Segments upon loop verification.

Segment Geometry For each available segment, we run a fresh Bidirectional Prefix over its cached features. Unlike Causal Continuation, every keyframe can attend to all other keyframes in the segment, providing full multi-view context. Each execution outputs local poses, per-pixel depths, intrinsics, and confidence maps. These service as one segment observation. Reusing the cached features avoids another Frame Encoder pass, and these independent executions can overlap with Front-end Causal Tracking.

Dense Optimization After the input sequence ends and iPGO produces its final active set, we retain all Sequential Segment observations and the Loop Segment observations associated with this set, denoted by $\mathcal{O}$. We initialize one shared pose T_i per keyframe from the final iPGO trajectory and optimize these poses together with one log-depth field z_i per keyframe and shared camera parameters $\boldsymbol{\alpha}$. For each retained segment s, we fit a $\mathrm{Sim}(3)$ alignment A_s from its prefix-local poses to their iPGO counterparts and keep A_s fixed. The shared 3D point at pixel $\mathbf{u}$ is $\mathbf{X}_i(\mathbf{u}) = T_i \pi^{-1}(\mathbf{u}, \exp z_i(\mathbf{u}), K(\boldsymbol{\alpha}))$.

Following confidence-weighted 3D pointmap alignment (Wang et al., 2024), we optimize

$$\begin{aligned} \min_{\{T_i, z_i\}, \boldsymbol{\alpha}} \quad & \sum_{\substack{s \in \mathcal{O},\, i \in \mathcal{K}_s \\ \mathbf{u} \in \Omega_{si}}} w_{si}(\mathbf{u})\, \rho_\delta(\|\mathbf{r}_{si}(\mathbf{u})\|_2)\,, \\ \mathbf{r}_{si}(\mathbf{u}) &= \frac{A_s^{-1}\mathbf{X}_i(\mathbf{u}) - \widetilde{\mathbf{X}}_{si}(\mathbf{u})}{\sigma_r \widetilde{\ell}_{si}(\mathbf{u})}. \end{aligned} \tag{6}$$

Here, $\mathcal{K}_s$ contains the keyframes in segment s, and Ω_{si} contains valid positive-depth predictions with confidence at least one. The weight w_{si} is the predicted log confidence normalized by the number of retained segments covering keyframe i. We set $\sigma_r = 0.1$ and use the Huber loss with $\delta = 1.5$. We fix the first keyframe pose, solve the remaining variables with Levenberg–Marquardt, and eliminate the per-pixel log depths at each iteration using an exact Schur complement. The optimized state yields refined keyframe poses, dense depths, and shared camera intrinsics.

3.4 Pose Correction Propagation

The back-end refines only keyframes, while non-keyframes retain their Causal Tracking poses relative to the preceding keyframe. The change from each keyframe's causal pose to its final pose defines a Sim(3) correction. We interpolate these corrections between adjacent keyframes and apply them to intermediate non-keyframes, recovering the full trajectory without additional model inference.

4 Experiments

4.1 Benchmark Evaluation

We evaluate trajectory accuracy on KITTI Odometry (Geiger et al., 2012), VBR (Brizi et al., 2024), and Oxford Spires (Tao et al., 2026), using ATE after a global $\mathrm{Sim}(3)$ alignment. We report per-sequence ATE and the mean across sequences, grouping methods into calibrated and uncalibrated settings. Methods with frequent tracking failures are not reported on the corresponding benchmarks: ORB-SLAM3 is reported only on KITTI, while MASt3R-SLAM (Murai et al., 2025) is omitted from all three. VGGT-SLAM 2.0 is included for reference only, as it reports poses at keyframes.

As shown in Figure 4(d–f), LingBot-Map(S) exhibits severe local jitter, whereas LingBot-Map(W) produces a smoother trajectory but accumulates cross-window drift that distorts the global route. Our system suppresses both effects and closely follows the ground truth. This improvement is also reflected quantitatively, with substantially lower ATE on KITTI, VBR, and Oxford Spires. These results suggest that model capability is only part of the equation; how its predictions are used matters too.

Table 1: Absolute trajectory error (ATE, m) on KITTI (Geiger et al., 2012). "–" denotes tracking failure. LC, Dense Geo., and Frame-wise denote loop closure, dense geometry, and per-frame tracking, respectively. LingBot-Map(S/W) denote the streaming/window modes. † VGGT-SLAM 2.0 uses native keyframes only and is excluded from ranking. The best and second-best results in each section are shown in bold and underlined, respectively.

Method	LC	Dense Geo.	Frame-wise	00 *4541 f 3.70 km*	01 *1101 f 2.50 km*	02 *4661 f 5.10 km*	03 *801 f 0.60 km*	04 *271 f 0.40 km*	05 *2761 f 2.20 km*	06 *1101 f 1.20 km*	07 *1101 f 0.70 km*	08 *4071 f 3.20 km*	09 *1591 f 1.70 km*	10 *1201 f 0.90 km*	Avg. ↓
Calibrated															
ORB-SLAM3 (Campos et al., 2021)	✓		✓	**7.77**	–	**25.18**	**0.85**	**0.72**	6.03	16.93	**2.57**	**64.55**	**6.76**	**8.48**	–
DROID-SLAM (Teed & Deng, 2021)	✓	✓	✓	110.19	90.94	138.72	3.20	1.47	73.64	61.50	16.09	103.85	92.01	22.21	64.89
DPV-SLAM++ (Lipson et al., 2024)	✓		✓	7.87	**27.48**	71.75	2.56	1.03	**5.92**	**12.49**	18.00	113.94	74.54	13.65	**31.75**
Uncalibrated															
DROID-Calib (Hagemann et al., 2023)		✓	✓	47.65	34.21	93.95	5.78	2.39	39.74	11.29	18.62	76.32	68.26	8.79	37.00
VGGT-Long (Deng et al., 2026)	✓	✓		10.83	66.41	55.69	5.57	3.95	10.13	6.95	2.83	57.59	31.21	22.16	24.85
SwiftVGGT (Lee et al., 2026b)	✓	✓		9.30	113.82	56.29	10.39	5.43	11.82	8.35	24.56	68.90	31.08	25.07	33.18
VGGT-SLAM 2.0† (Maggio & Carlone, 2026)	✓	✓		104.44	151.66	146.87	12.92	2.92	10.36	28.58	18.25	76.39	28.84	24.04	55.02
LoGeR (Zhang et al., 2026b)		✓		62.93	41.68	39.18	4.87	1.82	41.09	13.89	16.18	26.43	22.82	8.81	25.43
LingBot-Map(S) (Chen et al., 2026)		✓	✓	31.33	102.42	56.71	3.93	1.27	23.59	8.33	8.63	19.09	21.50	12.52	26.30
LingBot-Map(W) (Chen et al., 2026)		✓	✓	52.46	106.71	63.17	3.29	2.06	30.39	8.39	9.69	50.95	54.84	8.99	35.54
HorizonStream (Cheng et al., 2026)	✓	✓	✓	15.38	23.89	70.80	5.56	**0.71**	7.94	7.73	3.16	22.29	27.21	12.29	17.91
ABot-Recon (Han et al., 2026)	✓	✓	✓	11.67	**21.49**	39.82	4.36	2.90	5.91	8.65	**1.66**	22.39	14.04	16.22	13.56
Ours	✓	✓	✓	**3.51**	26.94	**11.02**	**1.92**	1.02	**3.40**	**2.06**	2.24	**16.18**	**9.65**	**5.24**	**7.56**

Table 2: Absolute trajectory error (ATE, m) on VBR (Brizi et al., 2024). "–" denotes tracking failure. LC, Dense Geo., and Frame-wise denote loop closure, dense geometry, and per-frame tracking, respectively. LingBot-Map(S/W) denote the streaming/window modes. † VGGT-SLAM 2.0 uses native keyframes only and is excluded from ranking. The best and second-best results in each section are shown in bold and underlined, respectively.

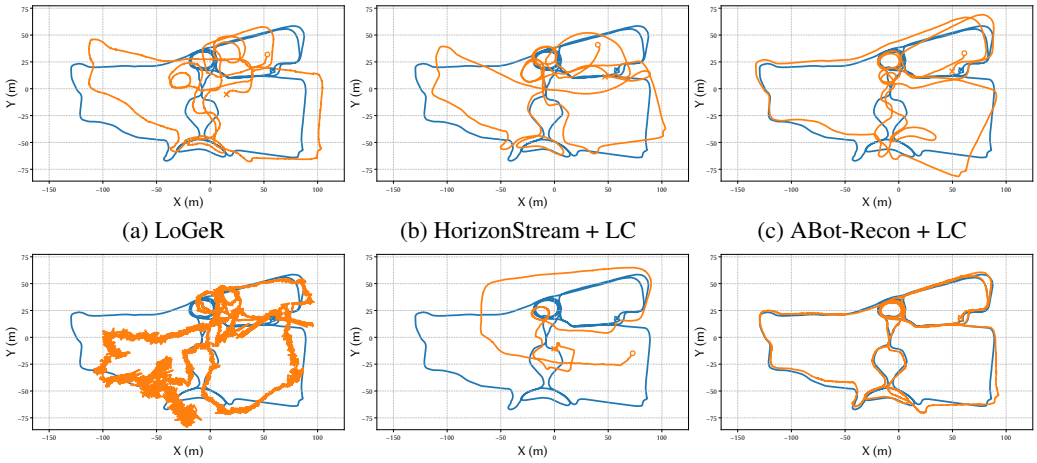
Method	LC	Dense Geo.	Frame-wise	Colosseo-0 *8815 f 1.45 km*	Campus-0 *12042 f 2.73 km*	Campus-1 *11671 f 2.95 km*	Pincio-0 *11142 f 1.27 km*	Spagna-0 *14141 f 1.56 km*	Diag-0 *10021 f 1.02 km*	Ciampino-1 *18846 f 5.20 km*	Avg. ↓
Calibrated											
DROID-SLAM (Teed & Deng, 2021)	✓	✓	✓	76.66	125.77	83.87	49.85	59.61	35.81	145.28	82.41
DPV-SLAM++ (Lipson et al., 2024)	✓		✓	**18.59**	**21.69**	**22.22**	**39.37**	**37.31**	**33.51**	**45.56**	**31.18**
Uncalibrated											
DROID-Calib (Hagemann et al., 2023)		✓	✓	27.36	76.05	68.52	14.09	25.38	–	124.98	–
VGGT-Long (Deng et al., 2026)	✓	✓		44.13	132.70	115.62	64.29	59.05	33.66	179.41	89.84
SwiftVGGT (Lee et al., 2026b)	✓	✓		21.24	131.36	115.94	65.80	58.43	34.52	176.58	86.27
VGGT-SLAM 2.0† (Maggio & Carlone, 2026)	✓	✓		18.75	26.88	22.90	33.25	10.17	6.47	49.80	24.03
LoGeR (Zhang et al., 2026b)		✓		25.51	19.69	29.23	27.91	22.87	34.99	93.04	36.18
LingBot-Map(S) (Chen et al., 2026)		✓	✓	14.85	20.30	9.92	30.46	23.71	22.68	65.09	26.71
LingBot-Map(W) (Chen et al., 2026)		✓	✓	86.86	123.09	104.64	63.40	56.68	23.68	82.20	77.22
HorizonStream (Cheng et al., 2026)	✓	✓	✓	25.00	28.79	10.54	15.49	21.62	7.44	18.36	18.18
ABot-Recon (Han et al., 2026)	✓	✓	✓	16.70	8.00	4.30	**5.12**	17.91	9.69	8.09	9.97
Ours	✓	✓	✓	**7.54**	**4.28**	**2.45**	6.57	**3.38**	**4.67**	**7.18**	**5.15**

Against existing uncalibrated methods, our system achieves the lowest mean ATE on all three benchmarks, improving over ABot-Recon by more than 40% in each case. This advantage is consistent across sequences, with our system ranking first or second on 26 of the 28 sequences. Despite using no camera intrinsics, it also outperforms the best complete calibrated baseline on KITTI and VBR; on Oxford Spires, DPV-SLAM++ remains better at $2.28\,\mathrm{m}$ versus $3.07\,\mathrm{m}$.

4.2 System Ablations

Pairwise Geometry Geometric consistency verification is critical for filtering false loop candidates before iPGO. As shown in Figure 5(a), without this check, adding Loop Retrieval and iPGO increases the mean ATE from $15.277\,\mathrm{m}$ to $71.825\,\mathrm{m}$; with Pairwise Geometry, it drops to $7.534\,\mathrm{m}$.

GNC-TLS Pairwise-verified loops can still conflict globally. Although GNC-TLS has little effect on the KITTI mean ATE ($7.534 \rightarrow 7.568\,\mathrm{m}$), it prevents the trajectory failure in the VBR *diag_train0* stress case in Figure 5(b). Its benefit therefore lies in robustness to conflicting loop constraints rather than average accuracy.



(a) LoGeR (b) HorizonStream + LC (c) ABot-Recon + LC

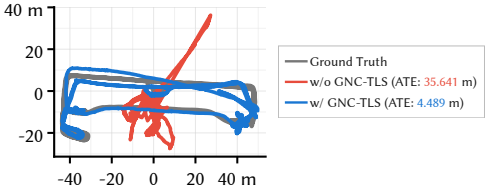
(d) LingBot-Map (S) (e) LingBot-Map (W) (f) Ours

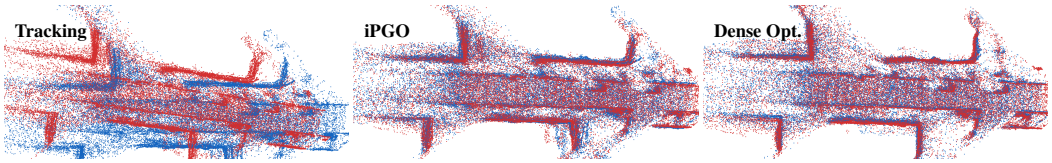
Figure 4: **Trajectory comparison on VBR *Spagna-0*.** The sequence contains 14,141 frames and covers 1.56 km. Blue and orange denote the reference and estimated trajectories, respectively.

(a) System component ablation on KITTI

Configuration	Geometric Check	GNC-TLS	Dense Opt.	Segment Sim(3)	ATE (m) ↓	RPE$_t$ (m) ↓	RPE$_r$ (deg) ↓
Causal Tracking	–	–	–	–	15.277	0.672	0.446
+ Retrieval + iPGO	×	×	–	–	71.825	2.919	0.602
+ Geometric Check	✓	×	–	–	7.534	0.605	0.424
+ GNC-TLS	✓	✓	–	–	7.568	0.604	0.423
+ Dense Opt.	✓	✓	✓	Free	12.962	**0.289**	**0.302**
Full method	✓	✓	✓	Fixed	**7.489**	0.296	0.306

(b) Robust loop selection with GNC-TLS





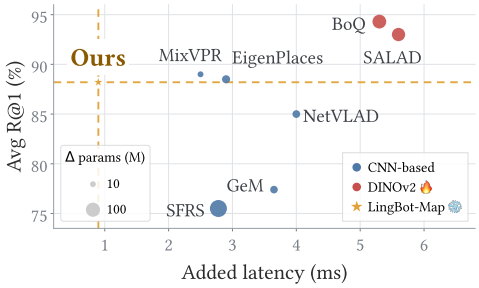
(c) Loop-region geometry across system stages

Figure 5: **System Ablations. (a)** Mean results on KITTI sequences 00–10 show that Geometric Check filters false loop candidates, while fixing segment alignments to the solution from Incremental Pose Graph Optimization (iPGO) recovers low ATE and retains low RPE. **(b)** On VBR *diag_train0*, GNC-TLS avoids the trajectory failure observed without it, reducing ATE from $35.641\,\mathrm{m}$ to $4.489\,\mathrm{m}$. **(c)** Across Tracking, iPGO, and Dense Pose–Depth Optimization (Dense Opt.), the same loop-region point samples from the earlier (blue) and later (red) visits are overlaid in a common coordinate frame.

Dense Geometric Refinement Free segment alignment substantially reduces translational and rotational RPE from $0.604\,\mathrm{m}/0.423°$ to $0.289\,\mathrm{m}/0.302°$, but degrades the mean ATE from $7.568\,\mathrm{m}$ to $12.962\,\mathrm{m}$, showing that better local geometry can come at the expense of global consistency. Fixing the iPGO-conditioned alignments restores the mean ATE to $7.489\,\mathrm{m}$ while retaining low RPE at $0.296\,\mathrm{m}/0.306°$. Dense refinement is applied only at sequence end; online global correction is provided by iPGO.

4.3 Analysis of the Loop Retrieval Descriptor

Our retrieval descriptor reuses cached token features captured during LingBot-Map's Frame Encoder pass. A 1.8M-parameter SALAD aggregation head converts these tokens into a global descriptor (Izquierdo & Civera, 2024). For 322×322 input images, this conversion takes only $0.94\,\mathrm{ms}$. We train only this head on GSV-Cities (Ali-Bey et al., 2022) for 10 epochs at 224×224 resolution while keeping the Frame Encoder frozen. Training takes approximately 2 hours on one NVIDIA RTX 5090D GPU with mixed precision. The resulting descriptor achieves a mean Recall@1 (R@1)



(a) Latency–accuracy trade-off

Block	MSLS	Pitts	SPED	Mean
17	86.76	**93.51**	81.38	87.22
18	87.84	93.24	**83.53**	**88.20**
19	**87.97**	93.21	83.20	88.13
20	87.84	93.06	82.70	87.87
21	87.84	93.03	81.22	87.36
22	86.62	92.77	80.56	86.65
23	85.68	92.40	78.42	85.50
24	84.73	92.44	78.91	85.36

(b) Ablation on Blocks

Shape	Aspect	R@1 ↑
280×280	1.00	96.17
238×336	1.41	96.50
196×392	2.00	96.55
168×448	2.67	96.29
154×518	3.36	96.22

(c) Ablation on Aspect

Figure 6: **Visual place retrieval with LingBot-Map's Frame Encoder.** All three panels report Recall@1 (R@1 ↑). **(a)** By reusing cached Frame Encoder features, our full SLAM system achieves VPR accuracy comparable to strong CNN-based baselines with only a 1.8M-parameter head and $0.94\,\mathrm{ms}$ of additional descriptor latency for 322×322 inputs. **(b)** Block 18 is the best retrieval tap among the last eight Frame Encoder blocks. **(c)** The square-trained head remains stable across aspect ratios, allowing retrieval to consume cached features directly at the runtime input shape.

of 88.2 across MSLS-val (Warburg et al., 2020), Pitts250k-test (Torii et al., 2013), and SPED (Chen et al., 2017), as shown in Figure 6(a). This result is comparable to strong CNN-based VPR baselines such as EigenPlaces (Berton et al., 2023) (88.5) and MixVPR (Ali-Bey et al., 2023) (89.0). Although it trails standalone DINOv2 models specialized for VPR, including SALAD (Izquierdo & Civera, 2024) and BoQ (Ali-Bey et al., 2024), the descriptor remains effective as the loop-candidate proposal stage of our full system. Pairwise Geometry verifies retrieved pairs before they enter the pose graph, and the complete-system results are reported in Tables 1, 2, and 3.

Among the tested late Frame Encoder blocks, intermediate features outperform the final-block features for retrieval. Block 18 achieves the highest mean R@1 of 88.20, closely followed by Block 19 at 88.13, whereas the final Block 24 reaches 85.36 (Figure 6(b)). This trend is consistent with prior observations that DINO-based VPR can peak before the final transformer layer (Keetha et al., 2023). Finally, we test whether the square-trained head transfers to the native wide input used by the SLAM system. On KITTI (Geiger et al., 2012), R@1 varies by only 0.38 points across five approximately token-matched input shapes and reaches 96.22 at the 154×518 runtime shape (Figure 6(c)). These results support applying the retrieval head directly to cached intermediate DINO tokens at the runtime input shape.

5 Conclusions

We presented Map2SLAM, which builds SLAM on a frozen LingBot-Map by scheduling its inference and keeping its predictions as revisable state. The front-end updates this state through Causal Continuation for stable online tracking, while the asynchronous back-end uses Bidirectional Prefix with newly available context to re-evaluate loop constraints for iPGO and to re-observe segments for Dense Geometric Refinement after the sequence. This design preserves local tracking stability while allowing global trajectory errors to be corrected over time. Experiments on KITTI, VBR, and Oxford Spires show that Map2SLAM reduces mean ATE by more than 40% compared with the second-best uncalibrated method, while maintaining a tracking speed above 28 FPS on an NVIDIA RTX 5090D GPU.

AI USE STATEMENT Generative AI tools were used to assist with manuscript drafting and language editing, as well as software development, including code generation, debugging, testing, and execution of experiments under instructions provided by the authors. The research ideas, methodology, experimental design, and interpretation of results were developed by the authors. All AI-assisted text, code, and experimental outputs were reviewed and verified by the authors. The authors take full responsibility for the final content and results of this work.

REFERENCES

Amar Ali-Bey, Brahim Chaib-Draa, and Philippe Giguere. Gsv-cities: Toward appropriate supervised visual place recognition. *Neurocomputing*, 2022.

Amar Ali-Bey, Brahim Chaib-Draa, and Philippe Giguere. Mixvpr: Feature mixing for visual place recognition. In *WACV*, 2023.

Amar Ali-Bey, Brahim Chaib-Draa, and Philippe Giguere. Boq: A place is worth a bag of learnable queries. In *CVPR*, 2024.

Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016.

Gabriele Berton and Carlo Masone. Megaloc: One retrieval to place them all. In *CVPRW*, 2025.

Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking visual geo-localization for large-scale applications. In *CVPR*, 2022.

Gabriele Berton, Gabriele Trivigno, Barbara Caputo, and Carlo Masone. Eigenplaces: Training viewpoint robust models for visual place recognition. In *ICCV*, 2023.

Leonardo Brizi, Emanuele Giacomini, Luca Di Giammarino, Simone Ferrari, Omar Salem, Lorenzo De Rebotti, and Giorgio Grisetti. VBR: A vision benchmark in Rome. In *ICRA*, 2024.

Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multi-map SLAM. *IEEE Trans. Robot.*, 2021.

Lin-Zhuo Chen, Jian Gao, Shangzhan Zhang, Yihang Chen, Nan Xue, Jianyuan Wang, Christian Rupprecht, Xun Cao, Xing Zhu, Yujun Shen, Yao Yao, and Yinghao Xu. Geometric context transformer for streaming 3D reconstruction. In *ECCV*, 2026.

Zetao Chen, Adam Jacobson, Niko Sünderhauf, Ben Upcroft, Lingqiao Liu, Chunhua Shen, Ian Reid, and Michael Milford. Deep learning features at scale for visual place recognition. In *ICRA*, 2017.

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

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 *ICRA*, 2026.

Tianye Ding, Yiming Xie, Yiqing Liang, Moitreya Chatterjee, Pedro Miraldo, and Huaizu Jiang. LASER: Layer-wise scale alignment for training-free streaming 4D reconstruction. In *CVPR*, 2026.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *NeurIPS*, 2019.

Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *ECCV*, 2014.

Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

Xianze Fang, Jingnan Gao, Zhe Wang, Zhuo Chen, Xingyu Ren, Jiangjing Lyu, Qiao-Mu Ren, Zhonglei Yang, Xiaokang Yang, Yichao Yan, et al. Dens3R: A foundation model for 3D geometry prediction. In *ICLR*, 2026a.

Zhijian Fang, Weicheng Zheng, Yijun Yuan, Weibang Wang, Zhuoguang Chen, Chang Sun, Junhao Huang, Kenan Li, Minghui Qin, and Hang Zhao. SLAMFormer-∞: Infinite SLAM transformer for unbounded frontend and backend processing. *arXiv*, 2026b.

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 *CVPR*, 2026.

Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. LDSO: Direct sparse odometry with loop closure. In *IROS*, 2018.

Yixiao Ge, Haibo Wang, Feng Zhu, Rui Zhao, and Hongsheng Li. Self-supervising fine-grained region similarities for large-scale image localization. In *ECCV*, 2020.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012.

Annika Hagemann, Moritz Knorr, and Christoph Stiller. Deep geometry-aware camera self-calibration from video. In *ICCV*, 2023.

Jiarong Han, Jincheng Xiong, Yuzhou Liu, Linzhe Shi, Changjie Wu, Ning Guo, Mu Xu, Hang Zhang, and Ming Qian. Revisiting local context for long-horizon streaming 3D reconstruction. *arXiv*, 2026.

Sergio Izquierdo and Javier Civera. Optimal transport aggregation for visual place recognition. In *CVPR*, 2024.

Jinwoo Jeon, Dong-Uk Seo, Eungchang Mason Lee, and Hyun Myung. AIM-SLAM: Dense monocular SLAM via adaptive and informative multi-view keyframe prioritization with foundation model. In *ICRA*, 2026.

Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *Int. J. Robot. Res.*, 2012.

Nikhil Keetha, Avneesh Mishra, Jay Karhade, Krishna Murthy Jatavallabhula, Sebastian Scherer, Madhava Krishna, and Sourav Garg. Anyloc: Towards universal visual place recognition. *IEEE Robot. Autom. Lett.*, 2023.

Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *ISMAR*, 2007.

Inha Lee, Dongjae Jeong, Junhee Lee, and Kyungdon Joo. UniSim-SLAM: Feed-forward SLAM with unified Sim(3) optimization. *arXiv*, 2026a.

Jungho Lee, Minhyeok Lee, Sunghun Yang, Minseok Kang, and Sangyoun Lee. SwiftVGGT: A scalable visual geometry grounded transformer for large-scale scenes. In *CVPR Findings*, 2026b.

Lahav Lipson, Zachary Teed, and Jia Deng. Deep patch visual SLAM. In *ECCV*, 2024.

Feng Lu, Xiangyuan Lan, Lijun Zhang, Dongmei Jiang, Yaowei Wang, and Chun Yuan. Cricavpr: Cross-image correlation-aware representation learning for visual place recognition. In *CVPR*, 2024.

Dominic Maggio and Luca Carlone. VGGT-SLAM 2.0: Real-time dense feed-forward scene reconstruction. In *RSS*, 2026.

Dominic Maggio, Hyungtae Lim, and Luca Carlone. VGGT-SLAM: Dense RGB SLAM optimized on the SL(4) manifold. In *NeurIPS*, 2025.

Avilasha Mandal, Rajesh Kumar, Sudarshan Sunil Harithas, and Chetan Arora. VGGT-SLAM++. In *CVPRW*, 2026.

Daniel McGann, John G Rogers, and Michael Kaess. Robust incremental smoothing and mapping (risam). In *ICRA*, 2023.

Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.*, 2015.

Riku Murai, Eric Dexheimer, and Andrew J Davison. MASt3R-SLAM: Real-time dense SLAM with 3D reconstruction priors. In *CVPR*, 2025.

Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

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. *Int. J. Robot. Res.*, 2026.

Zachary Teed and Jia Deng. DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. In *NeurIPS*, 2021.

Zachary Teed, Lahav Lipson, and Jia Deng. Deep patch visual odometry. In *NeurIPS*, 2023.

Akihiko Torii, Josef Sivic, Tomas Pajdla, and Masatoshi Okutomi. Visual place recognition with repetitive structures. In *CVPR*, 2013.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual geometry grounded transformer. In *CVPR*, 2025.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. In *CVPR*, 2026.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, 2024.

Frederik Warburg, Soren Hauberg, Manuel Lopez-Antequera, Pau Gargallo, Yubin Kuang, and Javier Civera. Mapillary street-level sequences: A dataset for lifelong place recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2626–2635, 2020.

Zhuang Xiong, Chen Zhang, Qingshan Xu, and Wenbing Tao. VGGT-Motion: Motion-aware calibration-free monocular SLAM for long-range consistency. In *ICML*, 2026.

Heng Yang, Pasquale Antonante, Vasileios Tzoumas, and Luca Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robot. Autom. Lett.*, 2020.

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 *CVPR*, 2026a.

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. *arXiv*, 2026b.

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 *CVPR*, 2025.

Wei Zhang, Yihang Wu, Songhua Li, and Qi Wang. VGGT-Align: Bridging local reconstruction and global consistency for long-sequence 3D reconstruction. In *ACM MM*, 2026c.

Dong Zhuo, Wenzhao Zheng, Jiahe Guo, Yuqi Wu, Jie Zhou, and Jiwen Lu. Streaming visual geometry transformer. In *ICLR*, 2026.

A Additional Benchmark Results

Table 3 reports the per-sequence results on Oxford Spires. All methods process each sequence at its full length, without temporal cropping. We compute ATE after a single global $\mathrm{Sim}(3)$ alignment using all predicted poses with valid ground-truth correspondences.

Table 3: Absolute trajectory error (ATE, m) on the Oxford Spires Dataset (Tao et al., 2026). KC, OQ, CC, and BL denote Keble College, Observatory Quarter, Christ Church, and Bodleian Library, respectively. "–" denotes tracking failure. LC, Dense Geo., and Frame-wise denote loop closure, dense geometry, and per-frame tracking, respectively. LingBot-Map(S/W) denote the streaming/window modes. † VGGT-SLAM 2.0 uses native keyframes only and is excluded from ranking. The best and second-best results in each section are shown in bold and underlined, respectively.

Method	LC	Dense Geo.	Frame-wise	KC-02 *5969 f 0.29 km*	KC-03 *5682 f 0.28 km*	KC-04 *13518 f 0.78 km*	KC-05 *11544 f 0.71 km*	OQ-01 *5746 f 0.40 km*	OQ-02 *5469 f 0.39 km*	CC-02 *11775 f 0.64 km*	CC-03 *6204 f 0.27 km*	CC-05 *3961 f 0.81 km*	BL-02 *4966 f 0.69 km*	Avg. ↓
Calibrated														
DROID-SLAM (Teed & Deng, 2021)	✓	✓	✓	7.89	31.50	3.93	36.44	**0.43**	**0.29**	4.09	0.86	**8.96**	**1.20**	9.56
DPV-SLAM++ (Lipson et al., 2024)	✓		✓	**2.62**	**0.41**	**1.29**	**0.53**	0.74	0.50	**1.77**	**0.08**	11.93	2.91	**2.28**
Uncalibrated														
DROID-Calib (Hagemann et al., 2023)		✓	✓	11.89	4.82	**1.83**	37.02	**1.01**	3.24	9.92	0.63	**2.63**	**1.36**	7.43
VGGT-Long (Deng et al., 2026)	✓	✓		1.76	14.74	32.53	**4.59**	3.70	5.66	11.65	4.00	13.51	16.16	10.83
SwiftVGGT (Lee et al., 2026b)	✓	✓		7.71	22.69	24.53	21.13	7.01	6.18	25.18	0.89	10.77	29.74	15.58
VGGT-SLAM 2.0† (Maggio & Carlone, 2026)	✓	✓		4.63	9.32	4.75	5.98	1.84	1.65	1.99	0.93	36.25	8.32	7.57
LoGeR (Zhang et al., 2026b)		✓		4.23	3.77	6.87	7.19	5.46	6.05	6.68	2.81	11.75	14.01	6.88
LingBot-Map(S) (Chen et al., 2026)		✓	✓	3.72	3.02	2.73	6.90	3.26	2.29	12.56	2.79	33.76	10.38	8.14
LingBot-Map(W) (Chen et al., 2026)		✓	✓	5.40	8.11	36.67	29.71	7.76	6.95	15.36	4.98	14.45	29.92	15.93
HorizonStream (Cheng et al., 2026)	✓	✓	✓	6.82	8.84	21.19	21.94	5.99	**1.54**	30.19	5.84	26.29	27.97	15.66
ABot-Recon (Han et al., 2026)	✓	✓	✓	2.36	4.25	5.12	4.65	3.71	2.86	7.94	1.92	9.37	11.41	5.36
Ours	✓	✓	✓	**1.53**	**1.89**	2.39	5.73	2.05	1.78	**2.68**	**0.52**	6.32	5.82	**3.07**

B SLAM System Profiling

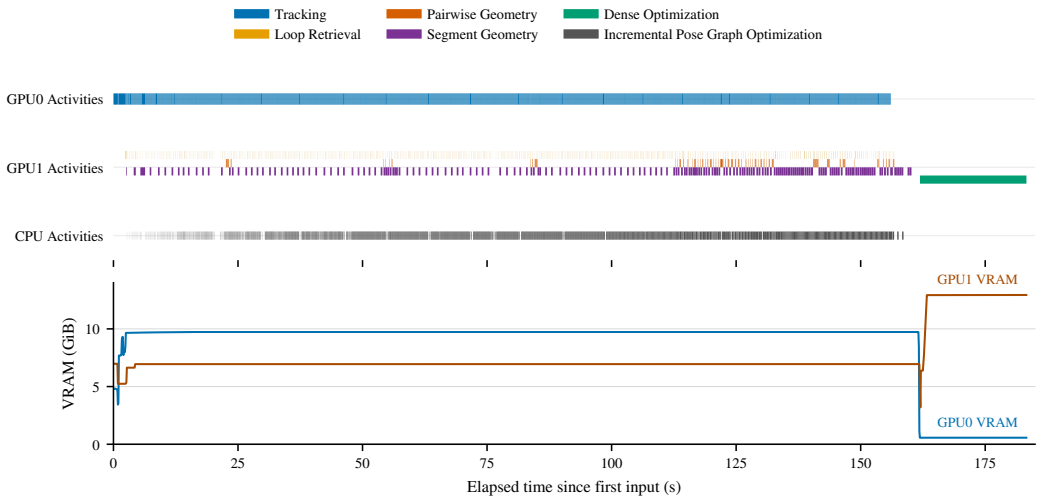


Figure 7: **System execution and memory profile for a complete KITTI 00 run on two NVIDIA RTX 5090D GPUs.** During the online stage, GPU 0 performs Tracking, GPU 1 interleaves Loop Retrieval, Pairwise Geometry, and Segment Geometry, and the CPU executes Incremental PGO. GPU 1 performs terminal Dense Optimization after the online stage. The lower plot reports per-GPU VRAM usage throughout the run.

We profile the complete KITTI Sequence 00 on two NVIDIA RTX 5090D GPUs. GPU0 and GPU1 are dedicated to the front end and back end, respectively. GPU0 executes Tracking, while GPU1 executes Loop Retrieval, Pairwise Geometry, Segment Geometry, and terminal Dense Optimization. Both Pairwise Geometry and Segment Geometry use fresh-context bidirectional prefix inference

over selected historical frames to obtain repeated geometric observations. On the CPU, we record only Incremental Pose Graph Optimization (iPGO). The activity bars show the measured execution spans of these components, while the memory curves report GPU memory usage sampled every $100\ \mathrm{ms}$.

The profile shows the following:

1. **Smooth front-end tracking.** GPU0 remains almost continuously occupied by Tracking and processes KITTI Sequence 00 at $29.1\ \mathrm{FPS}$, indicating steady front-end execution.
2. **Asynchronous front-end/back-end execution.** Tracking on GPU0 overlaps with loop processing and Segment Geometry on GPU1, as well as iPGO on the CPU. The repeated iPGO activity shows that the trajectory is updated online while new frames are being tracked.
3. **Effective loop-candidate filtering.** From $4{,}541$ input frames, the system selects $2{,}208$ keyframes and computes a SALAD descriptor for each, but issues only 429 retrieval queries. These queries produce $2{,}141$ candidate pairs, of which only 175 (8.2%) enter Pairwise Geometry. Among them, 59 pass verification, corresponding to 33.7% of the geometrically verified candidates and 2.8% of all retrieved pairs. The retrieval and candidate-pool filters therefore restrict expensive Pairwise Geometry to a small subset of candidate pairs.
4. **Opportunistic Segment Geometry.** Loop Retrieval and Pairwise Geometry occupy only short intervals on GPU1, leaving substantial idle capacity for Segment Geometry. By running Segment Geometry concurrently with tracking, most segment observations are completed online, and the final Segment Geometry execution finishes only approximately $4.15\ \mathrm{s}$ after tracking ends.
5. **Dense-optimization memory bottleneck.** During online operation, GPU0 and GPU1 peak at $9.73\ \mathrm{GiB}$ and $6.98\ \mathrm{GiB}$, respectively. Terminal Dense Optimization raises the GPU1 peak to $12.94\ \mathrm{GiB}$. Reducing the depth-map resolution may therefore lower the peak memory requirement.

C Detailed VPR Evaluation

Table 4 extends the main-paper analysis with the complete R@1/R@5/R@10 comparison on MSLS-val (Warburg et al., 2020), Pitts250k-test (Torii et al., 2013), and SPED (Chen et al., 2017). We measure every method within the same evaluation framework, using the same benchmark splits, ground-truth annotations, and FAISS L2 retrieval, while retaining each method's standard evaluation resolution. Latency is measured with batch size 1 and FP16 inference on a single NVIDIA RTX 5090D GPU using CUDA events after at least 20 warm-up iterations. For standalone baselines, the reported latency covers a full descriptor-model forward at each method's standard evaluation resolution. For our method, it covers only the 1.8M-parameter aggregation-head forward on precomputed tokens from 322×322 images. GeM and NetVLAD are our GSV-Cities reproductions because no standard checkpoints are publicly available; the remaining rows use the corresponding released weights.

We train only the 1.8M-parameter aggregation head while keeping the Frame Encoder frozen. Training runs for 10 epochs on GSV-Cities (Ali-Bey et al., 2022) at 224×224 resolution using 16-bit mixed precision on one NVIDIA RTX 5090D GPU and takes approximately two hours. We use AdamW with an initial learning rate of 10^{-3} and a batch of 128 places with four images per place. The learning rate decays linearly to 2×10^{-4} over training. We validate after each epoch on Pitts30k-val and select the checkpoint with the highest R@1.

On MSLS-val, Pitts250k-test, and SPED, our descriptor reaches R@1 scores of 87.8, 93.2, and 83.5, respectively. It outperforms SFRS (Ge et al., 2020), GeM (Radenović et al., 2018), Conv-AP (Ali-Bey et al., 2022), and CosPlace (Berton et al., 2022) on all three datasets, while remaining within 1.2 points of MixVPR on each dataset.

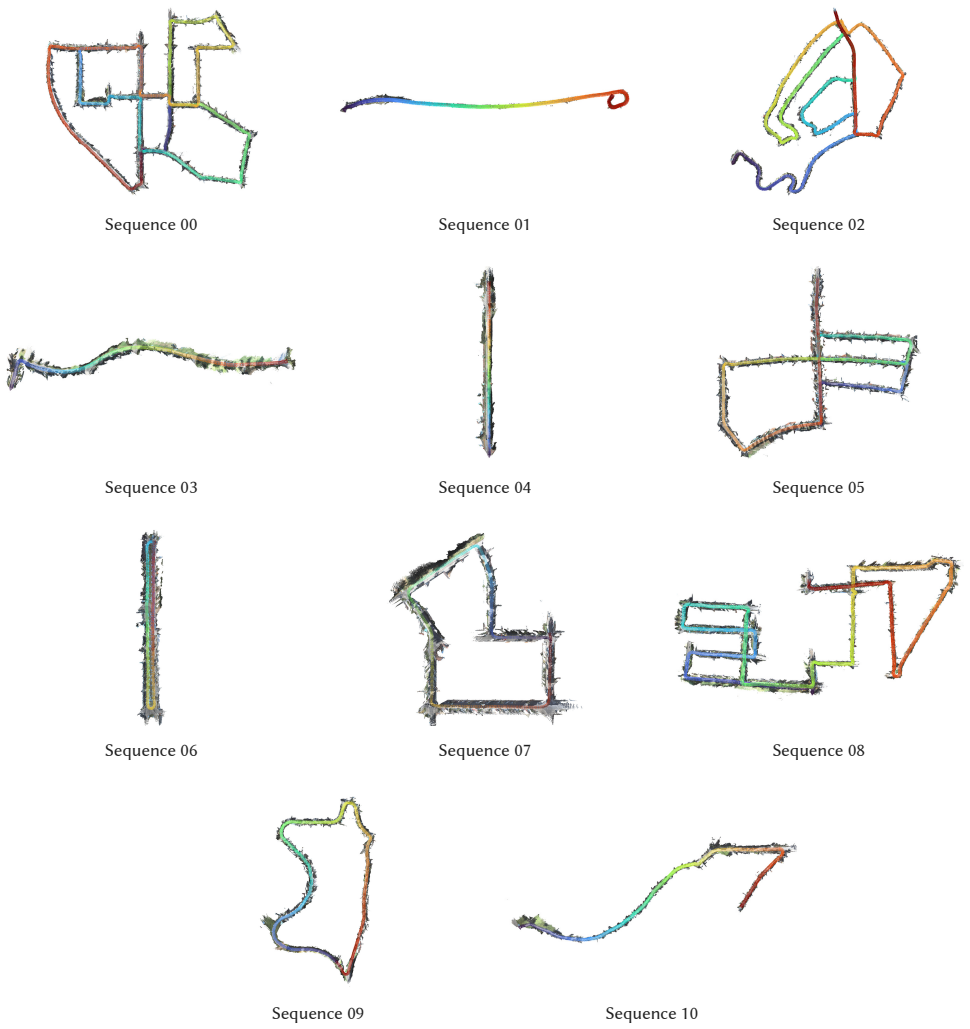


Figure 8: **Qualitative point-cloud results on KITTI.**Top-down views of the point clouds produced by our method for Sequences 00–10.

Table 4: **Detailed visual place retrieval results.** Each dataset reports R@1/R@5/R@10 (%); higher is better. ΔParams and ΔLatency denote the additional model parameters and inference latency incurred by adding each VPR method to our SLAM system. Standalone baselines add their full descriptor model, whereas Ours adds only the aggregation head on cached Frame Encoder tokens.

Method	Dim.	ΔParams (M)	ΔLatency (ms)	MSLS-val	Pitts250k-test	SPED
SFRS (Ge et al., 2020)	4096	149.0	2.78	64.2/74.6/77.8	89.5/95.3/96.5	73.0/85.7/87.8
GeM (Radenović et al., 2018)	2048	23.5	3.65	78.9/87.6/90.1	86.5/94.6/96.4	66.9/79.4/85.2
NetVLAD (Arandjelovic et al., 2016)	32768	24.6	4.03	83.1/90.7/92.7	92.6/97.5/98.3	79.2/89.3/91.9
Conv-AP (Ali-Bey et al., 2022)	4096	25.6	2.92	83.0/90.1/92.6	92.6/97.7/98.6	81.1/89.6/93.3
CosPlace (Berton et al., 2022)	2048	27.7	3.08	87.6/92.7/94.7	92.4/97.5/98.4	79.6/89.0/92.6
MixVPR (Ali-Bey et al., 2023)	4096	10.9	2.48	88.1/93.2/94.5	94.3/98.2/98.9	84.7/92.3/94.4
EigenPlaces (Berton et al., 2023)	2048	27.7	2.90	89.1/93.1/95.3	94.5/98.1/98.8	81.9/92.3/95.1
SALAD (Izquierdo & Civera, 2024)	8448	88.0	5.57	92.0/96.2/96.9	95.0/98.5/99.2	91.9/96.1/96.7
CricaVPR (Lu et al., 2024)	10752	106.8	13.55	89.5/95.4/96.2	95.6/98.5/99.2	89.6/94.6/95.4
BoQ (DINOv2) (Ali-Bey et al., 2024)	12288	95.2	5.34	93.8/96.8/97.0	96.6/99.1/99.5	92.6/95.9/96.9
MegaLoc (Berton & Masone, 2025)	8448	228.6	5.93	93.4/97.0/97.8	96.4/98.9/99.4	90.4/95.2/95.9
Ours	8448	1.8	0.94	87.8/93.8/94.6	93.2/97.6/98.6	83.5/92.1/94.6

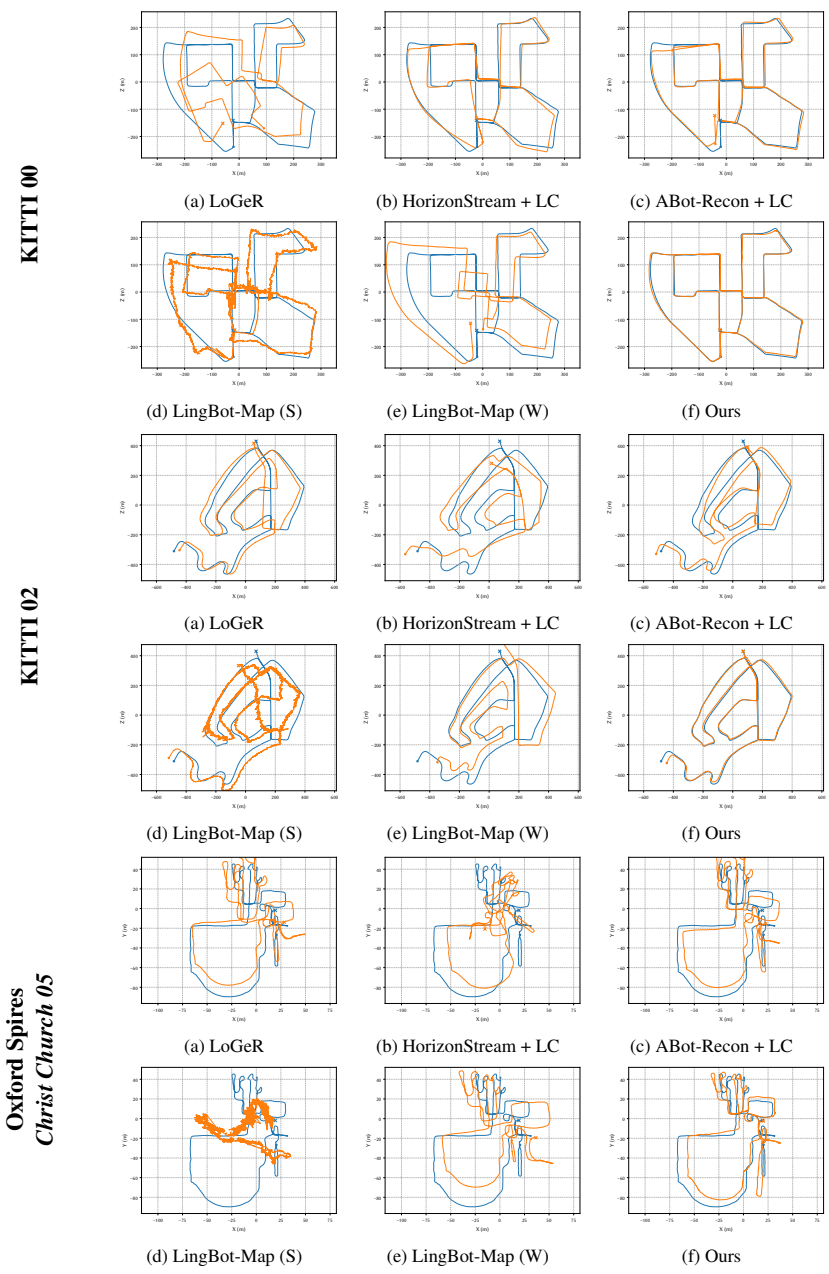


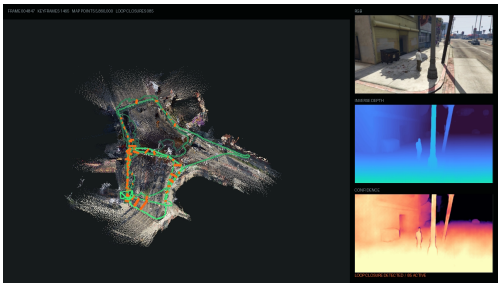
Figure 9: **Additional full-sequence trajectory comparisons.** Blue and orange denote the reference and estimated trajectories, respectively. Each estimate is aligned once over the full sequence by $\mathrm{Sim}(3)$, matching the ATE protocol.

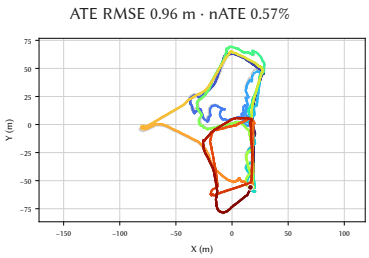
D INTERACTIVE QUALITATIVE RESULTS

Figure 10 shows qualitative SLAM results on 3 game sequences with different challenging conditions. The first GTAV sequence contains a transition **from nighttime to daytime**, while the second GTAV sequence features a **sudden change in speed** of camera motion. The third sequence, from Assassin's Creed Origins, contains **dense loop closures**.

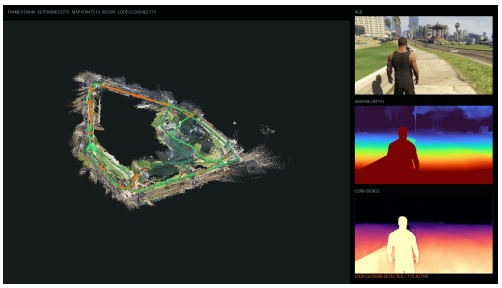
The complete videos are available on the offline supplementary website and provide a clearer view of the system's behavior over time than the static frames shown here. We therefore encourage readers to explore the supplementary website for a more comprehensive visualization of these sequences and our SLAM results.

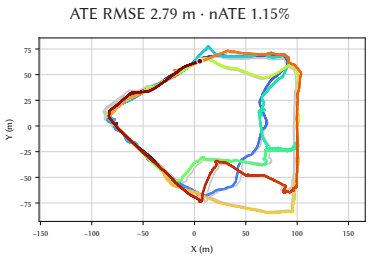
GTA V: Night-to-day transition



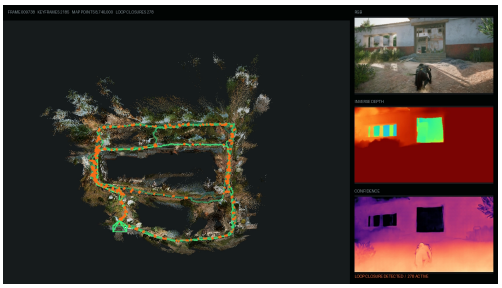


GTA V: Abrupt camera-speed changes





Assassin's Creed Origins: Dense loop closures



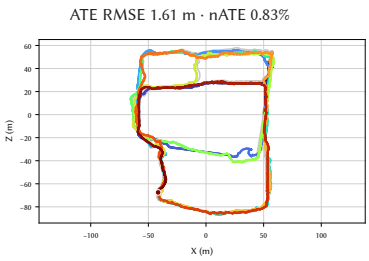


Figure 10: **Results on three game sequences.** Each row pairs the terminal system visualization (left) with the final trajectory against ground truth (right). The trajectory panels report ATE RMSE and normalized ATE (nATE).The trajectories estimated by our system are color-coded from blue to red over time, while the ground-truth trajectories are shown in gray. As illustrated, our estimated trajectories closely align with the ground truth across nearly all sequences.