MERLoc: Robust Robot Localization in Dynamic Environments with Pose-Based Retrieval over 3D Gaussian Splatting References

Michael Baumgartner[1,2], Dario Mylonopoulos[2], Sammy Christen[2], Markus Gross[1,2], Moritz Bächer[2]



Fig. 1: **MERLoc in a dynamic hallway.** Overhead view: scanning the empty hallway (left), pre-processing the 3D Gaussian Splatting scene into a Functional Reference Memory (middle), and localizing in the runtime scene with moving people and displaced objects (right). Yellow frustums mark FRM reference poses. The black trace shows our onboard pose estimate, solid when localization succeeds and dotted when relying on odometry. For one timestep, we show the current pose estimate (vermillion frustum), the camera view, and the retrieved references in reddish purple, orange, blue, and bluish green.

***Abstract*— Robots operating in previously mapped spaces must localize accurately despite moving people and rearranged objects. Most localization pipelines match the query against a sparse set of captured reference images retrieved by appearance, which is brittle under scene change and repeating structures. We present MERLoc, which turns a single metric 3D Gaussian Splatting scan into a dense *Functional Reference Memory*. Reference views are sampled over the navigable space for the robot's morphology and sensors at a target density and rendered offline. *Pre-encoding* them for the chosen off-the-shelf localizer removes the reference encoder from inference. A *pose-based* retrieval strategy then selects references by spatial proximity and viewing direction rather than appearance, sidestepping visual ambiguities whenever a prior exists. Across one classical and four learned localizers in simulated and real dynamic scenes, pose-based retrieval matches or outperforms appearance retrieval in most cases, even when the latter is gated by the same prior. On real data, it lowers position RMSE by 15–60 % in nine of ten backend/split combinations, and the rendered memory outperforms real-image caches posed by Structure-from-Motion. The system runs onboard a Jetson Orin NX at 5 Hz, enabling a navigation policy to successfully traverse a hallway with people roaming and rearranging objects.**

I. Introduction

Robots deployed in homes, hospitals, and warehouses operate in spaces that are easy to scan but rarely remain identical to those scans over time. People walk through them, objects appear, and furniture moves. While structural elements provide reliable localization cues, these dynamic changes introduce distractors and ambiguities that degrade localization pipelines built for static scenes.

The dominant localization pipeline for previously mapped environments queries a sparse set of posed reference images [1]. References are posed offline by Structure-from-Motion (SfM) [2], retrieved online via appearance descriptors [3], and passed to a relative localizer. Frozen at mapping time, the reference set is limited to the captured viewpoints and can neither be densified nor adapted to a robot's sensor suite. Furthermore, appearance-based retrieval is prone to *visual aliasing*, where distinct places look alike, such as repeated corridors or identical doors. Aliasing and scene changes since mapping both lead retrieval to return confidently wrong references that no relative localizer can recover from [4, 5, 6]. Recent feed-forward localizers replace the geometric solver but inherit both limitations; even when trained on dynamic scenes [7, 8, 9], most are, as we show, more sensitive than classical pipelines to aliased or changed references.

We propose a 3D Gaussian Splatting-based **M**emory **E**ncoding and **R**etrieval **Loc**alization (MERLoc) framework, which replaces the rigid, captured reference set with a controllable, backend-adaptive, rendering-based memory. Given a 3D Gaussian Splatting (3DGS) [10, 11] scan of the environment, we sample reference poses over the navigable space at a specified density, height range, and orientation profile tailored to the robot's sensor suite. We then render these views and encode them offline in a representation customized for

[1]M. Baumgartner and M. Gross are with ETH Zurich, Switzerland.
[2]M. Baumgartner, D. Mylonopoulos, S. Christen, M. Gross, and M. Bächer are with Disney Research, Switzerland.

the relative localizer. This forms a *Functional Reference Memory* (FRM): its content is determined by the target backend and robot morphology rather than by the scanner that captures it, e.g., depth-lifted keypoints for classical pipelines [1] or encoded visual tokens for latent-conditioned architectures [7, 8, 9, 12]. Because of this customized encoding, the relative localizer runs entirely off-the-shelf, without per-scene training or adaptation, and without encoding any reference at runtime, which removes one encoder pass per retrieved reference from the online path. Furthermore, the FRM density becomes a tunable design parameter that trades accuracy against storage without increasing online inference cost.

To select references, we index the dense FRM by pose. Given a pose prior, typically the previous estimate or odometry, we gate candidates within a radius and rank them by viewing-direction alignment and proximity, falling back to appearance-based matching only for initialization or recovery. Unlike SLAM, which routinely gates retrieval with a propagated pose [13, 14], localization pipelines still retrieve by appearance alone. Neither the dense FRM nor pose-based retrieval suffices alone: pose-based retrieval starves on a sparse reference set, and appearance retrieval still aliases on a dense one. Together, they make classical and learned relative localizers metric, more accurate, and more robust to scene change without touching backend weights.

We claim that (i) MERLoc turns one metric 3DGS scan into a *Functional Reference Memory* whose density, height band, and encoding are chosen per robot and per localizer, so that classical and feed-forward backends run off-the-shelf without encoding references at runtime (Sec. III); (ii) on real data, this rendered memory outperforms real-image caches of the same scene posed by SfM, even at a lower pose density for the classical backend (Sec. IV-D); (iii) *pose-based* context retrieval over this memory lowers position RMSE by 15–60 % relative to appearance retrieval for nine of ten real-world backend/split combinations and, under simulated scene change and human activity, matches or exceeds prior-gated appearance retrieval while clearly exceeding plain appearance retrieval for four of five backends (Secs. IV-B–IV-D); (iv) the system runs at $5\,\mathrm{Hz}$ on a Jetson Orin NX, enabling a navigation policy to reliably traverse a hallway while people move around the robot and displace objects (Sec. IV-E).

II. Related Work

A. Visual Localization and Aliasing

The main approaches driving visual localization research are scene coordinate regression (SCR), absolute pose regression (APR), and retrieval-based localization with classical or learned relative pose estimation.

1) Scene Coordinate and Absolute Pose Regression: SCR trains a network to predict per-pixel 3D scene coordinates [15, 16]; APR regresses the pose directly [17]. Both bake the scene into network weights and degrade under structural change [18]; APR moreover behaves like pose interpolation [19]. MERLoc trains nothing per scene and runs off-the-shelf, scene-agnostic backends over an explicit reference memory.

2) Retrieval-Based Localization: Classical structure-based pipelines and learned relative pose regression (RPR) estimate the query pose against a posed reference database, typically sparse and reconstructed offline via SfM [2]. They lead long-term localization benchmarks [6] and are our focus.

Classical pipelines [1] chain local feature extraction [20], matching [21], and robust Perspective-n-Point (PnP) [22, 23]. Recent feed-forward networks replace this pipeline with a single forward pass, predicting relative poses or pointmaps for two-view [7, 12] or multi-view [8, 9] sets. While these models reduce system complexity, most tolerate aliased reference sets worse than classical pipelines, even when trained on dynamic scenes [7, 8, 9].

3) Appearance-Based Retrieval: Classical and learned relative methods retrieve references with appearance-based Visual Place Recognition (VPR) [3, 24], which is vulnerable to visual aliasing. In repetitive or changed scenes, retrieved references can be confidently wrong, and no relative localizer recovers [4, 6]. Existing remedies stay in the appearance domain. Hierarchical pipelines cluster the retrieved frames by co-visibility before matching [1], and sequence-based VPR [25] enforces temporal consistency of descriptor matches. Gating the appearance search by a pose prior, which we evaluate as VPR+, still ranks by descriptor similarity and inherits its failure modes. We instead remove appearance from the retrieval decision and rank references by pose.

B. 3DGS Localization and SLAM

1) Localizing with Rendered References: Rendering a scene model for localization predates 3DGS: Torii et al. [26] synthesize database views so that query and reference share a viewpoint, MeshLoc extracts features on mesh renderings [27], LENS trains a pose regressor on NeRF views placed by the scene geometry [28], and Ress et al. [29] add 3DGS renders at random poses to an SfM reference set. Loc-NeRF [30], GS-CPR [31], and GSVisLoc [32] instead work at query time, respectively: per particle of a Monte Carlo filter, to refine a coarse pose from another localizer, or to refine a direct match of encoded 3D Gaussians at seconds per query. Active localization methods such as ActLoc [33] use the scene model to choose the viewing directions expected to localize best, which does not help a passive camera that cannot be steered. None samples its reference set by robot embodiment and density, pre-encodes it for an off-the-shelf localizer, or changes how references are retrieved. Unlike query-time methods, all of our rendering happens offline.

2) Pose Priors in SLAM and Localization: Using a propagated pose to decide what to match against is standard in robotics. Monte Carlo localization weights pose hypotheses propagated by odometry against a map [13], and ORB-SLAM2's localization mode tracks a saved map by matching to the map points predicted visible from the propagated pose [14]. Visual localization pipelines, in contrast, treat every query independently and retrieve by appearance. MERLoc brings the prior into the localization pipeline: it gates and

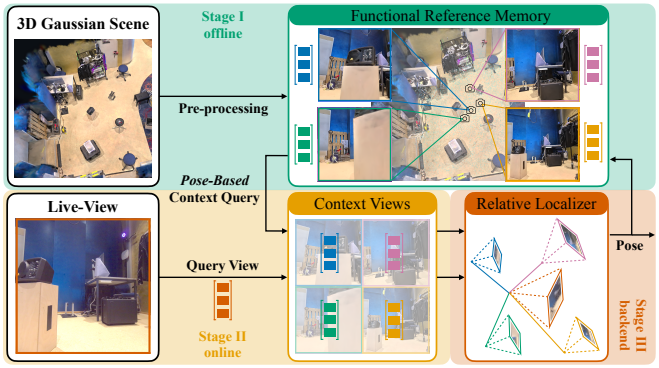


Fig. 2: **MERLoc** consists of three stages: an offline scene pre-processing stage that turns the 3D Gaussian Splatting scan into the Functional Reference Memory (Sec. III-A), an online pose-based context retrieval stage (Sec. III-B), and an interchangeable backend characterized by a relative localizer (Sec. III-C). An appearance-based relocalizer handles initialization and failure recovery.

ranks references in a dense, metric memory by a propagated pose and re-estimates the pose absolutely with an off-the-shelf localizer, avoiding error accumulation as in odometry.

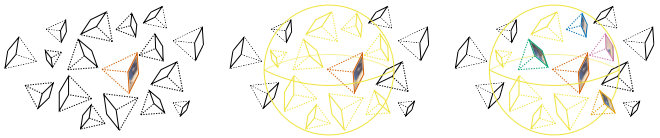
III. System Architecture

Our proposed system, MERLoc (Fig. 2), consists of three stages. The first stage is an offline pre-processing step, where we ingest the 3DGS scan of a scene (Sec. III-A). We sample camera poses *in a controlled manner* within a height band and density informed by the robot's morphology, sensor suite, storage budget, and desired localization accuracy, then render these views and pre-encode them into a format matched to the relative localizer. In the second stage, at runtime, we query context encodings from this FRM using a pose prior (Sec. III-B) from the previous localization result or, if available, odometry. In the third stage, we feed these retrieved references, along with the encoded live query view, into one of several backends (Sec. III-C). Each backend is characterized by a relative localizer, which is used to obtain a global pose. Our system design deliberately adapts the FRM to the backend, rather than the reverse, enabling relative localizers to run off-the-shelf without per-scene training or fine-tuning.

A. Stage I: 3DGS Scene Pre-Processing

Our method utilizes a high-fidelity 3DGS scan and at least a rough collision mesh, both readily obtainable with current 3DGS pipelines and commercial scanners. Our real-world scan comes from a handheld LiDAR-camera scanner that outputs both (Sec. IV-A). In simulation, the simulator's scene model replaces the scan.

1) Guided Pose Sampling: We first extract the navigable space of a scan: from the scene's collision mesh, we rasterize a 2D occupancy grid with cell size W. We retain cells with floor support and sufficient headroom H, remove cells within a clearance margin M of obstacles via erosion, and discard disconnected islands smaller than I, which the robot cannot reach. FRM viewpoints are then sampled to match the robot's specific morphology and sensor suite: positions are sampled uniformly over the navigable area within a sensor-height band $B = [b_{\min}, b_{\max}]$ of thickness $D = b_{\max} - b_{\min}$, with orientation ranges tailored to the platform. The number of FRM poses is strictly density-controlled: $N = \rho \cdot A \cdot D$ for a target density ρ (poses per m^3) over navigable area A. Density is the primary knob that trades accuracy against storage (Sec. IV-F).



(a) Query image pose (b) Candidate poses (c) Retrieved references

Fig. 3: For each query image (a), our context retrieval gathers FRM poses within radius R of the current pose guess (b), keeps the top P by orientation cosine similarity, and selects the K nearest among them (c). Best viewed in color.

2) Rendering and Backend-Specific Encoding: Every sampled FRM pose is rendered with color and depth using the target camera's intrinsics, from the 3DGS scan with a real-time Gaussian rasterizer [34] for real scenes and directly from the simulator for simulated ones, so simulated references and queries share one renderer. The FRM content is then pre-encoded to exactly match the input requirements of the relative localizer. For the classical backend, we extract sparse feature keypoints and descriptors per reference, lift them to metric 3D via the rendered depth, and persist them alongside the camera pose. For learned backends, we store pre-encoded, posed image latents. Where necessary, we also store depth. In all cases, we additionally store an image descriptor for appearance-based retrieval. Thus, we offload the heavy encoder runs for multiple reference views to the pre-processing stage, saving inference time. A position k-d tree and an approximate nearest-neighbor index over the image descriptors complete the FRM as its spatial and appearance indices.

FRM size scales linearly with density: $S_{\mathrm{FRM}} = \rho \cdot A \cdot D \cdot (S_{\mathrm{R}} + S_{\mathrm{ID}})$, where S_{R} and S_{ID} are the per-reference encoding and image-descriptor sizes.

B. Stage II: Pose-Based Context Retrieval

Because robots move continuously, the previous pose estimate can provide a strong prior for the current localization query. However, appearance-based retrieval cannot inherently benefit from such a prior beyond simple gating. In contrast, our retrieval strategy (Fig. 3) queries the FRM's k-d tree for all reference poses within a radius R of the prior position. It then ranks this candidate pool by viewing-direction alignment (cosine similarity between camera forward axes), retains the P best-aligned candidates, and selects the K nearest poses from this subset. Ranking orientation before position ensures the system prioritizes view alignment over spatial proximity.

A query fails only when the backend returns no pose or a non-finite pose, raises an exception, or, for PnP-based backends, fails the minimum-inlier test of Tab. II. We deliberately use no jump or consistency gate, meaning backends that

always emit a pose never fall back. To initialize, after five consecutive failures, or when fewer than K candidates lie within R, the system temporarily falls back to appearance-based retrieval before resuming autoregressive localization. Notably, our system does not *require* odometry and still beats appearance-based retrieval without it, as we show in Sec. IV.

When available, any forward-propagated guess, e.g., from odometry, can replace the previous localization estimate to further improve accuracy and robustness. We upper-bound this gain with an oracle prior in Sec. IV and make use of odometry in our real-world robot navigation experiment in a highly dynamic scene (Sec. IV-E).

C. Stage III: Relative Localization Backends

The relative localizer receives the encoded live query image alongside the K references from the FRM and predicts the relative pose. Every method that natively emits scale-normalized output becomes metric solely through the FRM, using known reference poses for alignment and pre-rendered depth for 3D lifting. We evaluate five different backends:

Classical multi-stage localizer: Based on the hloc pipeline [1], we extract ALIKED [20] features on the live query and match them via LightGlue [21] directly against the K references' pre-encoded keypoints. After a per-pair essential-matrix RANSAC filter, we execute a joint PnP [22] plus RANSAC [23] solve over the pooled 2D–3D correspondences, followed by Levenberg–Marquardt (LM) refinement.

MonST3R [7], Reloc3r [12] (two-view feed-forward): For MonST3R, we lift reciprocal pointmap matches between the query and each of the K references to metric 3D using the references' rendered depth, then run PnP with RANSAC on the pooled 2D–3D correspondences using the known query intrinsics. For Reloc3r, we use its native pipeline: pairwise rotation and translation-direction regression against each of the K references, followed by motion averaging.

VGGT-Ω [8], π^3 [9] (multi-view feed-forward): We conduct one forward pass over the set of [K references, query]. We align the predicted reference camera centers to their known metric FRM poses (Umeyama alignment with scale [35]), which maps the predicted query pose into the scene frame. For π^3, we additionally consume the references' poses, intrinsics, and rendered depth through its native conditioning branch.

All learned backends run with their publicly available weights for the configurations listed in Tab. II with zero fine-tuning on our scenes.

IV. Experiments

A. Experimental Setup

1) Evaluation Scenarios: We compare MERLoc with existing baselines across four scenarios: heavy scene changes, increasing human interaction, real-world localization, and in-the-loop with a navigation policy.

Controllable Scene Changes in Arena Rosnav: To freely control scene composition, we build two scenes in Arena Rosnav [36]: a $384\,\mathrm{m}^2$ hospital wing with 53 movable objects and a $273\,\mathrm{m}^2$ warehouse with 75 objects. In each, a navigation policy follows hand-selected waypoints, shared across both robot platforms (Tab. I), to produce a physically plausible $97\,\mathrm{m}/98.5\,\mathrm{m}$ robot trajectory (hospital/warehouse), and we sample six human trajectories crossing the robot's field of view (FOV). Keeping all trajectories fixed, we resample each scene ten times by adding, removing, or repositioning movable objects so they never intersect any path and stay within the robot's FOV along its trajectory. On average, each resampling moves 24.8 objects in each scene, removes 5.0/7.1 objects (plus 117.8 of 1436 shelf boxes in the warehouse), and adds 2.7/2.9 duplicates. Fig. 4a shows a view from the Boxer sensor suite in the hospital scene under changes.

Controllable Human Interactions in HM3D: We populate a $37\,\mathrm{m}^2$ region of the real-world HM3D scan 00807 [37] with simulated humans moving as a group, sampling ten trajectory sets for each group size $G \in \{4, 8, 16, 24\}$. A smooth $24\,\mathrm{m}$ camera trajectory follows the group, fitted through multiple waypoints with a curvature-adaptive speed of up to $0.4\,\mathrm{m/s}$ and kept within $\pm 3\,\mathrm{cm}$ of the sensor height (Tab. I). This controlled following setup measures performance as a function of dynamic view obstruction rather than at a single operating point. Fig. 4b shows a view at the same timestep with different numbers of humans.

Real-World Localization in a Motion Capture Scene: To assess real-world performance and the rendered-to-real gap, we record a dataset combining scene changes and human interaction with our custom bipedal robot on a $14.2\,\mathrm{m}^2$ MoCap stage densely filled with movable objects. It comprises a two-minute *static* sequence and a three-minute *dynamic* sequence, in which two humans walk through the scene, freely displace objects, and interact with the robot. Both are recorded at $30\,\mathrm{Hz}$ and queried offline at $10\,\mathrm{Hz}$, with ground truth from MoCap. To further exacerbate scene change, we scan the scene only after all recordings, once all objects have been moved, and localize queries from both sequences against the resulting *mismatched* FRM. We scan the scene with a handheld XGRIDS PortalCam and use the same images to form the SfM-posed caches of Sec. IV-D.

In-The-Loop Navigation in a Real Hallway: We run two real-world navigation experiments with MERLoc in the loop, using a custom navigation policy in a hallway with repetitive structures where human actors move around the robot and rearrange objects during operation (Fig. 1). We train the policy with hierarchical reinforcement learning following [38] on top of a standard locomotion policy, and it receives egocentric 2D raycasts and the global pose from our framework.

2) Retrieval Conditions: We evaluate MERLoc by running each relative localizer (Sec. III-C) under four retrieval conditions over the same FRM. *Ours* is our pose-based retrieval in autoregressive mode, where the previous localization output seeds the next query. *Oracle* seeds our retrieval with the ground-truth query pose and upper-bounds what accurate odometry could contribute. *VPR* is the appearance-based retrieval of standard pipelines. *VPR+* gates VPR to the radius R around its own autoregressive prior, as *Ours* does, but still ranks candidates by descriptor similarity.

In our experiments, the CNN-based NetVLAD [3] proved

TABLE I: Evaluated robot platforms and their sensor configurations.

Platform	Camera	Height	Pitch	HFOV	Image size
Simulation (Arena Rosnav)					
TurtleBot	OAK-D Pro	0.179 m	0°	69°	512×512
Boxer	RealSense D435	0.084 m	14.76°	69.4°	512×512
Simulation (HM3D)					
-	Pinhole	0.45 m	0°	90°	512×512
Real-world (MoCap and Navigation setups, respectively)					
Bipedal	IMX900	0.40 m	0°	90°	512×384
Bipedal	IMX335	0.42 m	0°	115°	640×480

TABLE II: MERLoc parameters. Values are shared across all experiments unless ablated in Sec. IV-F.

	Parameter	Value
Functional Reference Memory	Occupancy grid cell W	0.05 m
	Obstacle clearance M	0.05 m
	Required headroom H	1.8 m
	Min. island area I	0.5 m^2
	Height Band B (Arena, TurtleBot)	[5 cm, 32.9 cm]
	Height Band B (Arena, Boxer)	[5 cm, 23.4 cm]
	Height Band B (HM3D & real-world)	[20 cm, 1.2 m]
	Pitch / roll / yaw† (Arena)	±4° / ±0° / ±180°
	Pitch / roll / yaw† (HM3D & real-world)	±30° / ±5° / ±180°
	Density ρ	1000 m^{-3}
	Render size	512^2
	Keypoints per reference	≤ 256 (thr. 0.2)
Retrieval	Query radius R	1.5 m
	Orientation pool P	64
	Context size K	classical: 4 / learned: 8
	Relocalization trigger	5 consecutive failures
Solver	PnP solver	EPnP + LM
	Reprojection threshold	4 px
	Minimum inliers per reference	15
Backends	MonST3R	512-DPT, conf. 3.0
	Reloc3r	512, mixed precision
	VGGT-Ω	1B/512
	π^3	504^2

†Pitch, roll, and yaw sample ranges are relative to the sensor mount.

more robust to visual aliasing and less volatile overall than the more recent, transformer-based SALAD [24]. Hence, we use NetVLAD for our VPR baseline and as the reinitialization method for our proposed system.

All four conditions share the identical FRM and, for the classical backend, the same depth-lifted features. *Ours* vs. *VPR*/*VPR+* isolates the retrieval rule and *Ours* vs. *Oracle* isolates the cost of prior drift. *VPR+* specifically separates the benefit of a prior from that of orientation-aware pose ranking. For the real-world results, we also run backends on caches of real images posed by SfM and compare their performance to that on the FRM.

3) Platforms and Parameters: Tab. I lists the five evaluated sensor configurations, three simulated and two on the real robot. Tab. II lists the method parameters, which are shared across all experiments unless ablated in Sec. IV-F. We define FRM height bands per platform to encompass the mounted camera, so one scan can serve different robot morphologies with different mounts.

4) Metrics: We report recall@5 cm/5° and position root mean squared error (RMSE). Recall is computed over *all* ground-truth frames. This way, an unlocalized frame counts as a miss and coverage is folded into the metric. RMSE is calculated over *valid* frames and exposes outlier proneness [6]. Recall can improve at the expense of RMSE and vice versa; when a method solves more frames at the cost of more outliers, this does not directly indicate greater robustness.



(a) Arena Rosnav hospital (b) HM3D apartment

Fig. 4: **Simulated Benchmarks:** Views in the Arena Rosnav hospital and HM3D apartment scenes. (a) The same view under scene changes. (b) The same view with $G = 4$ (left) and $G = 24$ (right).

B. Simulated Scene Changes

This experiment tests whether pose-based retrieval survives object rearrangement that defeats appearance retrieval. Fig. 5 shows the localization performance of all evaluated backends with varying retrieval methods on the Arena Rosnav scenes.

For four of five backends, our retrieval outperforms VPR and VPR+ by raising the median recall and/or the recall floor (MonST3R is on par). The prior accounts for most of the gain, and orientation-aware ranking adds further improvement. Furthermore, a better pose prior, as assumed in oracle retrieval, mainly raises the recall floor, leading to slightly higher and tighter overall performance.

Notably, most learned backends degrade with plain VPR retrieval. On the Arena scenes, VPR-retrieved references lie a median 0.78 m from the ground-truth query pose and 1.01 m apart from each other, versus 0.28 m to 0.37 m and 0.36 m to 0.49 m for our pose-based retrieval, indicating visual aliasing. We hypothesize that Reloc3r, VGGT-Ω, and π^3 are sensitive to a mix of correct *and* incorrect references. The classical backend and MonST3R show less performance loss with VPR, likely because their confidence estimates let them discount bad references. VPR+ removes most of this

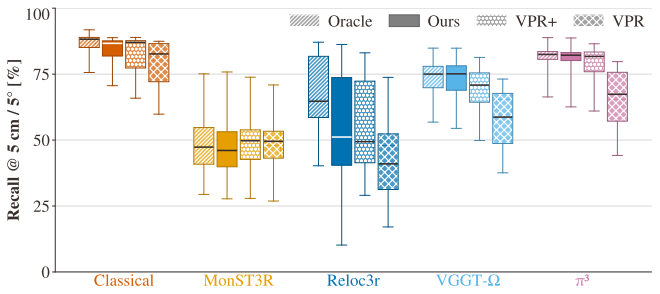


Fig. 5: **Simulated Dynamic Scenes:** Localization performance on two Arena Rosnav scenes with two sensor setups and ten object rearrangements each. Our context retrieval improves most localization backends except for MonST3R, while oracle pose priors mainly raise the recall floor and reduce variance further.

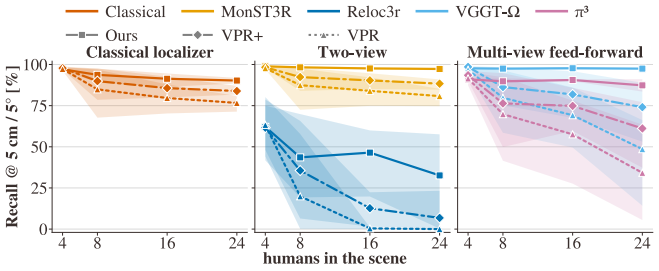


Fig. 6: **Simulated Human Interactions:** Mean localization performance with P10/P90 bands versus the number of simulated humans in the HM3D human-following scenario. For all localizers, VPR & VPR+ retrieval degrades markedly as we add humans and varies more across seeds, while ours is largely unaffected, except for Reloc3r.

loss, but most backends benefit from the consistent reference sets our retrieval provides.

C. Simulated Human Activity

This experiment tests how retrieval is affected by the number of dynamic distractors. Fig. 6 shows the performance trends of the evaluated backends on the HM3D scene.

As the group size G grows from 4 to 24, four of the five backends with our retrieval lose at most seven recall points, and Reloc3r drops from 62.1 % to 32.6 %, whereas every backend with VPR+ or VPR loses more, down to 6.8 % and 0 % for Reloc3r.

Backends using our retrieval also show a smaller P10/P90 spread across distractor samplings. This indicates that appearance retrieval is not only less accurate but also less predictable under visual distractors.

D. Real-World Localization Benchmark

This experiment tests the full system on real queries and compares rendered against real references. We summarize the recall and RMSE localization results for the different backends on our MoCap scene in Tab. III.

Relative to VPR, our retrieval lowers position RMSE in nine of ten backend/split combinations by 15–60 %, and raises recall for the classical, MonST3R, and Reloc3r backends by 2–27 %. For VGGT-Ω and π^3, recall changes by -6.6 to $+2.2$ %, and π^3 in the *dynamic* split is the one case where RMSE rises. Lower RMSE at similar recall means fewer outliers. VPR+ recovers only part of the RMSE gain.

We attribute the smaller gaps relative to simulation (Figs. 5 and 6) to the small $14.2\,\mathrm{m}^2$ scene, which offers little aliasing, and to having only two human distractors.

The lower block runs the classical localizer on two caches of real images of the same moved scene the FRM was built on, now fully mirroring hloc [1]. We pose the PortalCam images from which the 3DGS was built with COLMAP [2] (exhaustive matching, rig extrinsics fixed, intrinsics initialized from vendor values and refined), align them to the scanner's LiDAR-inertial odometry trajectory by one Sim(3) Umeyama alignment [35], and triangulate keypoints with no rendered views or depth. *Single cam* keeps the left pinhole camera's

TABLE III: **Localization evaluation on real-world MoCap data:** Recall@5 cm/5° and position RMSE per backend. Superscripts: relative change w.r.t. the VPR baseline (green better, orange worse). Best per column within each block in **bold**. Our retrieval improves nearly all localizers in RMSE, recall, or both. On the FRM, the classical backend beats the same pipeline on SfM-posed real-image caches in every column.

		Static		Dynamic	
Method	**Retr.**	R@5	RMSE [cm]	R@5	RMSE [cm]
FRM (ours, rendered images, 3DGS based, $1000\,\mathrm{m}^{-3}$)					
Classical	Ours	**0.889**$^{\uparrow 1.8\%}$	**6.3**$^{\downarrow 34.4\%}$	**0.864**$^{\uparrow 4.1\%}$	8.2$^{\downarrow 54.2\%}$
	VPR+	0.878$^{\uparrow 0.6\%}$	6.7$^{\downarrow 30.2\%}$	0.840$^{\uparrow 1.2\%}$	9.7$^{\downarrow 45.8\%}$
	VPR	0.873	9.6	0.830	17.9
MonST3R	Ours	0.729$^{\uparrow 7.2\%}$	7.3$^{\downarrow 54.1\%}$	0.689$^{\uparrow 6.8\%}$	**7.4**$^{\downarrow 43.9\%}$
	VPR+	0.681$^{\uparrow 0.1\%}$	14.5$^{\downarrow 8.8\%}$	0.664$^{\uparrow 2.9\%}$	16.1$^{\uparrow 22.0\%}$
	VPR	0.680	15.9	0.645	13.2
Reloc3r	Ours	0.826$^{\uparrow 15.5\%}$	26.9$^{\downarrow 51.2\%}$	0.778$^{\uparrow 26.7\%}$	16.3$^{\downarrow 59.7\%}$
	VPR+	0.727$^{\uparrow 1.7\%}$	52.1$^{\downarrow 5.4\%}$	0.652$^{\uparrow 6.2\%}$	33.1$^{\downarrow 18.1\%}$
	VPR	0.715	55.1	0.614	40.4
VGGT-Ω	Ours	0.690$^{\downarrow 6.6\%}$	41.6$^{\downarrow 20.6\%}$	0.746$^{\uparrow 2.2\%}$	43.4$^{\downarrow 15.1\%}$
	VPR+	0.746$^{\uparrow 0.9\%}$	57.0$^{\uparrow 8.8\%}$	0.752$^{\uparrow 3.0\%}$	49.9$^{\downarrow 2.3\%}$
	VPR	0.739	52.4	0.730	51.1
π^3	Ours	0.807$^{\downarrow 1.7\%}$	31.9$^{\downarrow 21.8\%}$	0.782$^{\downarrow 2.0\%}$	34.8$^{\uparrow 17.2\%}$
	VPR+	0.827$^{\uparrow 0.7\%}$	34.7$^{\downarrow 15.0\%}$	0.804$^{\uparrow 0.8\%}$	33.4$^{\uparrow 12.5\%}$
	VPR	0.821	40.8	0.798	29.7
Sparse reference cache (real images, posed with COLMAP [2])					
Classical single cam $18\,\mathrm{m}^{-3}$	Ours	0.396$^{\downarrow 39.6\%}$	32.7$^{\uparrow 7.9\%}$	0.374$^{\downarrow 33.2\%}$	43.3$^{\downarrow 35.9\%}$
	VPR+	0.664$^{\uparrow 1.2\%}$	21.1$^{\downarrow 30.4\%}$	0.598$^{\uparrow 6.8\%}$	**16.6**$^{\downarrow 75.4\%}$
	VPR	0.656	30.3	0.560	67.5
Classical four cam rig $66\,\mathrm{m}^{-3}$	Ours	0.649$^{\downarrow 6.9\%}$	**13.9**$^{\downarrow 32.2\%}$	**0.636**$^{\uparrow 6.0\%}$	31.1$^{\downarrow 28.7\%}$
	VPR+	**0.708**$^{\uparrow 1.6\%}$	15.5$^{\downarrow 24.4\%}$	0.615$^{\uparrow 2.5\%}$	21.1$^{\downarrow 51.6\%}$
	VPR	0.697	20.5	0.600	43.6

323 images ($\sim 18\,\mathrm{m}^{-3}$ over the $17.7\,\mathrm{m}^3$ capture volume), *four cam rig* all 1160 ($\sim 66\,\mathrm{m}^{-3}$ over the same volume).

In simulation, references and queries come from the same renderer. The MoCap experiment is the only one that pits real queries against both rendered and real references, shown in the upper and lower block of Tab. III. With the same pipeline, real queries localize better against our FRM (0.864 recall, 8.2 cm RMSE, dynamic) than against SfM-posed real images of the same scene (at best 0.636 and 16.6 cm). Additionally, in Fig. 7 an FRM of only $10\,\mathrm{m}^{-3}$ queried by the classical backend with VPR already exceeds the performance with the $66\,\mathrm{m}^{-3}$ rig cache. The performance cost of the appearance gap between rendered references and real query images is thus outweighed by the gains from controlled, embodiment-specific viewpoint coverage. Acquisition effort is identical, since the FRM and the SfM caches come from the same scan.

We also note that while our system improves most backends, the classical pipeline combines the highest recall in both splits (above 80 %) with an RMSE below 10 cm, and would be our recommendation for real-world deployment.

E. Robust Navigation in an Actively Changing Scene

This experiment is a qualitative end-to-end demonstration of the full system supporting a real navigation task in a busy scene. Given its real-world performance (Sec. IV-D),

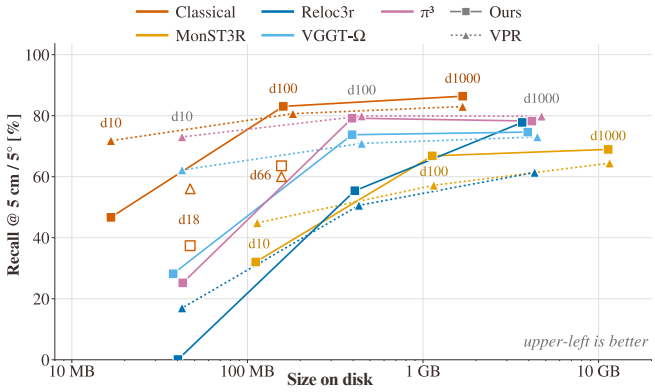


Fig. 7: **Real-World On-Robot Storage:** FRM density–storage trade-off per backend on the *dynamic* MoCap split. Denser references help all localizers under both retrieval methods, but our pose-based retrieval benefits from a sufficiently dense FRM ($\geq 100\,\mathrm{m}^{-3}$). Hollow markers: the classical backend on the SfM-posed real-image caches.

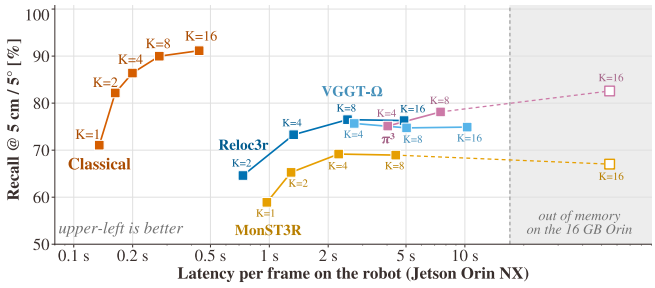


Fig. 8: **Real-World On-Robot Inference:** Inference time–performance trade-off per backend for different context sizes K with our retrieval on the *dynamic* MoCap split. Only the classical backend with $K = 4$ reaches the targeted 5 Hz onboard, while the best trade-off for most localizers lies between $K = 4$ and $K = 8$.

we deploy only the classical backend for hallway navigation, running onboard at 5 Hz with odometry-propagated pose priors (closer to the *Oracle* condition than *Ours*). Across two experiments with four and three goals spaced 30 m apart, the robot *reaches every goal* despite increasing human interference and objects being added and moved (Fig. 1).

We refer the reader to the accompanying video for a more in-depth qualitative analysis and scene inspection of the navigation experiment. There, we also show visual aliasing examples of the appearance-based context retrieval methods.

F. Ablations

1) FRM Density Ablation: We ablate FRM reference density on the *dynamic* real-world split, measuring performance with all retrieval methods along with storage footprint. As shown in Fig. 7, denser FRMs help all localizers regardless of retrieval method. On this split, pose-based retrieval needs at least $100\,\mathrm{m}^{-3}$ to beat VPR for all but π^3, since sparser FRMs lack well-aligned references within the query radius, even when tailored to the robot's morphology. Yet already at $10\,\mathrm{m}^{-3}$, the classical backend with VPR achieves higher recall on our FRM than on either SfM-posed real-image cache (hollow markers).

A light FRM is thus already preferable to SfM-posed caches, and appearance-based retrieval remains advantageous when only a sparse reference set is available. When a dense FRM is affordable, pose-based retrieval is the better choice, particularly in challenging scenes.

Storage scales linearly with density. With the conservative 1 m height band of Tab. II, which lets one FRM serve several camera heights, an FRM at $1000\,\mathrm{m}^{-3}$ requires 120 MB per m^2 of floor area for the classical backend (1.7 GB for our $14.2\,\mathrm{m}^2$ scene), of which 16 MB per m^2 are VPR descriptors. Narrowing the band to account for a single morphology and camera height reduces this proportionally. For demanding, changing scenes we recommend a density of 100–$1000\,\mathrm{m}^{-3}$ at a height band of 10 cm, about 1.2–12 MB per m^2. Note also that our FRM does not need to be sampled uniformly across a large space, so storage can be budgeted per region.

2) Reference Number and Runtime Ablation: Fig. 8 displays end-to-end inference time on the robot's onboard computer (Jetson Orin NX, 16 GB) for $K \in \{1, 2, 4, 8, 16\}$, alongside performance on the *dynamic* MoCap split. Nearly all backends show a predictable time–performance trade-off, with the best balance for most between $K = 4$ and $K = 8$, while multi-view feed-forward models typically fail for $K \leq 2$; MonST3R and π^3 run out of memory at $K = 16$ (shaded region in Fig. 8). Only the classical backend with $K = 4$ reaches the 5 Hz our navigation policy uses.

Our retrieval takes $\sim$4 ms versus $\sim$40 ms for VPR at a density of $1000\,\mathrm{m}^{-3}$, though learned forward passes still dominate inference time. Pre-encoding the rendered reference views into our FRM also avoids K ALIKED [20] ($\sim$70 ms each) or DINO [39] ($\sim$200 ms each) forward passes at runtime, although they can be batched for efficiency.

V. Conclusion

We presented MERLoc, which turns a metric 3D Gaussian Splatting scan into a dense *Functional Reference Memory* for localization: embodiment-aware reference views, rendered offline and pre-encoded for off-the-shelf localizers. On real data, this memory outperforms SfM-posed real-image caches of the same scene. Pose-based retrieval lowers position RMSE by 15–60 % over appearance retrieval in nine of ten real-world backend/split combinations. For most backends, it handles scene change and human activity better than even prior-gated appearance retrieval. We also demonstrate the system in-the-loop for navigation in a highly dynamic hallway.

Because the FRM is rendered rather than captured, it can be edited and extended after the scan. MERLoc already works with scans of furnished rooms, and segmenting movable objects from the scan and rendering references from their occupied space would keep coverage complete after rearrangement. Online FRM updates could track long-term change, and compression would scale FRMs to entire buildings. Characterizing the rendered-to-real gap under increasing image degradation would inform camera and scan requirements. Finally, the formulation is not specific to indoor scenes, making outdoor evaluation under changing illumination and seasons a natural next step.

ACKNOWLEDGMENT

Anthropic's Claude served as an editorial tool for language and consistency. All research content is the authors' own.

REFERENCES

[1] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From Coarse to Fine: Robust Hierarchical Localization at Large Scale," in *CVPR*, 2019, pp. 12 708–12 717.

[2] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *CVPR*, 2016, pp. 4104–4113.

[3] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *CVPR*, 2016, pp. 5297–5307.

[4] S. Lowry *et al.*, "Visual Place Recognition: A Survey," *T-RO*, vol. 32, no. 1, pp. 1–19, 2016.

[5] C. Masone and B. Caputo, "A Survey on Deep Visual Place Recognition," *IEEE Access*, vol. 9, pp. 19 516–19 547, 2021.

[6] T. Sattler *et al.*, "Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions," in *CVPR*, 2018, pp. 8601–8610.

[7] J. Zhang *et al.*, "MonST3R: A Simple Approach for Estimating Geometry in the Presence of Motion," in *ICLR*, 2025, arXiv:2410.03825.

[8] J. Wang *et al.*, "VGGT-Ω," in *CVPR*, 2026, pp. 21 486–21 499, arXiv:2605.15195.

[9] Y. Wang *et al.*, "π^3: Permutation-Equivariant Visual Geometry Learning," in *ICLR*, 2026, arXiv:2507.13347.

[10] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "EWA splatting," *TVCG*, vol. 8, pp. 223–238, 2002.

[11] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM TOG*, vol. 42, no. 4, pp. 1–14, 2023.

[12] S. Dong *et al.*, "Reloc3r: Large-Scale Training of Relative Camera Pose Regression for Generalizable, Fast, and Accurate Visual Localization," in *CVPR*, 2025, pp. 16 739–16 752.

[13] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *ICRA*, vol. 2, 1999, pp. 1322–1328.

[14] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *T-RO*, vol. 33, no. 5, pp. 1255–1262, 2017.

[15] E. Brachmann and C. Rother, "Visual Camera Re-Localization from RGB and RGB-D Images Using DSAC," *TPAMI*, vol. 44, no. 9, pp. 5847–5865, 2022.

[16] E. Brachmann, T. Cavallari, and V. A. Prisacariu, "Accelerated Coordinate Encoding: Learning to Relocalize in Minutes using RGB and Poses," in *CVPR*, 2023, pp. 5044–5053.

[17] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," in *ICCV*, 2015, pp. 2938–2946.

[18] J. Wald, T. Sattler, S. Golodetz, T. Cavallari, and F. Tombari, "Beyond Controlled Environments: 3D Camera Re-localization in Changing Indoor Scenes," in *ECCV*, 2020, pp. 467–487.

[19] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixé, "Understanding the Limitations of CNN-Based Absolute Camera Pose Regression," in *CVPR*, 2019, pp. 3297–3307.

[20] X. Zhao, X. Wu, W. Chen, P. C. Y. Chen, Q. Xu, and Z. Li, "ALIKED: A Lighter Keypoint and Descriptor Extraction Network via Deformable Transformation," *TIM*, vol. 72, pp. 1–16, 2023.

[21] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, "LightGlue: Local Feature Matching at Light Speed," in *ICCV*, 2023, pp. 17 581–17 592.

[22] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An Accurate O(n) Solution to the PnP Problem," *IJCV*, vol. 81, no. 2, pp. 155–166, 2009.

[23] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *CACM*, vol. 24, no. 6, pp. 381–395, 1981.

[24] S. Izquierdo and J. Civera, "Optimal transport aggregation for visual place recognition," in *CVPR*, 2024, pp. 17 658–17 668.

[25] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *ICRA*, 2012, pp. 1643–1649.

[26] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 Place Recognition by View Synthesis," *TPAMI*, vol. 40, no. 2, pp. 257–271, 2018.

[27] V. Panek, Z. Kukelova, and T. Sattler, "MeshLoc: Mesh-Based Visual Localization," in *ECCV*, 2022, pp. 589–609.

[28] A. Moreau, N. Piasco, D. Tsishkou, B. Stanciulescu, and A. de La Fortelle, "LENS: Localization enhanced by NeRF synthesis," in *CoRL*, vol. 164. PMLR, 2022, pp. 1347–1356, arXiv:2110.06558.

[29] V. Ress, J. Meyer, W. Zhang, D. Skuddis, U. Soergel, and N. Haala, "3D Gaussian Splatting aided Localization for Large and Complex Indoor-Environments," *ISPRS Archives*, vol. XLVIII-G-2025, pp. 1283–1290, 2025.

[30] D. Maggio, M. Abate, J. Shi, C. Mario, and L. Carlone, "Loc-NeRF: Monte Carlo Localization using Neural Radiance Fields," in *ICRA*, 2023, pp. 4018–4025.

[31] C. Liu *et al.*, "GS-CPR: Efficient Camera Pose Refinement via 3D Gaussian Splatting," in *ICLR*, 2025.

[32] F. Khatib, D. Moran, G. Trostianetsky, Y. Kasten, M. Galun, and R. Basri, "Generalizable Visual Localization for Gaussian Splatting Scene Representations," in *ICCVW*, 2025, pp. 178–189.

[33] J. Li, B. Sun, L. Di Giammarino, H. Blum, and M. Pollefeys, "ActLoc: Learning to Localize on the Move via Active Viewpoint Selection," in *CoRL*, vol. 305. PMLR, 2025, pp. 1225–1245.

[34] V. Ye *et al.*, "gsplat: An Open-Source Library for Gaussian Splatting," *JMLR*, vol. 26, pp. 34:1–34:17, 2025.

[35] S. Umeyama, "Least-squares estimation of transforma-

tion parameters between two point patterns," *TPAMI*, vol. 13, no. 4, pp. 376–380, 1991.
[36] L. Kästner *et al.*, "Arena-Rosnav: Towards Deployment of Deep-Reinforcement-Learning-Based Obstacle Avoidance into Conventional Autonomous Navigation Systems," in *IROS*, 2021, pp. 6456–6463.
[37] S. K. Ramakrishnan *et al.*, "Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI," in *NeurIPS Datasets and Benchmarks*, 2021.
[38] Z. Xu, X. Han, H. Shen, H. Jin, and K. Shimada, "NavRL: Learning Safe Flight in Dynamic Environments," *RA-L*, vol. 10, no. 4, pp. 3668–3675, 2025.
[39] O. Siméoni *et al.*, "DINOv3," 2025, arXiv:2508.10104.