LOCINAV: POSE-INDEXED VISUAL CACHING AND RETRIEVAL FOR ITERATIVE OBJECT-GOAL NAVIGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language model (VLM) driven navigation agents process streaming egocentric video to find objects in unseen environments, yet they discard the visual features computed at each step, thus losing access to useful observations as the context budget fills. When the agent pursues consecutive goals in the same scene, this per-step forgetting wastes both the information and the computation that earlier exploration has produced. We propose LociNav, which maintains a pose-indexed store of computed visual features that grows as the agent explores. The store is organized by spatial landmarks and viewing direction, keeping it compact without discarding distinct viewpoints. When a new object goal is issued, a scoring model selects the most relevant stored observations and assembles a compact, goal-tailored memory block that is prepended to the model's input. Since the stored features need not be re-encoded and the memory block is arranged to maximize key-value (KV) prefix reuse across navigation steps, the agent consults its memory at reduced computational cost. We further construct an iterative object-goal navigation dataset and benchmark by organizing standard HM3D ObjectNav and HM3D-OVON episodes into tours of consecutive goals and incorporating the natively iterative GOAT-Bench. LociNav achieves state-of-the-art performance across the evaluated benchmarks, with its advantage growing significantly as tours deepen, while achieving over 3 Hz on-device inference.

1 INTRODUCTION

Embodied navigation agents driven by vision-language models (VLMs) have become increasingly capable, learning to find objects, follow instructions, and explore previously unseen environments from streaming egocentric video (Zhang et al., 2024b;a; Wei et al., 2026; 2025). Yet, these agents treat each step in isolation, processing the current observation and discarding it before moving on. A viewpoint the agent has already seen is encoded from scratch if it reappears, whether through backtracking or when a new goal leads through previously explored areas. In iterative settings where the agent pursues consecutive goals in one environment (Krantz et al., 2023; Khanna et al., 2024), this forgetting wastes both the observations and the computation that earlier exploration produced.

Maintaining useful memory across a long navigation session remains an open challenge. Existing context management strategies such as temporal sampling (Zhang et al., 2024b), token merging (Zhang et al., 2024a), and sliding-window key-value (KV) caching (Wei et al., 2025) keep the model's input within budget but progressively lose access to earlier observations. Memory-augmented approaches take the opposite direction, building persistent representations of the explored environment. Classical spatial representations such as occupancy maps (Elfes, 1989; Chaplot et al., 2020a) and topological graphs (Chaplot et al., 2020b) are compact and efficient to query but commit to a fixed set of properties at write time, discarding the raw visual detail. Recent VLM-based methods retain richer information, including synthesized scene snapshots (Yang et al., 2025a) or text-based environment descriptions (Yang et al., 2025b), yet they have to reprocess this information at each decision step, or supply it to the model without selecting for what is most relevant to the agent's current goal.

Navigation platforms such as robots, drones, and autonomous vehicles typically carry inertial and depth sensors that provide continuous pose estimates, tying each observation to a known position and viewing direction. This spatial grounding opens a possibility that existing methods do not exploit (Fig. 1): the visual features the model computes from each observation can be stored persistently,

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

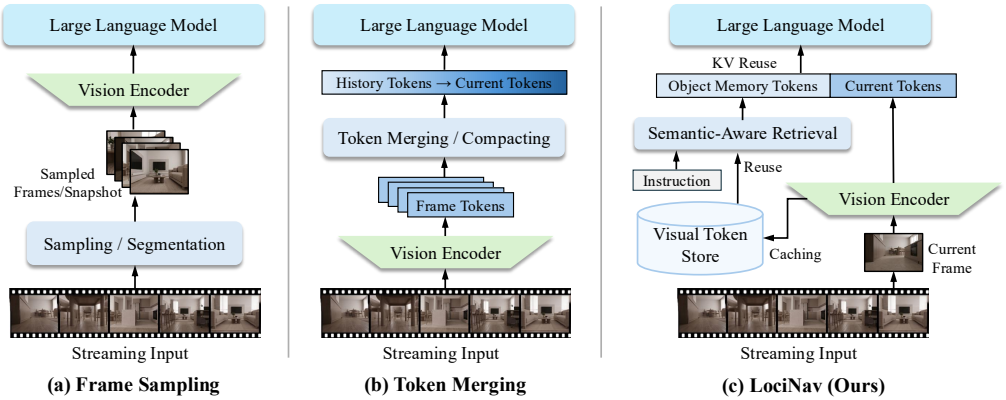


Figure 1: **Comparison of visual memory approaches in VLM-based navigation.** (a) Frame sampling methods select a subset of raw frames before encoding. (b) Token merging methods encode all frames and compact the history tokens. (c) LociNav caches the encoder's output in a persistent store, retrieves goal-relevant observations, and reuses KV states across steps.

indexed by where and how they were taken. A later goal in the same environment can then draw on this accumulated record, not by re-encoding past images but by retrieving features that have been computed. The resulting memory grows with exploration and can be queried selectively based on the current goal, supplying relevant past observations to the agent without the cost of reprocessing them.

We propose LociNav, which realizes this idea through three components. A pose-indexed visual feature store retains the vision encoder's output as the agent explores, organized by spatial landmarks and viewing direction to consolidate redundant viewpoints. When a new object goal is issued, a scoring model ranks stored observations by relevance to the goal category and assembles a compact memory block that is prepended to the VLM's input. The memory block combines candidate frames that indicate where the target may be found, path frames that preview the route toward it, and live frames that provide local guidance. Candidate and path frames persist as long as the agent makes progress toward the target, while live frames update as the agent moves, and the block is recomposed when the agent switches to a new candidate. By placing the stable frames at the front of the block, the prefix remains largely unchanged across consecutive steps, enabling KV cache reuse. Since the store holds already-computed features, supplying them requires no re-encoding, and the stable KV prefix structure further reduces prefilling cost over long navigation horizons.

To train and evaluate LociNav on navigation with persistent memory, we construct an iterative object-goal navigation dataset and benchmark by organizing episodes from standard HM3D ObjectNav (Ramakrishnan et al., 2021) and HM3D-OVON (Yokoyama et al., 2024b) splits into tours where the agent pursues consecutive goals within the same scene without being reset, and incorporate GOAT-Bench (Khanna et al., 2024) which is natively iterative. Experiments across these benchmarks show that LociNav achieves state-of-the-art success rate across the evaluated benchmarks, with its advantage growing significantly as tours deepen and the memory store accumulates observations. The ViT cache and KV prefix reuse enable LociNav to utilize persistent memory at only around 40 ms additional per-step latency over the memoryless baseline, achieving 3 Hz inference on a workstation GPU and over 0.6 Hz on a resource-limited edge GPU.

2 Related Work

VLM-based Embodied Navigation. Object-goal navigation requires an agent to locate a target object in an unseen environment (Ramakrishnan et al., 2021). Recent work leverages large language models for greater flexibility, either as modular planners reasoning over text descriptions of the scene (Long et al., 2024; Shah et al., 2023; Rajvanshi et al., 2024) or as end-to-end vision-language-action (VLA) models that process streaming egocentric video and output low-level actions directly (Zhang et al., 2024b;a; Cheng et al., 2024; Wei et al., 2026; 2025). The latter operate on-device and in real time, avoiding the latency of cloud inference and the dependence on panoramic sensors or pre-built maps. However, the video stream grows continuously and quickly exceeds the model's context capacity. Existing methods address this through online token merging (Zhang et al., 2024a), sliding-window context with spatial pruning (Wei et al., 2025), or discrete KV cache grouping (Yang et al., 2025b), all

of which compress or discard past observations to fit within a fixed budget. LociNav takes a different approach: it stores the visual features the encoder has already computed, indexed by pose, so that past observations can be recalled without re-encoding.

Memory for Navigation Agents. Equipping navigation agents with persistent spatial memory has a long history. Classical approaches maintain occupancy grids (Elfes, 1989), topological maps (Chaplot et al., 2020b), or scene graphs (Gu et al., 2024), but commit to a fixed representation at write time and discard the raw visual detail. Recent VLM-based methods retain richer information, such as synthesized scene snapshots (Yang et al., 2025a) or text-based environment descriptions (Yang et al., 2025b), yet they reprocess this information at each decision step or cache text rather than visual computation. LociNav stores the vision encoder's output directly, indexed by pose, so that retrieval involves no re-encoding and the cached features can be reused during inference.

Object-Goal Navigation. Object-goal navigation spans modular systems with learned semantic maps (Chaplot et al., 2020a; Yokoyama et al., 2024a; Busch et al., 2025), LLM-based planners (Yang et al., 2025b), and end-to-end policies (Wijmans et al., 2019; Ramrakhya et al., 2023; Yadav et al., 2023b). Standard benchmarks such as HM3D ObjectNav (Yadav et al., 2023a) and HM3D-OVON (Yokoyama et al., 2024b) evaluate each episode independently, resetting the agent between goals. IVLN (Krantz et al., 2023) introduced iterative evaluation for vision-and-language navigation, MultiON (Wani et al., 2020) formulated sequential multi-object search without resets, and GOAT-Bench (Khanna et al., 2024) evaluates multi-goal navigation with mixed modalities. We extend iterative object-goal evaluation to photorealistic HM3D scenes and open-vocabulary categories by organizing standard episodes into tours of consecutive goals, directly testing whether accumulated experience improves later navigation.

3 Design of LociNav

LociNav maintains a pose-indexed store of visual features the vision encoder produces throughout the tour. The store persists across episode boundaries, making past observations available to subsequent object navigation episodes in the same scene. Fig. 2 illustrates the overall design. As the agent navigates, LociNav indexes each encoded frame by the agent's position and viewing direction, retaining a compact set of representative observations per location (Sec. 3.1). When a new object goal is issued, a scoring model ranks stored frames by relevance to the goal category and assembles a memory block that is prepended to the VLM's input (Sec. 3.2). By retaining the encoder's output rather than raw frames, LociNav avoids redundant visual encoding throughout the tour. The memory block is carefully arranged so that its prefix remains stable across consecutive steps, enabling KV cache reuse to reduce inference cost over long navigation horizons (Sec. 3.3).

3.1 Pose-Indexed Visual Feature Store

As the agent navigates, it accumulates egocentric observations. To make these observations available to future object navigation episodes within the same tour, LociNav maintains a persistent memory store $\mathcal{S}$ that records the visual features the vision encoder has already computed, indexed by the pose at which each observation was taken. Concretely, for each frame $\mathbf{x}_t$ observed at pose $\mathbf{p}_t = (x_t, y_t, \theta_t)$, the store retains the visual feature $\mathbf{v}_t$ produced by the vision encoder together with the pose, as the pair $(\mathbf{v}_t, \mathbf{p}_t)$. Since the store holds the encoder's output rather than raw frames, a stored observation can later be supplied to the VLM directly without re-running the visual encoder.

Landmark-based indexing. Naively retaining every encoded frame would cause the store to grow without bound. To keep it compact, LociNav organizes the store around spatial landmarks extracted from the agent's trajectory. As the agent moves, its path forms a walk graph, and a set of landmarks $\mathcal{L}$ is derived from this graph. Each incoming frame is assigned to the nearest landmark $\ell \in \mathcal{L}$, and the viewing directions around each landmark are partitioned into K angular sectors. Each sector retains a single representative frame, preferring the most recent observation and breaking ties by proximity to the sector center. The store thus keeps at most K frames per landmark, ensuring that observations from the same position and similar viewing directions are consolidated, while observations from different directions are kept separately. The entire process is streaming: each frame is indexed in $O(1)$ time as it arrives, and the store persists across episode boundaries throughout the tour.

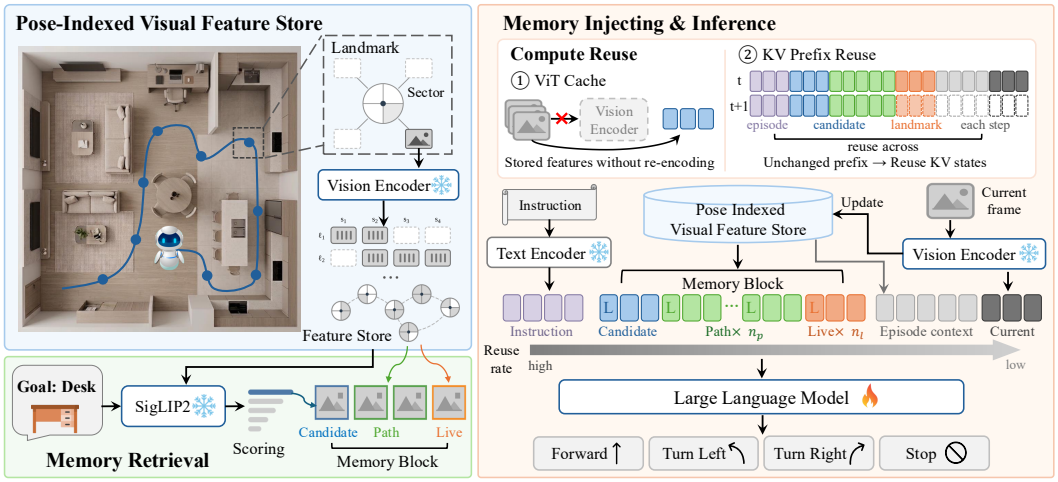


Figure 2: **Architecture of LociNav.** The agent's visual features are indexed by landmark and viewing direction into a persistent store (left). When a new instruction arrives, relevant frames are scored and assembled into a memory block of candidate, path, and live frames (bottom left). The memory block is prepended to the VLM's input, with a ViT cache that avoids re-encoding and tokens ordered by update frequency to maximize KV prefix reuse (right).

3.2 Memory Retrieval and Injection

Candidate scoring. At the start of each object navigation episode, LociNav scores every frame in the store that was recorded during all prior episodes against the current goal category using an image-text matching model (Tschannen et al., 2025), whose image embeddings are stored alongside the visual features in the store. The scores are reduced to one value per landmark by taking the maximum over all frames at that landmark. To balance relevance against reachability, the landmarks are then ranked by the ratio of normalized score to graph distance from the agent's current position:

$$\ell^* = \arg\max_{\ell} \frac{\tilde{s}_\ell}{d_{\text{graph}}(\mathbf{p}_{\text{agent}}, \mathbf{p}_\ell) + d_0}, \tag{1}$$

where $\tilde{s}_\ell$ is the score of landmark ℓ clipped and rescaled to $[0, 1]$, d_{graph} is the shortest-path distance on the walk graph, and d_0 is a constant that prevents the ratio from diverging for nearby landmarks. The highest-scoring frame at ℓ^* is taken as the candidate frame. The candidate table is frozen for the entire episode. Landmarks whose sector the agent has already visited are masked out, so the agent does not pursue locations it has already inspected.

Slot assembly. Once the candidate frame is determined, LociNav computes a shortest path on the walk graph from the agent's current landmark to the candidate's landmark. n_p path frames are sampled at evenly spaced intervals along this route, each oriented in the direction of travel, previewing the route toward the candidate. n_l live frames are selected at the next landmarks along the path, also facing the direction of travel, providing immediate guidance for the agent's next steps. The memory block thus contains up to $m = 1 + n_p + n_l$ frames.

Memory injection. The selected memory frames are assembled into a compact block and prepended to the VLM's input, followed by the episode's visual stream compressed via online token merging (Zhang et al., 2024a). Each memory frame contributes n_v visual tokens, produced by spatially pooling the stored feature $\mathbf{v}_t$, plus one label token described below. This keeps the memory block and the episode context separate, so that the model retains its full view of the ongoing episode while gaining access to past observations. The memory block spans $m \cdot (n_v + 1)$ tokens in total.

Memory frame encoding. Each memory frame carries a label token prepended to its visual tokens. The label token is a learned type embedding that identifies the frame's role (candidate, path, or live). To communicate the spatial relationship between memory frames and the current observation, we add a pose-aware bias directly to the attention logits. Encoding raw spatial coordinates as input tokens would be difficult for the LLM to interpret, and since the agent moves continuously, any coordinate-based representation would require recomputing the memory block and break the stable KV prefix that enables cache reuse. An attention-level bias avoids both issues: it depends only on the

displacement between query and key poses and leaves cached key-value states untouched. The bias for attention head h between a query token at pose $\mathbf{c}_q$ and a key token at pose $\mathbf{c}_k$ is

$$b_h(\mathbf{c}_q, \mathbf{c}_k) = \frac{g_h}{2M} \sum_{j=1}^{M} \big[\cos(s_j \cdot \Delta c_j) + \sin(s_j \cdot \Delta c_j)\big], \tag{2}$$

where Δc_j is the relative displacement along the spatial axis assigned to frequency j, s_j is its frequency scale, g_h is a learnable per-head gate, and $M = 20$ is the total number of frequencies, each contributing both a cosine and a sine term. The cosine component encodes spatial proximity while the sine component resolves directional ambiguity. Translational displacements are projected into the agent's egocentric frame before encoding. Yaw frequencies are constrained to integers to ensure periodicity across the $\pm\pi$ boundary, while translational frequencies follow a geometric ladder from scene scale down to the store's spatial resolution. Each gate is initialized calibrated to the backbone's attention logit scale so that training begins close to the pretrained behavior, and the model gradually learns which heads benefit from spatial awareness. Together, the type embedding and spatial bias add only around 13K new parameters.

Memory block updates. The candidate frame and path frames are selected when a new object goal is issued and remain in place during navigation. If the agent fails to get closer to the candidate for a sustained number of steps, the current candidate is retired and the next entry from the candidate table takes its place, with path frames recomputed from the agent's current position. The live frame updates more frequently, recomputed each time the agent crosses a landmark boundary to point to the next hop toward the current candidate. As a result, the memory block exhibits a layered stability: the candidate and path frames form a stable prefix that persists over long stretches of navigation, while only the trailing live frame adapts to the agent's local movement. This ordering is chosen to maximize the length of the reusable prefix, enabling the KV cache reuse mechanism (Sec. 3.3). The complete per-step retrieval procedure is summarized in Sec. A.

3.3 ViT Cache and KV Prefix Reuse

Equipping a navigation agent with memory typically increases inference cost, as additional observations lengthen the prompt and demand more computation at each step (Yang et al., 2025b; Wei et al., 2025). LociNav offsets this cost through two reuse mechanisms.

Vision Transformer (ViT) cache. When a stored frame is selected for the memory block, its visual feature $\mathbf{v}_t$ is read directly from the store and does not pass through the vision encoder again. Over the course of a tour, the store accumulates observations from an increasing number of locations, yet the encoding cost of each observation is paid only once, at the time it is first observed. The ViT cache thus decouples the cost of maintaining a large memory from the cost of using it.

KV prefix reuse. At each navigation step, the VLM processes the full input during prefilling to populate its key-value (KV) cache before generating an action. As the memory block occupies the leading positions of the input, and the candidate and path frames within it remain unchanged as long as the current candidate persists, the KV states computed for this prefix can be carried across steps without recomputation. Only the trailing live frame, which changes when the agent crosses a landmark boundary, and the subsequent episode context need to be processed anew. The prefilling cost is therefore reduced to the few tokens that actually changed between steps.

Together, the ViT cache and KV prefix reuse allow LociNav to carry a persistent and growing memory throughout a tour while keeping inference cost comparable to a memoryless baseline. Both mechanisms draw on the same pose-indexed store, requiring no auxiliary data structures.

4 Dataset and Training

HM3D ObjectNav and OVON. Standard object-goal navigation benchmarks such as HM3D ObjectNav (Yadav et al., 2023a) and HM3D-OVON (Yokoyama et al., 2024b) evaluate each episode independently, resetting the agent between tasks. To create an iterative navigation setting, we organize episodes from the same scene into tours: within a tour, the agent navigates consecutive object goals without being reset, guided to each new starting position between episodes so that the full trajectory remains continuous. We refer to the resulting benchmarks as ObjectNav-Tour and OVON-Tour. For

evaluation, ObjectNav-Tour covers 1,000 episodes and OVON-Tour covers 3,000 episodes per split, all across 36 HM3D v2 (Ramakrishnan et al., 2021) scenes. Each official evaluation episode appears exactly once across all tours, so results reported by methods that evaluate episodes independently remain directly comparable. For training, we apply the same construction to the training splits (145 scenes), producing 404,271 training frames. Further construction details are provided in Sec. B.

GOAT-Bench. We also train and evaluate on GOAT-Bench (Khanna et al., 2024), a multi-goal navigation benchmark in which each episode consists of 5 to 10 consecutive sub-goals within the same scene. Unlike ObjectNav and OVON, GOAT episodes are already iterative by design, so no tour stitching is required. Goal modalities in GOAT are mixed. Following prior work (Ren et al., 2025), we represent all goals as text strings. GOAT provides 360 evaluation episodes per evaluation split across 36 scenes. From its training split, we use 2,310 episodes totaling 295,501 frames.

Training objective. At each training step, the model predicts an answer string containing a visibility tag followed by the next four navigation actions. The visibility tag describes the agent's proximity to the current goal using the same distance-to-viewpoint criterion as the evaluation metric, so that learning to predict visibility and learning when to stop are aligned. The training loss is standard next-token cross-entropy applied to the answer span with uniform weight across tokens. No auxiliary loss terms or prediction heads are introduced: the new supervision signal is packed into the same predicted string as the actions, adding zero hyperparameters and keeping the loss function identical between the memory-equipped and baseline models. Details are provided in Sec. C.

Training. We train on a mixture of the ObjectNav-Tour, OVON-Tour, and GOAT-Bench training splits, totaling approximately 700K frames. Following Zhang et al. (2024a), the model uses EVA-CLIP (Sun et al., 2023) as the vision encoder and Vicuna-7B (Chiang et al., 2023) as the language backbone. The vision encoder is kept frozen throughout training. We add approximately 13K new parameters for the type embedding and spatial pose bias (Sec. 3.2) and fine-tune the full language model. Following the training strategy of visual instruction tuning (Liu et al., 2023), we optimize for a single epoch with a learning rate of 1×10^{-5}, cosine schedule, and an effective batch size of 128. Training takes approximately 200 GPU hours on NVIDIA RTX 6000 Ada GPUs.

5 Experiments

5.1 Experimental Setup

Implementation details. We set $K = 4$ angular sectors per landmark, matching the camera's horizontal field of view so that one frame approximately fills its sector. The memory block contains $n_p = 2$ path frames and $n_l = 1$ live frame, for a maximum of $m = 4$ memory frames. Each memory frame is represented by $n_v = 4$ visual tokens produced by coarse spatial pooling (Li et al., 2024), preserving the spatial structure of selectively retrieved frames. The spatial pose bias uses $M = 20$ cosine frequency terms across three channels (4 for yaw, 8 each for x and y), with a learnable per-head gate on each of the attention heads. The candidate scoring applies a distance constant $d_0 = 64\,\text{m}$. The selection of all above parameters is detailed in Sec. D. We further deploy LociNav on a workstation with a single RTX 4090 GPU and on an NVIDIA Jetson Orin to measure inference latency in resource-constrained settings.

Benchmarks and metrics. We evaluate on three benchmark families described in Sec. 4: ObjectNav-Tour, OVON-Tour, and GOAT-Bench. For ObjectNav-Tour and OVON-Tour, only the top rung of each tour is evaluated to avoid over-counting shallow episodes. The agent receives pose from the simulator at each step. We adopt the standard evaluation metrics (Anderson et al., 2018) Success Rate (SR) and Success weighted by Path Length (SPL) to evaluate navigation accuracy. For system efficiency, we follow Yang et al. (2025b) to report (i) real-time latency (RtL): the average agent planning time in one navigation step, and (ii) end-to-end latency (E2EL): the average latency for the agent to complete one navigation task, including multiple navigation steps and moving time.

5.2 Object-Goal Navigation Benchmarks

ObjectNav-Tour and OVON-Tour. As shown in Tab. 1, LociNav achieves the highest SR across all four evaluation splits. On ObjectNav-Tour, LociNav reaches 78.5 SR, surpassing Qwen-RobotNav (Zhang et al., 2026b) (75.6) and Uni-NaVid (Zhang et al., 2024a) (73.7). On the three

Table 1: **SR and SPL comparison on ObjectNav-Tour and OVON-Tour.** "Pano." and "Pose" indicate whether panoramic images and pose are used as input. "OD" indicates on-device VLM inference rather than cloud-based planning. **Bold**: best; underline: second best.

Method	Attributes			ObjectNav-Tour		OVON-Tour					
						Seen		Synonyms		Unseen	
	Pano.	Pose	OD	SR	SPL	SR	SPL	SR	SPL	SR	SPL
Single-Episode Navigation											
VLFM (Yokoyama et al., 2024a)		✓	✓	52.5	30.4	35.2	18.6	32.4	17.3	35.2	19.6
CogNav (Cao et al., 2025)		✓		72.5	26.2	-	-	-	-	-	-
MTU3D (Zhu et al., 2025)		✓	✓	-	-	55.0	23.6	45.0	14.7	40.8	12.1
Uni-NaVid (Zhang et al., 2024a)			✓	73.7	**37.1**	41.3	21.1	43.9	21.8	39.5	19.8
NavFoM (Zhang et al., 2026a)	✓		✓	-	-	40.1	27.1	45.4	32.6	45.2	31.9
ABot-N0 (Chu et al., 2026)	✓			-	-	55.3	32.1	55.4	**33.2**	54.0	30.5
Qwen-RobotNav (Zhang et al., 2026b)			✓	75.6	30.6	57.7	24.4	60.1	25.1	53.1	20.9
Iterative Navigation											
AstraNav-Memory (Ren et al., 2025)	✓		✓	-	-	62.1	**34.2**	57.2	33.0	55.5	**32.5**
Uni-NaVid-Mem (Zhang et al., 2024a)			✓	65.5	32.0	37.1	18.6	39.5	17.7	36.9	16.8
LociNav		✓	✓	**78.5**	36.3	**65.5**	32.1	**62.4**	32.7	**58.0**	31.9

Table 2: **SR and SPL comparison on GOAT-Bench (Khanna et al., 2024).** "Pano." and "Pose" indicate whether panoramic images and pose are used as input. "OD" indicates on-device VLM inference rather than cloud-based planning. **Bold**: best; underline: second best.

Method	Attributes			Seen		Synonyms		Unseen	
	Pano.	Pose	OD	SR	SPL	SR	SPL	SR	SPL
Modular GOAT (Chang et al., 2023)		✓	✓	26.3	17.5	33.8	24.4	24.9	17.2
Modular CLIP on Wheels (Chang et al., 2023)		✓	✓	14.8	8.7	18.5	11.5	16.1	10.4
SenseAct-NN Skill Chain (Khanna et al., 2024)		✓	✓	29.2	12.8	38.2	15.2	29.5	11.3
SenseAct-NN Monolithic (Khanna et al., 2024)		✓	✓	16.8	9.4	18.5	10.1	12.3	6.8
3DMem (Yang et al., 2025a)		✓		-	-	-	-	42.6	22.8
EvoMemNav (Ge et al., 2026)		✓		-	-	-	-	59.6	38.9
MTU3D (Zhu et al., 2025)		✓	✓	52.2	30.5	48.4	30.3	47.2	27.7
AstraNav-Memory (Ren et al., 2025)	✓		✓	65.5	**49.0**	**66.8**	**54.7**	62.7	**56.9**
LociNav		✓	✓	**67.1**	45.5	66.0	53.8	**64.6**	51.1

OVON-Tour splits, LociNav outperforms the iterative baseline AstraNav-Memory (Ren et al., 2025) by +3.4, +5.2, and +2.5 SR while operating with a single forward-facing camera rather than panoramic observations, which shows that goal-directed retrieval of stored visual observations effectively guides the agent toward goal objects. Uni-NaVid-Mem, a variant of Uni-NaVid (Zhang et al., 2024a) that retains its visual memory across episodes without resetting, degrades in both SR and SPL compared to the single-episode Uni-NaVid on all splits, as the extended history forces more aggressive token compression that dilutes the information density of past observations. LociNav achieves SPL close to ABot-N0 (Chu et al., 2026) and AstraNav-Memory but slightly lower on several splits. This is because their panoramic observations provide broader scene coverage for route planning, while LociNav tends to navigate toward stored locations along memorized paths to improve goal-finding rate but at the cost of longer paths.

GOAT-Bench. As shown in Tab. 2, LociNav outperforms AstraNav-Memory (Ren et al., 2025) by +1.6 and +1.9 SR on the Seen and Unseen splits but is slightly lower on Synonyms (66.0 vs. 66.8), while operating with a single forward-facing camera rather than panoramic observations. LociNav achieves higher SPL than all other baselines but lower than AstraNav-Memory, as panoramic observations provide broader scene coverage that benefits path efficiency. The gains of SR on GOAT-Bench are smaller than on ObjectNav-Tour and OVON-Tour, which is consistent with the shorter episode structure of GOAT-Bench: each episode contains 5 to 10 sub-goals (Tab. 7) compared to tours of up to 28 sub-goals, leaving fewer stored observations for retrieval.

Effect of tour depth. Fig. 3 compares SR and SPL at different tour depths on the OVON-Tour unseen split, where depth indicates the position of an episode within the tour. Uni-NaVid (Zhang et al., 2024a) processes each episode independently and maintains flat SR and SPL across all depths. Uni-NaVid-Mem, which carries observations across episodes, gradually degrades in both metrics, as the fixed context budget forces increasingly token compression over the growing history. LociNav achieves SR rising from 41.0 to 61.6 across the tour, as the pose-indexed store accumulates observations

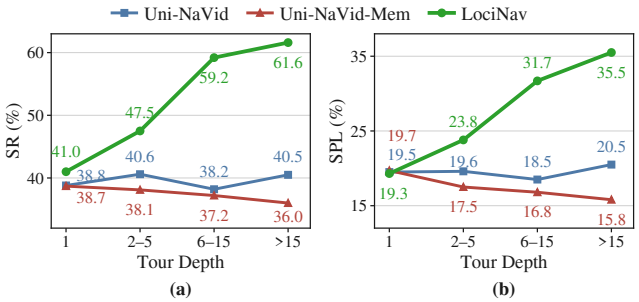


Figure 3: SR and SPL comparison at different tour depth on OVON-Tour Unseen.

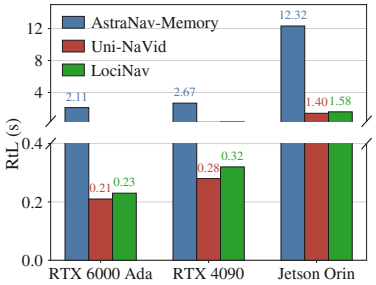


Figure 4: Real-time latency (RtL) comparison across platforms.

that provide increasingly complete scene coverage, and goal-directed retrieval supplies better target memory. SPL follows a similar upward trend (19.3 to 35.5) but with a smaller gain, as memory-guided navigation directs the agent along stored paths toward recalled target locations rather than necessarily the shortest route. Complete results across all splits are shown in Sec. E.1.

5.3 Efficiency of LociNav

Real-time and end-to-end latency comparison. Tab. 3 compares the average RtL and E2EL on OVON-Tour unseen split with an RTX 6000 Ada GPU. CogNav (Cao et al., 2025) queries a cloud-based LLM for waypoint-level planning, which incurs the highest per-call latency (3.54 s) but requires fewer model calls than the other methods that plan continuous actions at every step. Uni-NaVid (Zhang et al., 2024a) and Uni-NaVid-Mem perform VLM inference with a fixed and compact token budget, achieving the lowest RtL (0.21 s) regardless of whether observations are carried across episodes. AstraNav-Memory (Ren et al., 2025) feeds up to 200 history frames into the VLM with vision encoding and compression at each step, which provides rich visual memory and achieves high SR and SPL (as shown in Tab. 1) but results in significantly higher RtL (2.11 s) and E2EL (118.47 s). LociNav also prepends a memory block to the VLM input but avoids redundant visual encoding through its ViT cache and arranges the memory tokens by update frequency so that the stable KV prefix can be reused across consecutive steps (Sec. 3.3) to reduce KV prefilling overhead. This yields an RtL of only 0.24 s, marginally above Uni-NaVid. Furthermore, LociNav achieves the lowest E2EL (17.03 s): compared to AstraNav-Memory, this stems from the per-step inference speedup (9× lower RtL), while compared to the memoryless Uni-NaVid, it results from fewer navigation steps due to memory-guided goal finding.

Table 3: **RtL and E2EL comparison.** "Mem": persistent memory across episodes. "OD": on-device inference.

Method	Mem	OD	RtL (s)	E2EL (s)
CogNav (Cao et al., 2025)	✗	✗	3.54	56.52
Uni-NaVid (Zhang et al., 2024a)	✗	✓	**0.21**	24.56
Uni-NaVid-Mem (Zhang et al., 2024a)	✓	✓	**0.21**	28.90
AstraNav-Memory (Ren et al., 2025)	✓	✓	2.11	118.47
LociNav	✓	✓	0.24	**17.03**

Latency comparison on different platforms. We deploy two on-device VLM baselines and LociNav on three platforms to evaluate per-step planning latency. As shown in Fig. 4, LociNav achieves RtL significantly lower than AstraNav-Memory (Ren et al., 2025) across all platforms (9.2× on RTX 6000 Ada, 8.3× on RTX 4090, and 7.8× on Jetson Orin), while maintaining only marginally higher latency than Uni-NaVid (Zhang et al., 2024a), which operates with a compact token budget. LociNav thus attains navigation accuracy comparable to AstraNav-Memory at latency close to Uni-NaVid, as goal-directed memory retrieval efficiently supplies target-relevant observations without re-encoding a long visual history. On the RTX 6000 Ada and RTX 4090, LociNav achieves approximately 4 Hz and 3 Hz inference speed, and even on the resource-constrained Jetson Orin it sustains over 0.6 Hz.

5.4 Ablation Study

Effectiveness of slots in the memory block. Tab. 4 reports a cumulative ablation on the frame slots of the memory block, evaluated on a fixed OVON-Tour subset. Without memory, the fine-tuned model performs comparably to the Uni-NaVid baseline (Tab. 1). Supplying only the candidate frame slightly

lowers both SR and SPL (40.3/20.5 vs. 41.2/22.0), as a single distant observation without spatial context does not help the model decide where to navigate. Adding path frames recovers and improves performance (45.4 SR) by previewing the route toward the candidate, while adding live frames yields a larger gain (53.4 SR) by providing immediate guidance around the agent's current position. The full memory block with all three slots achieves the best result (61.6 SR, 31.6 SPL), as it supplies both a complete route toward the recalled target and local directional cues.

Table 4: **Ablation study on the memory block structure.**

Cand.	Path	Live	SR	SPL
✗	✗	✗	41.2	22.0
✓	✗	✗	40.3	20.5
✓	✓	✗	45.4	24.7
✓	✗	✓	53.4	28.1
✓	✓	✓	**61.6**	**31.6**

Effectiveness of ViT cache and KV prefix reuse. Tab. 5 ablates the two reuse mechanisms. Without either, the model re-encodes all memory frames and re-prefills the full input at every step, doubling latency over the non-memory backbone (0.41 vs. 0.21 s). Enabling the ViT cache eliminates redundant encoding of up to 4 frames per step, reducing RtL to 0.28 s. KV prefix reuse further avoids re-prefilling the stable memory block into the LLM's KV cache, bringing RtL down to 0.25 s. We detail the RtL breakdown in Sec E.2. Together, the two mechanisms add only around 40 ms per step over the non-memory baseline, while the improved path efficiency (SPL) results in a lower overall task completion time (19.24 vs. 25.23 s).

Table 5: **Ablation study on the ViT and KV reuse modules.**

ViT Cache	KV Reuse	RtL (s)	E2EL (s)
✗	✗	0.41	29.64
✗	✓	0.36	26.85
✓	✗	0.28	22.12
✓	✓	**0.25**	**19.24**
non-memory baseline		0.21	25.23

Comparison of frame pose injection approaches. Tab. 6 compares different approaches for communicating memory frame poses to the model. Adding an MLP-based pose embedding to the label token yields SR close to the no-injection baseline in both world (44.2) and agent-relative (46.3) coordinates, indicating that the LLM struggles to interpret explicit spatial coordinates as input tokens. Overwriting the backbone's RoPE frequency pairs with spatial coordinates improves SR to 52.8 but repurposes model capacity originally devoted to sequential position encoding. Agent-relative representations and RoPE both encode coordinates that shift as the agent moves, requiring recomputation of the memory block and preventing KV prefix reuse. The additive attention bias avoids these issues: it operates on the displacement between query and key poses at the attention logit level, leaving cached key-value states untouched and enabling full prefix reuse. It achieves the highest SR (61.6) and SPL (31.6) at the lowest RtL (0.25 s).

Table 6: **Comparison of different memory frame pose injection approaches.**

Method	SR	SPL	RtL (s)
w/o pose injection	44.4	22.7	0.25
MLP embedding (world)	44.2	23.0	0.25
MLP embedding (agent)	46.3	23.4	0.29
RoPE	52.8	25.9	0.28
Additive attention bias (ours)	**61.6**	**31.6**	**0.25**

Comparison of visual cache approaches. We further compare caching raw images against caching ViT features at different spatial resolutions. Re-encoding raw images at each step incurs 0.85 s RtL even after landmark-and-sector filtering, while caching the full ViT output (n_v=64) eliminates re-encoding and matches raw-image SR (65.0) at 0.70 s but consumes 1.85 GB per tour. Spatially pooling to n_v=4 reduces storage to 0.15 GB and RtL to 0.25 s at a modest SR trade-off (61.6), providing the best balance of accuracy, latency, and storage. Full results are provided in Sec. E.3.

6 Conclusion

We proposed LociNav, which equips VLM-based navigation agents with a pose-indexed store of computed visual features for iterative object-goal navigation. When a new goal is issued, LociNav scores stored observations by relevance to the goal category and assembles a compact memory block tailored to the target. The ViT cache and KV prefix reuse substantially reduce the cost of utilizing memory. Experiments on our self-constructed benchmarks demonstrate state-of-the-art navigation accuracy that improves with tour depth, at per-step latency comparable to the memoryless baseline.

Limitations and Future Work. Our evaluation is conducted in the Habitat simulator using image frames and poses provided by the simulator. While navigation platforms commonly provide continuous pose estimates through visual-inertial odometry or SLAM, real-world deployment introduces pose noise that may affect store indexing and retrieval quality. The design of LociNav is architecture-agnostic, though the quality of candidate scoring and visual understanding depends on the image-text matching model and the LLM backbone. We leave the extension to other VLMs and real-world deployment with onboard sensors in dynamic environments as future work.

AI USE STATEMENT

We used large language models for the following tasks. To implement code: model, training, and evaluation code written with LLM assistance was reviewed and tested by the authors, and the design of the method and of every experiment is our own. To draft and edit manuscript text from author-supplied results and arguments, with all content reviewed and revised by the authors. We did not use generative AI tools to design the method, to design experiments, or to interpret results.

ETHICS STATEMENT

This work uses the publicly released HM3D (Ramakrishnan et al., 2021) and GOAT-Bench (Khanna et al., 2024) datasets under their respective licenses and involves no experiments with human subjects. All experiments are conducted in the Habitat simulator (Savva et al., 2019) with no real-world data collection. As with other embodied navigation systems, the method could be applied to autonomous agents operating in real environments. Deployment in settings involving human occupants should follow appropriate privacy and safety guidelines.

REPRODUCIBILITY STATEMENT

We will release our code, trained model weights, and dataset construction scripts upon acceptance. Sec. 3 describes the full system design, including the store organization, retrieval algorithm, and pose bias mechanism. Sec. 4 details the dataset and training procedure, including the tour construction protocol, expert policy, and training objective. Sec. 5.1 specifies the implementation details and evaluation metrics. Sec. B provides additional dataset construction details and evaluation statistics, Sec. C gives the full training hyperparameters, and Sec. D documents the design hyperparameter selection. Algorithm 1 in Sec. A provides the complete per-step retrieval procedure.

REFERENCES

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

Finn Lukas Busch, Timon Homberger, Jesús Ortega-Peimbert, Quantao Yang, and Olov Andersson. One map to find them all: Real-time open-vocabulary mapping for zero-shot multi-object navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14835–14842. IEEE, 2025.

Yihan Cao, Jiazhao Zhang, Zhinan Yu, Shuzhen Liu, Zheng Qin, Qin Zou, Bo Du, and Kai Xu. CogNav: Cognitive process modeling for object goal navigation with LLMs. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9550–9560. IEEE, 2025.

Matthew Chang, Theophile Gervet, Mukul Khanna, Sriram Yenamandra, Dhruv Shah, So Yeon Min, Kavit Shah, Chris Paxton, Saurabh Gupta, Dhruv Batra, et al. GOAT: Go to any thing. *arXiv preprint arXiv:2311.06430*, 2023.

Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020a.

Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological SLAM for visual navigation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12872–12881. IEEE, 2020b.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. NaVILA: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*, 2024.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality. 2023.

Zedong Chu, Shichao Xie, Xiaolong Wu, Yanfen Shen, Minghua Luo, Zhengbo Wang, Fei Liu, Xiaoxu Leng, Junjun Hu, Mingyang Yin, et al. ABot-N0: Technical report on the VLA foundation model for versatile embodied navigation. *arXiv preprint arXiv:2602.11598*, 2026.

Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

Zuhao Ge, Xiaosong Jia, Chao Wu, Yuchen Zhou, Zuxuan Wu, and Yu-Gang Jiang. EvoMemNav: Efficient self-evolving fine-grained memory for zero-shot embodied navigation. *arXiv preprint arXiv:2606.03509*, 2026.

Qiao Gu, Ali Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, et al. ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5021–5028. IEEE, 2024.

Mukul Khanna, Ram Ramrakhya, Gunjan Chhablani, Sriram Yenamandra, Theophile Gervet, Matthew Chang, Zsolt Kira, Devendra Singh Chaplot, Dhruv Batra, and Roozbeh Mottaghi. GOAT-Bench: A benchmark for multi-modal lifelong navigation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16373–16383. IEEE, 2024.

Jacob Krantz, Shurjo Banerjee, Wang Zhu, Jason Corso, Peter Anderson, Stefan Lee, and Jesse Thomason. Iterative vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Yanwei Li, Chengyao Wang, and Jiaya Jia. LLaMA-VID: An image is worth 2 tokens in large language models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 323–340, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in Neural Information Processing Systems*, 36:34892–34916, 2023.

Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. InstructNav: Zero-shot system for generic instruction navigation in unexplored environment. *arXiv preprint arXiv:2406.04882*, 2024.

Abhinav Rajvanshi, Karan Sikka, Xiao Lin, Bhoram Lee, Han-Pang Chiu, and Alvaro Velasquez. SayNav: Grounding large language models for dynamic planning to navigation in new environments. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pp. 464–474, 2024.

Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. PIRLNav: Pretraining with imitation and RL finetuning for objectnav. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17896–17906. IEEE, 2023.

Botao Ren, Junjun Hu, Xinda Xue, Minghua Luo, Jintao Chen, Haochen Bai, Liangliang You, and Mu Xu. AstraNav-Memory: Contexts compression for long memory. *arXiv preprint arXiv:2512.21627*, 2025.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Brian Ichter, and Sergey Levine. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Conference on Robot Learning*, pp. 2683–2699. PMLR, 2023.

Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. EVA-CLIP: Improved training techniques for CLIP at scale. *arXiv preprint arXiv:2303.15389*, 2023.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, et al. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv preprint arXiv:2502.14786*, 2025.

Saim Wani, Shivansh Patel, Unnat Jain, Angel Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. *Advances in Neural Information Processing Systems*, 33:9700–9712, 2020.

Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, Wenzhe Cai, Hanqing Wang, Yilun Chen, et al. Streamvln: Streaming vision-and-language navigation via slowfast context modeling. *arXiv preprint arXiv:2507.05240*, 2025.

Meng Wei, Chenyang Wan, Peng Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, et al. Ground slow, move fast: A dual-system foundation model for generalizable vision-language navigation. In *International Conference on Learning Representations*, volume 2026, pp. 12380–12396, 2026.

Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

Karmesh Yadav, Jacob Krantz, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Jimmy Yang, Austin Wang, John Turner, Aaron Gokaslan, Vincent-Pierre Berges, Roozbeh Mootaghi, Oleksandr Maksymets, Angel X Chang, Manolis Savva, Alexander Clegg, Devendra Singh Chaplot, and Dhruv Batra. Habitat challenge 2023. `https://aihabitat.org/challenge/2023/`, 2023a.

Karmesh Yadav, Arjun Majumdar, Ram Ramrakhya, Naoki Yokoyama, Alexei Baevski, Zsolt Kira, Oleksandr Maksymets, and Dhruv Batra. OVRL-v2: A simple state-of-art baseline for ImageNav and ObjectNav. *arXiv preprint arXiv:2303.07798*, 2023b.

Yuncong Yang, Han Yang, Jiachen Zhou, Peihao Chen, Hongxin Zhang, Yilun Du, and Chuang Gan. 3D-Mem: 3D scene memory for embodied exploration and reasoning. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17294–17303. IEEE, 2025a.

Zebin Yang, Sunjian Zheng, Tong Xie, Tianshi Xu, Bo Yu, Fan Wang, Jie Tang, Shaoshan Liu, and Meng Li. EfficientNav: Towards on-device object-goal navigation with navigation map caching and retrieval. *Advances in Neural Information Processing Systems*, 38:4286–4312, 2025b.

Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. VLFM: Vision-language frontier maps for zero-shot semantic navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 42–48. IEEE, 2024a.

Naoki Yokoyama, Ram Ramrakhya, Abhishek Das, Dhruv Batra, and Sehoon Ha. HM3D-OVON: A dataset and benchmark for open-vocabulary object goal navigation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5543–5550. IEEE, 2024b.

Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-NaVid: A video-based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*, 2024a.

Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. NaVid: Video-based vlm plans the next step for vision-and-language navigation. In *Robotics: Science and Systems (RSS)*, 2024b.

Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, et al. Embodied navigation foundation model. In *International Conference on Learning Representations*, volume 2026, pp. 127293–127322, 2026a.

Jiazhao Zhang, Gengze Zhou, Hale Yin, Yiyang Huang, Zixing Lei, Qihang Peng, Haoqi Yuan, Jie Zhang, Xudong Guo, Xiaoyue Chen, et al. Qwen-RobotNav technical report: A scalable navigation model designed for an agentic navigation system. *arXiv preprint arXiv:2606.18112*, 2026b.

Ziyu Zhu, Xilin Wang, Yixuan Li, Zhuofan Zhang, Xiaojian Ma, Yixin Chen, Baoxiong Jia, Wei Liang, Qian Yu, Zhidong Deng, et al. Move to understand a 3D scene: Bridging visual grounding and exploration for efficient and versatile embodied navigation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8120–8132. IEEE, 2025.

A Memory Retrieval Algorithm

Algorithm 1 describes the complete memory block assembly procedure that LociNav executes at each navigation step. If the store is empty (first episode of a tour), the memory block is omitted and the model receives only the episode context. Otherwise, the candidate table is built once at the start of each episode and remains frozen thereafter. The candidate and path frames persist until a stall is detected or the episode ends, while the live frame is recomputed whenever the agent crosses a landmark boundary. The resulting memory block is ordered so that the stable frames (candidate and path) occupy the prefix, maximizing KV cache reuse.

Algorithm 1: Memory Block Assembly (per navigation step)

Input: Store $\mathcal{S}$, goal category g, walk graph G, agent landmark ℓ_{now}
Output: Memory block $\mathcal{B}$
if *$\mathcal{S}$ contains no prior-episode frames* **then**
  return $\varnothing$; `// first episode`
if *new episode* **then**
  Score all prior-episode frames in $\mathcal{S}$ against g;
  Build candidate table ranked by $\tilde{s}_\ell/(d_{\text{graph}} + d_0)$; `// Eq. 1`
  Select first candidate ℓ^* from table;
  Compute shortest path $P \leftarrow G.\text{path}(\ell_{\text{now}}, \ell^*)$;
  Sample n_p path frames along P;
  Live frame $\leftarrow$ frame at next hop along P;
if *candidate $\ell^* = \varnothing$ **or** stall detected* **then**
  Select next unmasked candidate ℓ^* from table;
  Recompute $P \leftarrow G.\text{path}(\ell_{\text{now}}, \ell^*)$;
  Sample n_p path frames along P;
if *agent crossed landmark boundary* **then**
  Live frame $\leftarrow$ frame at next hop of $G.\text{path}(\ell_{\text{now}}, \ell^*)$;
$\mathcal{B} \leftarrow [\text{candidate},\ \text{path}_1, \ldots, \text{path}_{n_p},\ \text{live}_1, \ldots, \text{live}_{n_l}]$;
Prepend label tokens; apply pose bias;
return $\mathcal{B}$

B Dataset Construction Details

Tour structure. Tours are constructed by organizing episodes from the same scene into nested prefixes (rungs). A master tour of depth D yields rungs at depths $k \in \{1, 2, 4, 8, 16, D\}$, where rung k consists of the first k episodes of the master. Evaluation runs only the top rung of each master tour to avoid over-counting episodes at shallow depths, where the memory store is least populated.

Connecting segments. Between consecutive episodes of a tour, an oracle phase driven by the simulator's shortest-path follower (Savva et al., 2019) walks the agent to the next episode's starting position and turns to match its starting heading. The observations from these connecting legs are recorded in the persistent store but are excluded from training supervision.

Training corpus construction. Training tours are constructed identically to evaluation tours, using the HM3D ObjectNav and OVON training splits (145 scenes). GOAT-Bench training episodes are used directly without stitching, as they are natively iterative. A three-state search expert generates training trajectories for all three sources (Sec. 4). Episodes in which the expert fails to reach the goal viewpoint are excluded before tour construction, removing 7.3% of OVON episodes and 5.1% of ObjectNav episodes. Tours are then built from the remaining successful episodes. The final training set contains 6,587 OVON, 2,845 ObjectNav, and 2,310 GOAT episodes, totaling 699,772 frames. Training samples are drawn at natural proportions (mixture weights equal to each venue's frame count).

Expert trajectories. For all three sources, training trajectories are generated by a search expert that explores frontiers when no goal-relevant memory is available, follows stored observations when memory provides a candidate, and navigates directly to the goal when it becomes visible. Following MTU3D (Zhu et al., 2025), navigation data is composed of four discrete actions: `MOVE_FORWARD`, `TURN_LEFT`, `TURN_RIGHT`, and `STOP`. The resulting trajectories contain exploration, looking around, and recovery from incorrect candidates, capturing behaviors that a shortest-path oracle cannot exhibit.

Expert policy. Training trajectories are generated by a three-state search expert rather than a shortest-path oracle. The expert operates in one of three states at each step:

- **Visible:** A goal instance is detected in the current view (in frustum, unoccluded, within range). The expert follows the shortest path to that instance's nearest viewpoint and stops.

- **Memory:** The goal is not currently visible, but the memory store contains a candidate. The expert navigates to the candidate's landmark. If it arrives without seeing the goal, it looks around, retires the candidate, and tries the next one.
- **Frontier:** No candidates remain. The expert walks to the nearest unexplored frontier, looks around, and repeats.

The simulator's shortest-path follower (Savva et al., 2019) executes each step, but the goal position is never taken from privileged information: it comes only from what is currently visible, what the store remembers, or the navmesh structure.

Evaluation statistics. Table 7 summarizes the benchmarks used for evaluation. ObjectNav-Tour and OVON-Tour are constructed from standard HM3D evaluation splits, with tours of depth up to 28 episodes. GOAT-Bench episodes are natively iterative, with 5 to 10 sub-goals per episode. All ObjectNav-Tour and OVON-Tour benchmarks share the same 36 HM3D v2 validation scenes, and each official episode appears exactly once.

Table 7: Evaluation benchmark statistics.

Benchmark	Tours	Episodes	Scenes	Depth
ObjectNav-Tour val	36	1,000	36	1–28 sub-goals
OVON-Tour val_seen	122	3,000	36	1–28 sub-goals
OVON-Tour val_seen_syn.	124	3,000	36	1–28 sub-goals
OVON-Tour val_unseen	125	3,000	36	1–28 sub-goals
GOAT-Bench val_seen	360	—	36	5–10 sub-goals
GOAT-Bench val_seen_syn.	340	—	34	5–10 sub-goals
GOAT-Bench val_unseen	360	—	36	5–10 sub-goals

C Training Details

Visibility tag (`see`). The `see` tag is a four-way label assigned to each training frame based on the agent's distance to the nearest viewpoint of the current goal category, using the same `distance_to_goal` criterion as the evaluation metric:

- `here`: the goal is visible in the current view and the geodesic distance to its nearest viewpoint is ≤ 1.0 m.
- `near`: the goal is visible and the distance is between 1.0 m and 3.0 m.
- `far`: the goal is visible but the distance exceeds 3.0 m.
- `no`: the goal is not visible in the current view (regardless of distance).

By aligning the `see` labels with the evaluation criterion, learning to predict `here` is equivalent to learning when the agent has reached the goal.

Training hyperparameters. The vision encoder (EVA-CLIP (Sun et al., 2023)) is frozen throughout training, while the full Vicuna-7B (Chiang et al., 2023) language model is fine-tuned. Approximately 13K new parameters are added for the type embedding (12,288) and per-head attention gates (1,024). We train for a single epoch on approximately 700K frames (ObjectNav + OVON + GOAT), mixed at natural frame-count proportions. The learning rate is 1×10^{-5} with a cosine schedule and 3% warmup, weight decay is 0, the effective batch size is 128, and the maximum sequence length is 2048 tokens. We use FSDP full sharding because ZeRO-2 exceeds GPU memory at all tested micro-batch sizes. The frozen vision encoder's forward pass is chunked along the frame axis (chunk size 128) to limit peak memory, verified to produce bit-identical outputs to the unchunked version. Training takes approximately 200 GPU hours on 4 NVIDIA RTX 6000 Ada GPUs (48 GB each).

D Design Hyperparameters

Landmark and sector parameters. Landmarks are instantiated online as the agent moves: each incoming frame is assigned to the nearest existing landmark within a horizontal distance of $D = 1.0$ m and a vertical distance of 1.0 m. If no landmark qualifies, a new one is created at the frame's position.

Landmark centers are fixed once created. Each landmark partitions viewing directions into $K = 4$ angular sectors, and each sector retains the single observation closest to the sector center. A later observation replaces the incumbent only if it is strictly closer to the sector center.

Sectors per landmark (K). We measure the fraction of the panoramic coverage retained by K stored frames per landmark across 1,000 evaluation episodes (2,372 landmarks). As shown in Fig. 5, the marginal gain drops sharply at $K = 4$: from $K = 3$ to $K = 4$ the gain is $+14.2$ percentage points, while from $K = 4$ to $K = 5$ it falls to $+3.6$ pp. This knee coincides with a geometric explanation: $K = 4$ yields 90-degree sectors, which nearly match the camera's horizontal field of view (79 degrees), so one frame approximately fills its sector.

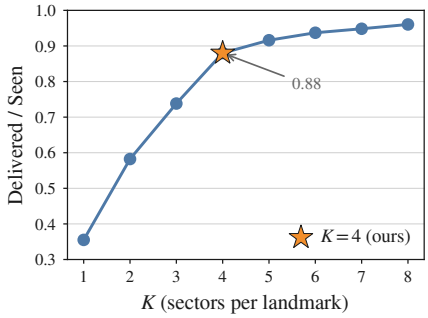


Figure 5: **Panoramic coverage retained by K stored frames per landmark.** The marginal gain drops sharply at $K = 4$, which matches the camera's 79-degree horizontal field of view.

Path frames (n_p) and memory frame tokens (n_v). Fig. 6 sweeps n_p (the number of path frames) and n_v (the number of visual tokens per memory frame), plotting SR against inference frequency to visualize the accuracy-efficiency trade-off. Increasing n_p from 0 to 2 yields a large SR gain (+8.2) with only a modest frequency drop (4.44 to 4.03 Hz), while the gain from 2 to 3 is marginal (+0.4). We therefore set $n_p = 2$. For n_v, a single token ($n_v = 1$) loses too much spatial detail (50.5 SR), while $n_v = 16$ and 64 improve SR by only 1.8 and 3.4 over $n_v = 4$ but reduce frequency to 2.67 and 1.43 Hz. We choose $n_v = 4$, which retains the 2×2 spatial structure of the pooled feature while keeping inference above 4 Hz.

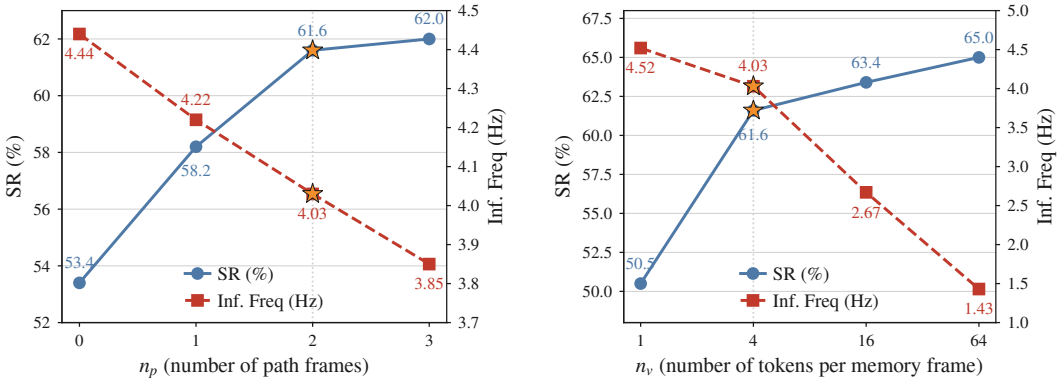


Figure 6: **Accuracy-efficiency trade-off for (a) the number of path frames n_p and (b) the number of visual tokens per memory frame n_v.** Stars mark the chosen operating points ($n_p = 2$, $n_v = 4$).

Spatial pose bias. The pose bias (Eq. 2) uses $M = 20$ cosine frequency terms distributed across three spatial channels. The yaw channel uses 4 frequencies constrained to integers $s \in \{1, 2, 3, 4\}$, which ensures that the bias is exactly periodic across the $\pm\pi$ boundary. The resolution is set by the store's angular sector width (90 degrees at $K = 4$). The horizontal x and y channels each use 8 frequencies arranged in a geometric ladder from $\pi/S_{\max}$ to π/r, where $S_{\max}$ is the maximum scene diameter and r is the store's spatial cell radius, covering spatial relationships from room scale down to the landmark resolution. Each of the backbone's 32 attention heads receives a scalar gate g_h initialized at $g_{\text{init}} \approx 1.0 \times \sigma_{\text{logit}}$, where σ_{logit} is the measured standard deviation of the backbone's

Table 8: **Inference latency breakdown by component.**

ViT Cache	KV Reuse	ViT Encoding (ms)	KV Prefilling (ms)	Decoding (ms)	Total (ms)
✗	✗	148	149	110	407
✗	✓	145	102	109	356
✓	✗	**42**	134	**108**	284
✓	✓	**42**	**97**	109	**248**

attention logits. This places the initial bias magnitude at a level that is perceptible but not dominant, allowing training to amplify or suppress spatial awareness per head.

Distance constant (d_0). The candidate scoring formula (Eq. 1) divides the normalized score by $d_{\text{graph}} + d_0$. With $d_0 = 64$ m and a median graph distance of approximately 13 m in our training scenes, the denominator varies by at most $1.6\times$ across candidates, producing a mild bias toward nearby candidates without collapsing into a nearest-first rule. This value was selected to balance relevance against reachability based on a grid search over $d_0 \in \{0.25, 1, 4, 16, 64, \infty\}$.

Evaluation metrics. Success Rate (SR) is the fraction of episodes in which the agent calls `STOP` within 1.0 m geodesic distance of the nearest goal viewpoint. Success weighted by Path Length (SPL) (Anderson et al., 2018) additionally penalizes inefficient paths: $\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{L_i^*}{\max(L_i, L_i^*)}$, where $S_i \in \{0, 1\}$ is the success indicator, L_i^* is the shortest-path length, and L_i is the agent's actual path length.

E Additional Evaluation Results

E.1 Performance across Tour Depths

Fig. 7 extends the tour-depth analysis of Sec. 5.2 to the remaining three evaluation splits. The trends are consistent with the OVON-Tour Unseen results in Fig. 3. On both OVON-Tour Seen and Synonyms, LociNav shows a steep SR increase with tour depth (41.5 to 72.6 and 44.5 to 68.0), while Uni-NaVid (Zhang et al., 2024a) remains flat and Uni-NaVid-Mem gradually degrades. SPL follows the same pattern, rising from around 21 to above 35 for LociNav while declining for Uni-NaVid-Mem. On ObjectNav-Tour, where the closed category set makes goals easier to locate, all three methods start from a higher baseline (SR above 72). LociNav still improves with depth (73.3 to 79.7), though the margin is smaller than on OVON-Tour, consistent with the reduced difficulty. Uni-NaVid-Mem degrades more steeply on ObjectNav-Tour (72.5 to 63.0) than on the OVON splits, suggesting that the token compression bottleneck is more harmful when the baseline performance is high and small accuracy losses are more visible.

E.2 Ablation Study on Inference Latency Breakdown

Tab. 8 breaks down the per-step inference latency of Tab. 5 into three phases: ViT encoding, KV prefilling, and decoding. Without either reuse mechanism, ViT encoding and KV prefilling each take roughly 150 ms, dominating the total latency. The ViT cache eliminates redundant visual encoding of memory frames, reducing ViT encoding from 148 to 42 ms. KV prefix reuse avoids re-prefilling the stable memory tokens, cutting KV prefilling from 134 to 97 ms (with ViT cache enabled). Decoding time remains approximately constant ($\sim$109 ms) across all configurations, confirming that the reuse mechanisms affect only the input processing phases without impacting action generation.

E.3 Visual Cache Approaches

Tab. 9 compares different approaches for caching visual observations across the tour. Storing raw images for all observed frames consumes 3.25 GB per tour and requires re-encoding at each step, resulting in high RtL (1.47 s). Applying the landmark-and-sector organization to raw images reduces storage to 0.13 GB and RtL to 0.85 s by limiting the number of retained frames, while achieving the highest SR (65.0). Caching the full ViT features (n_v=64) matches this SR at 0.70 s RtL but requires 1.85 GB, as the feature tensors are larger than compressed images. Spatially pooling the cached features to n_v=16 and n_v=4 tokens progressively reduces both storage and latency, with n_v=4

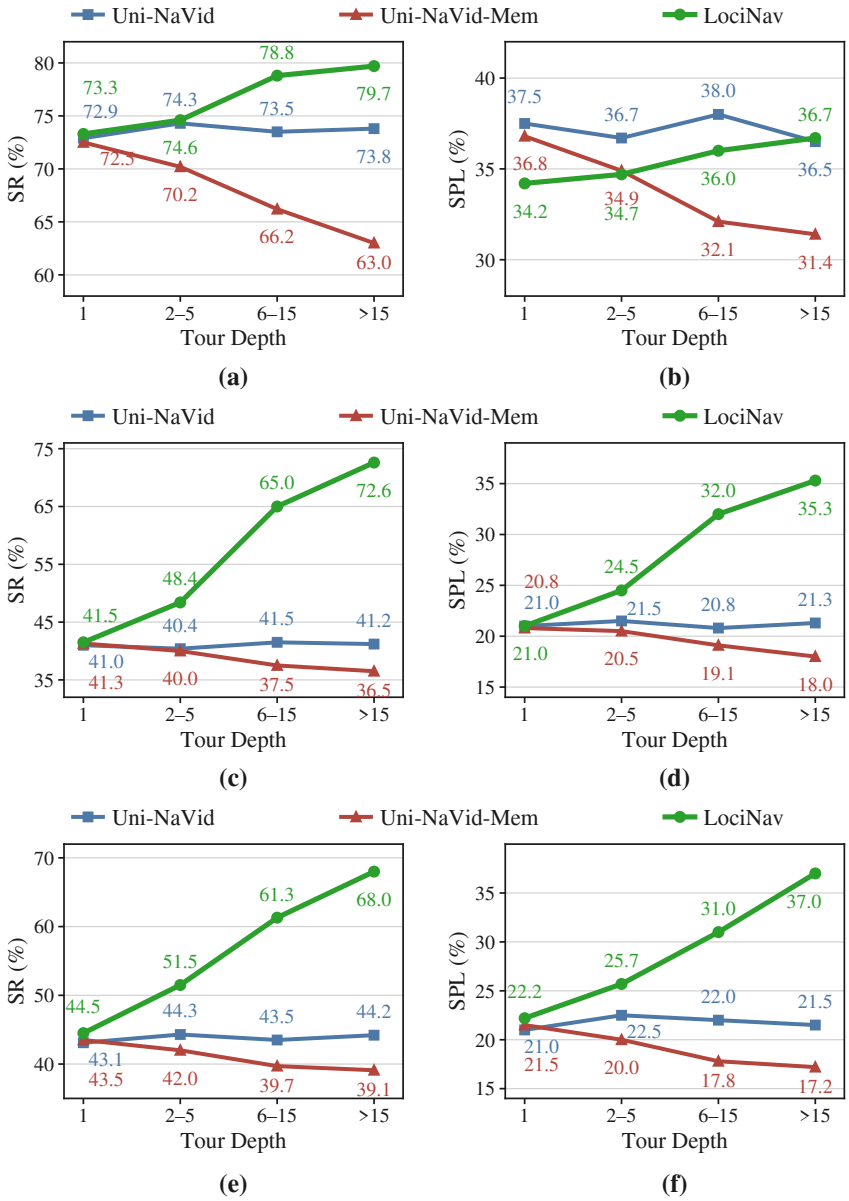


Figure 7: **SR and SPL comparison at different tour depths on ObjectNav-Tour (a, b), OVON-Tour Seen (c, d), and OVON-Tour Synonyms (e, f).**

Table 9: **Comparison of different visual cache approaches.**

Type	Storage (GB)	SR	RtL (s)
Raw image (all)	3.25	63.7	1.47
Raw image (landmarked & sectored)	**0.13**	**65.0**	0.85
ViT Cache (full, n_v=64)	1.85	**65.0**	0.70
ViT Cache (n_v=16)	0.47	63.4	0.37
ViT Cache (n_v=4)	0.15	61.6	**0.25**

reaching 0.15 GB and 0.25 s RtL at a modest SR trade-off (61.6). LociNav uses this configuration, as it provides the best balance of storage, latency, and accuracy.