ATTENTION IS THE MESSAGE: COMMUNICATING THROUGH PLANNER ATTENTION FOR MULTI-ROBOT EMBODIED PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Robots that share a semantic map still spend computation encoding the same regions in their onboard planners. Can they also share the planner states computed from those regions, even as the map changes and each robot pursues a different query? We introduce region-addressed attention messages, which let robots with frozen planners reuse one another's computation without knowing the receiver's query or training for communication. Each room has an owner that encodes it using only a shared prefix and that room's evidence. The resulting attention state is tagged with the room identity, evidence version, and owner. Receivers accept a message only when these tags match the current map, then use it directly through the planner's native attention. This construction ensures that every accepted message equals the state the receiver would compute itself. With two robots, attention messages reduce planner GPU time by 45% compared with cached re-perception. Across teams of two to four robots, they match the computational cost of a centralized shared cache while keeping planning onboard. Experiments on HM3D and MP3D show that sharing room information improves navigation success, with re-perception recovering most of the same gains. Our contribution is to achieve these benefits at lower computational cost. We further validate the system through real-world navigation experiments with two physical robots, demonstrating shared planner computation with onboard planning.

1 INTRODUCTION

A shared semantic map gives a robot team a common account of the world, but it does not preserve the computation performed over that account. In multi-robot navigation built around a shared map, synchronized observations are repeatedly converted into planner state before each robot chooses where to move next (Liu et al., 2022; Shen et al., 2025). This duplication remains even when the robots run the same frozen vision–language planner: a room one planner has already processed is constructed again by the next planner that needs it. We study this reuse problem after map synchronization, when evidence is already shared but the planner state derived from it is not.

The duplication arises because a planner call couples two quantities that change for different reasons. Regional construction should change when the evidence within a region changes, whereas decision reasoning should change with the current query. Joint encoding ties both to one sequence. A later region's construction depends on the regions before it and their prompt layout, so each regional state belongs to the combination in which it was built. Text reports and map crops transfer evidence, but the receiver must still rebuild planner state. Independent context construction, positional adaptation, and KV transport are established transformer capabilities (Gim et al., 2024; Ma et al., 2025; Yang et al., 2025; Ye et al., 2025); what remains unresolved is a regional planner state whose construction follows synchronized world evidence and can finish before the receiver's query is known. We therefore separate construction driven by world evidence from reading driven by the current query.

We introduce region-addressed attention messages (Figure 1), which factor planner computation at the boundary between shared world state and the current decision. A designated owner materializes regional planner state from a compatible prefix and the synchronized evidence of one region alone, excluding the current query. At decision time, a robot's task tokens read the records consistent with

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.

the current map through the frozen planner's native attention. This split makes completed regional construction reusable across decisions and transferable between compatible planners, while task reasoning stays with the receiver; region version and owner identity decide when that state is current.

This factorization also fixes what an attention message computes. Every regional branch is constructed independently from the same prefix, and the branches meet only when the receiver processes its current task. With one admitted region, this canonical fork–join computation equals ordinary joint encoding. With several, it removes the causal paths through which an earlier region would alter the construction of a later one; positional relocation places each stored key in the receiver's layout but cannot restore the hidden context removed with those paths (Proposition 1). We therefore treat multi-region composition as an approximation to joint encoding and measure the gap.

Controlled planner studies and closed-loop navigation show that this state remains useful as the map changes and delivers what re-perception rebuilds with 45% less planner GPU time at two robots.

2 Related Work

Map-mediated goal-oriented navigation. Modular navigation represents observations in top-down semantic or obstacle maps. PLMD completes unobserved label-map regions (Shen et al., 2026); multi-agent semantic navigation communicates compressed map information (Liu et al., 2022); MCoCoNav uses a global semantic map as its communication bridge (Shen et al., 2025); and RoCo coordinates robots through language-model dialogue (Mandi et al., 2024). These works build, complete, or reason over the shared map. We keep the map and ask what planner state to exchange once it is synchronized, comparing attention messages with text and map crops.

Communication between agents. Cooperative agents learn continuous messages, communication graphs, and intermediate feature fusion (Lo et al., 2024; Hu et al., 2024; Ding et al., 2024; Hu et al., 2022; Yang et al., 2023); these messages require communication training. Language-model agents now exchange transformer state directly. KVCOMM reuses overlapping content across agent-specific prefixes by correcting cache offsets from an online anchor pool (Ye et al., 2025); KVComm transmits selected KV layers (Shi et al., 2026); Cache-to-Cache learns projections and gated fusion between heterogeneous caches (Fu et al., 2026); and LatentMAS shares hidden states through a latent working memory (Zou et al., 2025). These messages are bound to a conversation context or a learned translator and have no notion of staleness. Ours are constructed before any query, addressed to a region of a changing world, stamped with a version and owner that let receivers reject stale state, and read by frozen planners without a translator or fusion module.

Independent context construction and cache reuse. Prompt Cache, Parallel Context Windows, and Block-Attention construct context blocks independently and reuse them at stable or re-encoded positions (Gim et al., 2024; Ratner et al., 2023; Ma et al., 2025). APE aligns parallel with sequential encoding (Yang et al., 2025), EPIC formalizes position-independent caching (Hu et al., 2025), and CacheBlend and MPIC recompute selected cross-block interactions (Yao et al., 2025; Zhao et al., 2025). These methods serve one reader over a fixed context. Viewed as a system, our protocol is a distributed prefix cache keyed by map regions; what it adds is the key's semantics: which state may be committed before the reader is known, about a world that changes, and when a reader must reject it. We use their rephasing as a component.

Distributed attention and execution placement. Serving systems manage, transport, and place KV caches (Kwon et al., 2023; Liu et al., 2024b; Zhong et al., 2024), and FlashAttention and RingAttention supply the partial-softmax reduction for distributed exact attention (Dao et al., 2022; Liu et al., 2024a). We use this reduction only in remote recall, the reference placement for Property 3.

3 Method

Intuitively, each room is a cached document: its owner, the robot that explored most of it, encodes it once without the goal tokens or other rooms and labels the keys and values with the room, a hash of its rendering and selected patches, and its own identity. A deciding robot reads the records whose labels match the current map as if it had encoded them itself (Algorithm 1).

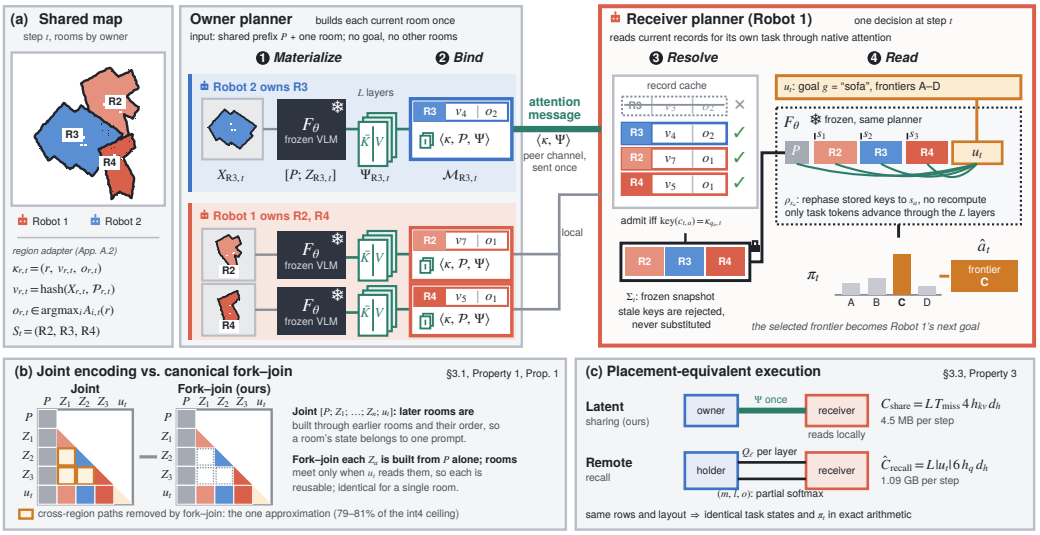


Figure 1: **Region-addressed attention messages.** (a) An owner materializes and binds each current regional record; the receiver resolves a valid snapshot and reads it through native attention. (b) Joint encoding is contrasted with the canonical fork–join construction. (c) Latent sharing and remote recall are placement-equivalent executions of the same fixed-snapshot read.

Algorithm 1 One decision of robot i at step t.

Input: synchronized map M_t, active rooms S_t, task tokens u_t, record cache $\mathcal{C}_i$
for each room $r \in S_t$ **do**
  render $X_{r,t}$ and patch set $\mathcal{P}_{r,t}$; key $\kappa_{r,t} \leftarrow (r,\ v_{r,t},\ o_{r,t})$ with $v_{r,t} = \mathrm{hash}(X_{r,t}, \mathcal{P}_{r,t})$
  if no record in $\mathcal{C}_i$ has key $\kappa_{r,t}$ **then**
    if $o_{r,t} = i$: $\Psi \leftarrow \mathrm{Materialize}_\theta(P, X_{r,t}; \mathcal{P}_{r,t})$ ▷ prefix and this room only
    else: receive $\langle \kappa_{r,t}, \Psi \rangle$ from owner $o_{r,t}$ ▷ attention message
    store $\mathrm{Bind}(\kappa_{r,t}, \mathcal{P}_{r,t}, \Psi)$ in $\mathcal{C}_i$
$\Sigma_t \leftarrow \mathrm{Resolve}_t(S_t, \boldsymbol{\kappa}_t)$ ▷ exact key matches, in S_t order
$\pi_t \leftarrow \mathrm{Read}_\theta(P, \Sigma_t, u_t)$ ▷ rephase keys; task tokens attend
return frontier $\hat{a}_t$ from π_t; evict received records unused for eight decisions

Formally, at decision step t, N robots observe a common top-down semantic map M_t, and a frozen vision–language planner F_θ selects a frontier from $\mathcal{A}_t$ for target g.

The navigation front end fixes the synchronized map, ordered selected-region sequence S_t, regional inputs, and task tokens before the planner-state layer runs. Let P be the shared fixed prefix, $X_{r,t}$ the current rendering of tracked region r, $\mathcal{P}_{r,t}$ its retained patch set (the front end's patch selection restricted to r; Appendix A.2), u_t the receiver's task tokens, and $\boldsymbol{\kappa}_t$ the expected record keys. The factorization has two computational stages, Materialize and Read. Bind and Resolve provide the state identity and admission used across decisions:

$$\begin{aligned} \Psi_{r,t} &= \mathrm{Materialize}_\theta(P, X_{r,t}; \mathcal{P}_{r,t}), & \mathcal{M}_{r,t} &= \mathrm{Bind}(\kappa_{r,t}, \mathcal{P}_{r,t}, \Psi_{r,t}), \\ \Sigma_t &= \mathrm{Resolve}_t(S_t, \boldsymbol{\kappa}_t), & \pi_t &= \mathrm{Read}_\theta(P, \Sigma_t, u_t). \end{aligned} \tag{1}$$

Materialize never attends to the target or task tokens; Read supplies the task tokens and returns π_t, from which the fixed decision rule selects $\hat{a}_t$. Figure 1 summarizes the flow and Appendix A.1 gives the complete interface; all properties are pointwise in one realized decision.

3.1 Regional construction and current task reading

Materialize evaluates $[P; Z_{r,t}]$ with the frozen transformer and stores the regional pre-RoPE keys and values; the target, task tokens, and other regions are absent from this pass, so the payload is a function of the prefix and the regional input $(X_{r,t}, \mathcal{P}_{r,t})$ alone, both of which the version digests.

Read places the admitted branches in the supplied order $A_t = (r_1, \ldots, r_n)$ and applies RoPE at their resolved positions; this receiver-side rephasing ρ_{s_a}, with branch start $s_a = L_0 + \sum_{b<a} T_{r_b,t}$, changes only the rotary phase (Appendix A.3).

This construction defines a canonical fork and join computation: the shared prefix forks into regional branches that are constructed independently, and the branches join only when the current task tokens read them. At each layer the receiver advances only the task tokens, whose queries attend to the prefix, every admitted regional row, and earlier task tokens; the regional rows stay fixed, and locally materialized and received records use the same attention interface (Equations 13 and 14, Appendix A.4).

With one admitted region, fork and join coincides with joint encoding. With several, joint encoding gives later regions construction paths through earlier ones, which rephasing cannot recreate.

Property 1 (Message equals canonical construction) *Under the compatibility conditions in Appendix A.5, Materialize followed by Read realizes the canonical fork–join graph exactly.*

Why build regions from the prefix alone. A message is sent before the receiver's context, its set and order of regions, is known.

Proposition 1 (Receiver-exact messages) *Under the compatibility conditions of Property 1, the canonical branch equals the joint-encoding rows of region r, up to rotary phase, in every receiver context that reads r first. Hence any message that is exact in every receiver context equals the canonical branch, and such a message exists only if earlier regions never change the rows of r.*

The argument is short (Appendix A.5); its role is to show that the cross-region error cannot be removed by a different choice of receiver-independent message, only by receiver-side recomputation. Section 4.4 measures it.

3.2 Current state binding and selective reuse

A message must identify the world state for which it is valid. Bind associates the payload with an episode-local record key $\kappa_{r,t} = (r, v_{r,t}, o_{r,t})$:

$$\begin{aligned} \mathcal{M}_{r,t} &= \langle \kappa_{r,t}, \mathcal{P}_{r,t}, \Psi_{r,t} \rangle \,, \quad \kappa_{r,t} = (r, v_{r,t}, o_{r,t}), \\ \Psi_{r,t} &= \left\{ (\bar{K}_r^\ell, V_r^\ell) \right\}_{\ell=1}^{L} . \end{aligned} \tag{2}$$

The payload holds every layer's regional pre-RoPE keys and values; r addresses the region, $v_{r,t}$ identifies its rendering and retained patches, and $o_{r,t}$ names the sole writer. In exact arithmetic the version alone would fix the payload; the owner matters because int4 kernels are not bitwise deterministic, so one designated writer per version makes every receiver read the same bytes, and it tells a receiver whom to ask for a missing record. An attention message carries $\langle \kappa_{r,t}, \Psi_{r,t} \rangle$. Any change to region, version, or owner invalidates the record; Resolve admits only complete-key matches, never substitutes stale state, and fixes the snapshot Σ_t for one planner forward. Appendices A and A.1 specify the lifecycle and edge cases.

Property 2 (Received equals fresh) *Replacing freshly materialized current records by resident or received records with the same complete keys leaves the decision unchanged in exact arithmetic.*

The assumptions and proof are in Appendix A.5.

3.3 Execution placement of the read operation

The factorization does not prescribe where Read executes. In *latent sharing*, owners send each missing attention payload once and the receiver evaluates Equation 14 locally. In *remote recall*, the receiver sends post-RoPE task queries at every layer; each holder attends over its assigned rows and returns a stable partial-softmax summary. These are two execution placements of the same fixed-snapshot computation.

Property 3 (Placement invariance) *For one fixed snapshot, centralized Read, remote recall, and latent sharing produce the same decision in exact arithmetic.*

Appendix A.7 states the complete partition and determinism conditions and gives the stable partial-softmax proof, multi-holder extension, and finite-precision boundary.

The two placements also answer when agents should exchange state rather than queries. Sending a payload once is cheaper than per-layer query traffic whenever the missing regional tokens are fewer than $1.5\, g_{\mathrm{GQA}}|u_t|$ with $g_{\mathrm{GQA}} = h_q/h_{kv}$ (Appendix A.7), which holds with a wide margin in our setting. We use remote recall only as a reference placement that tests this property, not as a competing baseline; its traffic model (Equation 21) and measurements are in Appendices A.7 and B.

We impose these properties as design constraints on the protocol. They follow from causal suffix invariance, deterministic evaluation of hash-identical inputs, and the standard partial-softmax merge (Dao et al., 2022), and they fix the exact reference the protocol preserves. With non-deterministic int4 kernels they define the reference computation; the one approximation is independent regional construction, and Section 4.4 measures both.

4 Experiments

4.1 Experimental setup

End-to-end and controlled studies run two robots, each with an int4 GLM-Edge-V-5B planner (Zhipu AI (Z.ai), 2024) ($L = 40$, $h_q = 24$, $h_{kv} = 6$, $d_h = 128$), for up to 500 synchronized steps per episode; the replay sweep extends to four robots (Table 3). End-to-end results use the full HM3D v0.2 (1,000 episodes) and MP3D (2,195 episodes) validation splits (Yadav et al., 2023; Chang et al., 2017), plus multi-robot IIN on HM3D v0.2. Controlled studies use val-72 (two episodes from each of the 36 HM3D validation scenes; 3-seed means unless noted); the network study uses val-18. The latent-sharing configuration scores 0.708/0.372 SR/SPL on val-72 (Table 2) and 0.818/0.463 on the 1,000-episode HM3D split (Table 1); the episode samples differ. Following MCoCoNav (Shen et al., 2025), episode SPL takes the better of the two robots. Joint agreement is the fraction of decisions on which an interface picks the same frontier as joint encoding, a single pass over the full prompt (Section 3.1). Payload (planner-channel bytes after map synchronization) and server-s (planner-server wall time summed over robots) are per step, which ends when every robot has decided; latencies are per decision.

Planner input. A *pruned* planner reads only the visual tokens of the active rooms instead of all $24 \times 24 = 576$ map tokens. The *active-room rule* marks a room active if it contains a robot or a frontier, is four-neighbor adjacent to such a room, changed version since the previous decision, or contains the previous frontier. The ***WallPPR*** patch selector scores map tokens with a personalized-PageRank random walk on the free-space lattice, seeded at recent map changes, robots, frontiers, and the previous goal and damped at walls, and keeps the highest-scoring tokens within the budget. Restricted to a room, the kept tokens form its patch set $\mathcal{P}_{r,t}$ (Equation 8); because $\mathcal{P}_{r,t}$ enters the version, a record is admitted only when sender and receiver select the same patches.

Baselines. End-to-end comparisons use Greedy, Cost-Utility, Random Sampling, Multi-SemExp, Co-NavGPT, MCoCoNav, and PLMD; controlled comparisons use text reports, map crops, local re-perception with and without a version-keyed cache, and latent sharing with the checkpoint, WallPPR selector, retention policy, and task sample fixed. Remote recall serves only as the placement reference (Appendix B).

4.2 Results and analysis

Comparison with baselines. Table 1 compares complete systems on multi-robot ObjNav and Multi-IIN. Because the published baselines use different planners and training methods, these results compare the overall systems but do not isolate the effect of our delivery interface. Our complete system records 0.818/0.463 SR/SPL on HM3D and 0.621/0.392 on MP3D. In the matched internal comparison, the pruned no-sharing control records 0.752/0.394 and 0.563/0.356, so reading teammate regional state adds 6.6 and 5.8 SR points. Most of the margin over MCoCoNav (0.716 on HM3D) comes from the front end (0.752 without sharing) and the shared information (0.802 with re-perception) rather than the protocol. Latent sharing and re-perception are closer (0.818 vs.

Table 1: Multi-robot ObjNav on the full HM3D v0.2 and MP3D validation splits, and Multi-IIN on HM3D v0.2 (MCoCoNav rerun under our protocol). TR: teammate rooms. All methods except Co-NavGPT (centralized) run distributed. Training: × none, RL reinforcement, SSL self-supervised. Published baselines differ in planner and training, so the comparison is system-level.

			ObjNav				Multi-IIN	
			HM3D v0.2		MP3D		HM3D v0.2	
Method	Planner	Train	SR	SPL	SR	SPL	SR	SPL
Greedy (Visser et al., 2013)	None	×	0.611	0.328	0.406	0.225	–	–
Cost-Utility (Julià et al., 2012)	None	×	0.625	0.323	0.419	0.212	–	–
Random Sampling	None	×	0.636	0.336	0.435	0.281	–	–
Multi-SemExp (Chaplot et al., 2020)	None	RL	0.612	0.327	–	–	–	–
Co-NavGPT (Yu et al., 2023)	GPT-3.5 (remote)	×	0.661	0.331	–	–	–	–
MCoCoNav (Shen et al., 2025)	GLM-4V 9B	×	0.716	0.387	0.568	0.334	0.521	0.237
PLMD (Shen et al., 2026)	GLM-4V 9B	SSL	0.762	0.406	0.591	0.382	–	–
Full map, no sharing	2×GLM-Edge-V 5B	×	0.722	0.376	0.552	0.349	–	–
Pruned, no sharing	2×GLM-Edge-V 5B	×	0.752	0.394	0.563	0.356	–	–
Pruned + TR (re-perception)	2×GLM-Edge-V 5B	×	0.802	0.446	0.599	0.391	–	–
Ours: pruned + TR (latent)	2×GLM-Edge-V 5B	×	**0.818**	**0.463**	**0.621**	**0.392**	**0.563**	**0.254**

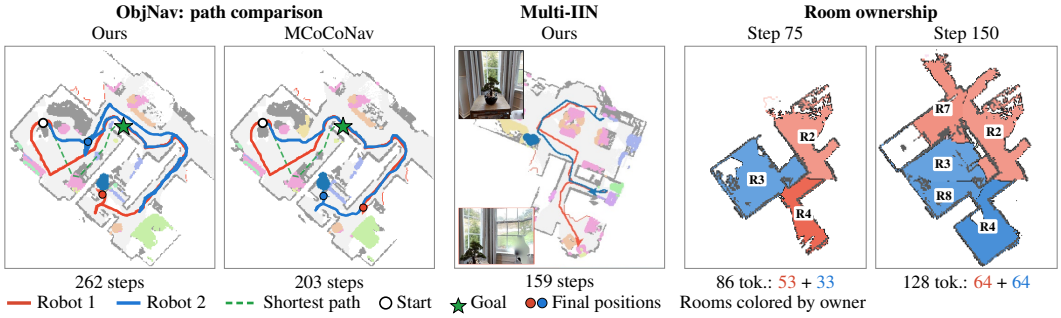


Figure 2: Qualitative results on HM3D v0.2 validation. Left: final trajectories of our system and of MCoCoNav rerun on the same ObjNav episode (target: toilet) and seed, with the geodesic shortest path. Middle: a Multi-IIN episode; the insets show the goal image (top) and robot 1's view at success (bottom). Right: room ownership in an ObjNav episode (target: sofa). Each room is colored by its owner, the robot that explored most of it and builds its attention message; tokens read at each decision are split by owner. Between steps 75 and 150, R4 moves to robot 2 and R7 and R8 appear.

0.802 and 0.621 vs. 0.599), so we attribute the delivery interface's benefit to cost rather than navigation quality (Section 4.3). On Multi-IIN, the complete system records 0.563/0.254. Figure 2 (left, middle) shows an ObjNav episode against MCoCoNav and a Multi-IIN episode.

4.3 Communication Interface

Information delivered. We compare what the deciding planner receives about teammate rooms, holding the model, patch selector, retention policy, and episodes fixed (Table 2). Text reports and map crops send the fewest bytes but match joint encoding's frontier on only 59.8% and 72.1% of decisions, versus 76.1% for attention messages. Joint encoding itself agrees with a rerun on 94.8% of decisions under int4 kernels, so attention messages retain 80% of attainable agreement, map crops 76%, and text reports 63% (Figure 3). Map crops come close only because the receiver re-encodes them (4.7 vs. 2.7 s mean execution latency); attention messages are the fastest interface and match local re-perception (76.1% vs. 75.5%), which rebuilds the same memories (Property 2) but pays the encoding on every robot. The three model-native interfaces are navigation-equivalent: their SRs span 0.667–0.708, within the 7.7-point standard error of a difference over 72 episodes. Int4 kernels

Table 2: Communication interfaces on HM3D val-72. Payload: planner-channel bytes per step. Server-s: summed planner-server time per step, measured for model-native interfaces. Exec.: mean/p90 execution latency. Joint agr.: agreement with joint encoding; joint encoding agrees with its own int4 rerun on 0.948. SR/SPL: 3-seed means.

Interface	Payload	Server-s	Exec. (s)	Joint agr.	SR	SPL
Full map, no sharing	0	—	4.6 / 6.0	—	0.625	0.362
Text report	175 B	—	3.1 / 5.2	0.598	0.611	0.362
Map crop	8.9 KB	—	4.7 / 6.1	0.721	0.625	0.366
Local re-perception	0	3.62	4.4 / 5.9	0.755	0.667	0.369
Local re-perception (cached)	0	3.07	3.3 / 4.2	0.750	0.694	0.354
Ours: latent sharing	4.5 MB	**1.68**	**2.7 / 3.4**	**0.761**	0.708	0.372

Table 3: Robot-count sweep on replayed HM3D val-18 requests, per step. Lat.: mean / p90 latency (s). GPU-s: summed planner-server time. Traffic: planner-channel bytes. Distributed setups use one GPU per robot, so GPU-s is the matched quantity.

	Centralized, shared cache		Distributed, cached re-perc.		**Distributed, latent sharing (ours)**		
N	Lat.	GPU-s	Lat.	GPU-s	Lat.	GPU-s	Traffic
2	1.64 / 2.20	1.64	1.62 / 2.25	3.07	**0.92 / 1.25**	1.68	4.2 MB
3	2.18 / 2.99	2.25	1.97 / 2.74	5.66	**0.90 / 1.23**	2.18	11.3 MB
4	2.67 / 3.66	2.63	2.32 / 3.22	8.92	**0.91 / 1.25**	2.67	20.3 MB

are not bitwise deterministic (Appendix C), so near-tie choices can flip; this explains the 2.7-point gap even between cached and uncached re-perception, which compute identical memories.

Cost. These interfaces differ in cost. Re-perception repeats encoding on every robot. Caching rooms by version cuts its server time only from 3.62 to 3.07 s, because rendered robot arrows and frontier letters change room versions frequently. Latent sharing sends 4.5 MB per step and needs 1.68 s, 45% less, because each changed room is encoded once by its owner. The saving is not an artifact of this rendering: with arrows and letters moved from the room images into the task text, both costs fall and latent sharing still needs 42% less (1.24 vs. 2.14 s; Table 8). It also has the lowest execution latency. Measured traffic matches Equation 21 within 3%, and Figure 2 (right) shows which rooms each robot owns and therefore encodes and sends. The advantage holds on constrained links (Figure 4a): from 100 to 1000 Mbps, latent sharing waits under 0.35 s and decides in 2.9–3.6 s.

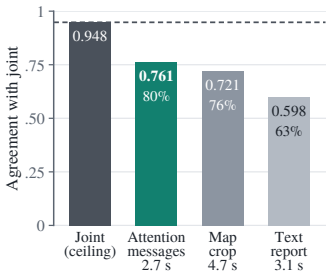


Figure 3: Agreement with joint encoding (% of int4 ceiling) and mean latency per decision.

Robot-count sweep. Table 3 and Figure 4(b,c) replay HM3D val-18 requests with N=2 to 4 robots; a step ends when all N robots have decided. Cached re-perception's GPU time grows from 3.07 to 8.92 s per step, because every robot re-encodes every changed room. Latent sharing encodes each room once, at its owner, and its GPU time matches that of a centralized server with a shared room cache within 4% at every team size (1.68/2.18/2.67 vs. 1.64/2.25/2.63 s), with 4.2 to 20.3 MB of traffic per step. Its step latency stays near 0.9 s while the single-GPU centralized server slows from 1.64 to 2.67 s. This latency gap comes from running N planners on N GPUs, so our claim is equal GPU time with planning kept onboard, not less compute or lower latency than a centralized server with equal GPUs.

Real-world experiments. We also test our method on a physical two-robot platform without real-world fine-tuning: a Unitree Go2 quadruped with a RealSense D435i RGB-D camera and a wheeled base with an Odin1 RGB-D, LiDAR, and IMU module. Both robots perform perception, planning, and control entirely on their onboard compute; the robot network carries only the synchronized map and regional attention state. We evaluate a cluttered classroom and a locker hall with narrow aisles, using a plant as the target for five trials in each environment. As shown in Figure 5, the robots

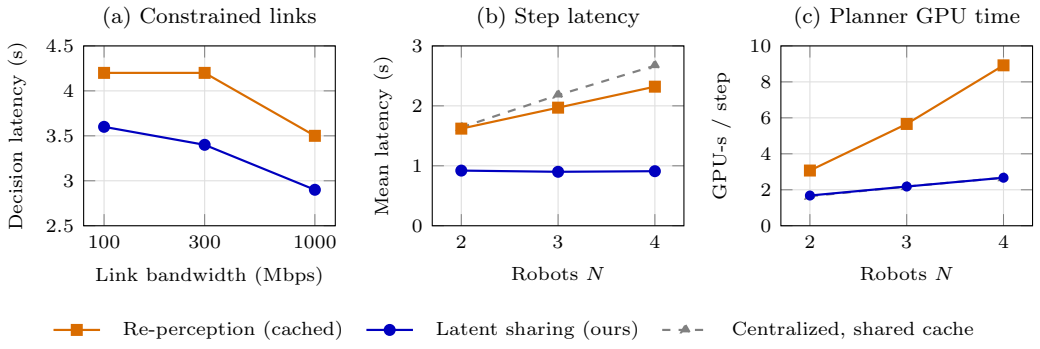


Figure 4: Communication cost at scale. (a) Request-to-frontier latency on HM3D val-18 with a 20-ms round-trip time. Remote recall, the placement reference, is off scale (43.7 s at 100 Mbps; Appendix B). (b,c) Replayed HM3D val-18 requests; a step ends when all N robots have made one decision. Latent sharing matches the centralized server's GPU time in (c) with flat step latency in (b).

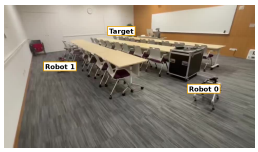
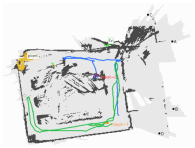
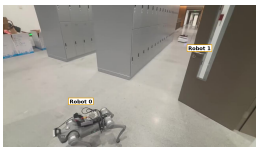
Environment	Scene	Trajectories	L (m)	SR	SPL
Classroom			14	3/5	0.546
Locker hall			11	2/5	0.361

Figure 5: Real-world deployment. Scene: both robots at the start of an episode. Trajectories: robot paths on the shared semantic map in a successful episode. L: shortest path length measured on site. SR/SPL: over five trials.

succeed in 5 of 10 trials with 0.454 SPL; observed failures arise mainly from perception or low-level control. The robots differ in body and sensors but use the same planner configuration, so these trials show cross-embodiment feasibility rather than a comparison with a real-world baseline.

4.4 Ablation Studies

Composition. We vary how room states are built and where they are read (Table 4(a)). *Joint* encodes all selected rooms in one pass, so later rooms see earlier ones; *independent* builds each room alone from the prefix, as our messages do. A *single* planner builds and reads every room itself; a *peer* reader receives teammate rooms as messages. Independent construction preserves 75–77% of joint-encoding choices, 79–81% of the 0.948 int4 ceiling, yet SR moves by at most 1.4 points (0.653–0.667 vs. 0.667), below the 7.9-point standard error of a difference over 72 episodes. In a paired replay of 42 fixed-memory requests, peer and single-planner readers select the same frontier in 41; this small replay suggests little added disagreement rather than none. Independent construction is thus the main measured approximation.

Lifecycle. We stress-test five events (room splits, merges, delayed updates, owner changes, and prompt changes), replaying three recorded requests per event against re-perceiving every room (Table 4(b)). The deciding robot recovers within one decision (for delayed updates, once the update

Table 4: Controlled ablations. (a) HM3D val-72, in runs separate from Table 2. Joint agr.: agreement with joint encoding of the selected rooms. Every row reads the same selected rooms, including teammate rooms; Joint encodes them in one pass. (b) Three recorded requests per event. Stale and Recovery: decisions. Max $|\Delta p|$: largest deviation from re-perceiving every room.

(a) Composition

Perception	Reader	Servers	Joint agr.	SR / SPL
Joint	Single	1	—	0.667 / 0.357
Independent	Single	1	0.753	0.653 / 0.344
Independent	Peer	2	0.769	0.667 / 0.362

(b) Lifecycle

Event	Stale	Recovery	Max $\|\Delta p\|$
Room split	0–1	≤ 1	0.030
Room merge	0	≤ 1	0.010
Delayed update	$\leq$ delay	after update	0.010
Owner change	0	≤ 1	0.025
Prompt change	0	immediate	0.010

Table 5: Spatial-address ablation on HM3D val-72 (seed 0, per decision). Mean / p90: end-to-end decision latency including segmentation and environment stepping, a wider boundary than the execution latency of Table 2. MP3D MB: per step, offline replay, 12-m count-matched tiles.

Partition	Units	Tokens	Re-perceptions	Encode (ms)	Mean / p90 (s)	SR	SPL	MP3D MB
Tiles, 4 m	8.95	77	5.72	635	7.71 / 12.3	0.694	0.350	8.4
Tiles, size-matched	7.31	80	4.57	511	7.30 / 14.6	0.681	0.366	8.8
Tiles, count-matched (8 m; 12 m for MP3D)	4.25	87	2.79	339	5.55 / 8.47	0.625	0.341	11.3
Rooms	3.39	83	2.72	354	**4.67 / 8.10**	**0.708**	**0.370**	**7.4**

arrives), and its frontier probabilities stay within 0.03 of full re-perception. Figure 2 (right) shows an owner change and two new rooms.

Spatial addressing. We replace rooms with 4-m, size-matched, or count-matched (8-m) square tiles, keeping perception, retention, sharing, and planning fixed (Table 5). Rooms need the fewest re-perceptions per decision (2.72) and give the lowest end-to-end latency (4.67 s) and highest SR (0.708). At a similar token budget to count-matched tiles (83 vs. 87), rooms cut mean latency by 16%, because a change inside one room does not invalidate its neighbors; in an offline MP3D replay, rooms move 35% fewer bytes than count-matched tiles.

5 Scope and Limitations

Exact reuse needs one shared planner configuration and a causal decoder whose only position-dependent key transformation is a rotary phase after the stored key (Appendix A.3). Multimodal variants such as M-RoPE also rotate keys after projection and should admit rephasing over shifted position triples, but we test one int4 planner; heterogeneous teams would need a learned translator such as Cache-to-Cache (Fu et al., 2026). Independent construction reaches 79–81% of joint encoding's own run-to-run agreement; closing the rest needs receiver-side recomputation, and we do not compare against KVCOMM-style correction or CacheBlend-style recomputation (Ye et al., 2025; Yao et al., 2025), which is compatible with our admission layer. Savings depend on version churn (45% with rendered annotations, 42% without; Table 8). Total traffic grows superlinearly with team size (4.2/11.3/20.3 MB per step for $N{=}2/3/4$) because each robot receives from $N{-}1$ owners. Missing messages fall back to local encoding (Appendix A.1), but loss, dropout, and links below 100 Mbps are not evaluated, and the real-robot study has 10 trials.

6 Conclusion

Map synchronization shares evidence but not the computation performed over it. Region-addressed attention messages factor a frozen planner's computation at that boundary: each room's owner builds its attention state once from the shared prefix and the room alone, and a receiver admits it only while its region, version, and owner match the current map. Its one approximation, building rooms independently, retains 79–81% of attainable agreement without a measurable loss in SR. Delivered this way, teammate rooms need 45% less planner GPU time than re-perception and match a centralized shared cache while planning stays onboard, in simulation and on two physical robots.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, figure planning, and LaTeX drafting. The authors verified cited metadata, technical claims, equations, numerical results, and retained artifacts against the original publications and experiment records. Generative AI was not used to generate experimental measurements.

REPRODUCIBILITY STATEMENT

The method specifies the transmitted tensors, positional treatment, compatibility assumptions, message lifecycle, and traffic model. Captions report dataset splits, latency boundaries, repetitions, and payload scope. Appendix C records the model, map, NVIDIA hardware used in the experiments, and metric definitions; Appendix D provides retention, lifecycle, partition, memory, and deployment measurements. The reproduced planner-channel boundary begins after map construction and synchronization.

REFERENCES

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pp. 667–676, 2017. doi: 10.1109/3DV.2017.00081. URL https://doi.org/10.1109/3DV.2017.00081.

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16344–16359, 2022. doi: 10.52202/068431-1189. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

Ziluo Ding, Zeyuan Liu, Zhirui Fang, Kefan Su, Liwen Zhu, and Zongqing Lu. Multi-agent coordination via multi-level communication. In *Advances in Neural Information Processing Systems*, volume 37, pp. 118513–118539, 2024. doi: 10.52202/079017-3763. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d6be51e667e0b263e89a23294b57f8cf-Paper-Conference.pdf.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.03215.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In *Proceedings of Machine Learning and Systems*, volume 6, pp. 325–338, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/a66caa1703fe34705a4368c3014c1966-Paper-Conference.pdf.

Junhao Hu, Wenrui Huang, Weidong Wang, Haoyi Wang, Tiancheng Hu, Zhang Qin, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. EPIC: Efficient position-independent caching for serving large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24391–24402. PMLR, 2025. URL https://proceedings.mlr.press/v267/hu25j.html.

Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. In *International Conference on Learning Representations*, pp. 12963–12978, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/37c6d0bc4d2917dcbea693b18504bd87-Paper-Conference.pdf.

Yue Hu, Shaoheng Fang, Zixing Lei, Yiqi Zhong, and Siheng Chen. Where2comm: Communication-efficient collaborative perception via spatial confidence maps. In *Advances in Neural Information Processing Systems*, volume 35, pp. 4874–4886, 2022. doi: 10.52202/068431-0352. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/1f5c5cd01b864d53cc5fa0a3472e152e-Paper-Conference.pdf.

Miguel Julià, Arturo Gil, and Oscar Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33:427–444, 2012.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Hao Liu, Matei Zaharia, and Pieter Abbeel. RingAttention with blockwise transformers for near-infinite context. In *International Conference on Learning Representations*, pp. 3992–4008, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/1119587863e78451f080da2a768c4935-Paper-Conference.pdf.

Xinzhu Liu, Di Guo, Huaping Liu, and Fuchun Sun. Multi-agent embodied visual semantic navigation with scene prior knowledge. *IEEE Robotics and Automation Letters*, 7(2):3154–3161, 2022. doi: 10.1109/LRA.2022.3145964. URL https://doi.org/10.1109/LRA.2022.3145964.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. CacheGen: KV cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 38–56, 2024b. doi: 10.1145/3651890.3672274. URL https://doi.org/10.1145/3651890.3672274.

Yat Long (Richie) Lo, Biswa Sengupta, Jakob Foerster, and Michael Noukhovitch. Learning multi-agent communication with contrastive learning. In *International Conference on Learning Representations*, pp. 51029–51046, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/de2ad3ed44ee4e675b3be42aa0b615d0-Paper-Conference.pdf.

Dongyang Ma, Yan Wang, and Lan Tian. Block-attention for efficient prefilling. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2409.15355. URL https://arxiv.org/abs/2409.15355.

Zhao Mandi, Shreeya Jain, and Shuran Song. RoCo: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 286–299, 2024. doi: 10.1109/ICRA57147.2024.10610855. URL https://doi.org/10.1109/ICRA57147.2024.10610855.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. Parallel context windows for large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6383–6402, 2023. doi: 10.18653/v1/2023.acl-long.352. URL https://aclanthology.org/2023.acl-long.352/.

Zhixuan Shen, Haonan Luo, Kexun Chen, Fengmao Lv, and Tianrui Li. Enhancing multi-robot semantic navigation through multimodal chain-of-thought score collaboration. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(14):14664–14672, 2025. doi: 10.1609/aaai.v39i14.33607. URL https://doi.org/10.1609/aaai.v39i14.33607.

Zhixuan Shen, Yijie Zeng, Shengxiang Luo, Tianrui Li, and Haonan Luo. Plug-and-play label map diffusion for universal goal-oriented navigation. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/61342.

Xiangyu Shi, Marco Chiesa, Jr. Maguire, Gerald Q., and Dejan Kostic. KVComm: Enabling efficient LLM communication through selective KV sharing. In *International Conference on Learning Representations*, 2026. doi: 10.48550/arXiv.2510.03346. URL https://arxiv.org/abs/2510.03346.

Arnoud Visser, Julian de Hoog, Abraham Jiménez-González, and José-Ramón Martínez de Dios. Discussion of multi-robot exploration in communication-limited environments. In *Workshop on Towards Fully Decentralized Multi-Robot Systems: Hardware, Software and Integration at the IEEE International Conference on Robotics and Automation*, 2013.

Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Theo Gervet, John Turner, Aaron Gokaslan, Noah Maestre, Angel Xuan Chang, Dhruv Batra, Manolis Savva, Alexander William Clegg, and Devendra Singh Chaplot. Habitat-matterport 3d semantics dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4927–4936, June 2023. doi: 10.1109/CVPR52729.2023.00477. URL https://openaccess.thecvf.com/content/CVPR2023/html/Yadav_Habitat-Matterport_3D_Semantics_Dataset_CVPR_2023_paper.html.

Dingkang Yang, Kun Yang, Yuzheng Wang, Jing Liu, Zhi Xu, Rongbin Yin, Peng Zhai, and Lihua Zhang. How2comm: Communication-efficient and collaboration-pragmatic multi-agent perception. In *Advances in Neural Information Processing Systems*, volume 36, pp. 25151–25164, 2023. doi: 10.52202/075280-1093. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4f31327e046913c7238d5b671f5d820e-Paper-Conference.pdf.

Xinyu Yang, Tianqi Chen, and Beidi Chen. APE: Faster and longer context-augmented generation via adaptive parallel encoding. In *International Conference on Learning Representations*, 2025. doi: 10.48550/arXiv.2502.05431. URL https://arxiv.org/abs/2502.05431.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 94–109, 2025. doi: 10.1145/3689031.3696098. URL https://doi.org/10.1145/3689031.3696098.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. KVCOMM: Online cross-context KV-cache communication for efficient LLM-based multi-agent systems. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2510.12872.

Bangguo Yu, Hamidreza Kasaei, and Ming Cao. Co-NavGPT: Multi-robot cooperative visual semantic navigation using large language models, 2023. URL https://arxiv.org/abs/2310.07937.

Shiju Zhao, Junhao Hu, Rongxiao Huang, Jiaqi Zheng, and Guihai Chen. MPIC: Position-independent multimodal context caching system for efficient MLLM serving. *arXiv preprint arXiv:2502.01960*, 2025. doi: 10.48550/arXiv.2502.01960. URL https://arxiv.org/abs/2502.01960.

Zhipu AI (Z.ai). GLM-Edge-V-5B. Official model repository, 2024. URL https://huggingface.co/zai-org/glm-edge-v-5b/tree/595da783cdf468bf0616b9c05757e911676b2f39. Revision 595da783cdf468bf0616b9c05757e911676b2f39.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation*, pp. 193–210, 2024. ISBN 978-1-939133-40-3. URL https://www.usenix.org/conference/osdi24/presentation/zhong-yinmin.

Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in

multi-agent systems. *arXiv preprint arXiv:2511.20639*, 2025. doi: 10.48550/arXiv.2511.20639. URL https://arxiv.org/abs/2511.20639.

A Regional-State Construction and Protocol Details

This appendix summarizes the evaluated regional adapter, canonical extraction, and lifecycle invariants. The proposed planner-state layer is parameterized by any front-end adapter that emits the fields in $\mathfrak{i}_t$: live episode-local handles, regional inputs and patch sets, and the resulting ordered token sequences. Its guarantees hold pointwise for the emitted descriptor.

A.1 Realized decision interface

Equation 1 uses $\mathrm{Resolve}_t(S_t, \boldsymbol{\kappa}_t)$ as high-level notation and suppresses execution-layer candidate availability. The full interface is defined here.

At the memory-layer boundary, the navigation front end has fixed the synchronized map M_t, the ordered selected-region sequence S_t, every regional input $(X_{r,t}, \mathcal{P}_{r,t})$, the resulting ordered visual tokens $Z_{r,t}$, and the task-token stream u_t. For $S_t = (q_1, \ldots, q_m)$, current metadata fix the expected key vector $\boldsymbol{\kappa}_t = (\kappa_{q_1,t}, \ldots, \kappa_{q_m,t})$. The surrounding execution layer supplies one aligned candidate slot per selected region:

$$\mathcal{C}_t = (c_{t,1}, \ldots, c_{t,m}), \qquad c_{t,a} \in \{\bot\} \cup \mathbb{M}, \tag{3}$$

where $\mathbb{M}$ is the record space and $\bot$ denotes an unavailable candidate. Resolve applies the current-key test

$$I_t = \langle a \in \{1, \ldots, m\} \mid c_{t,a} \neq \bot \ \wedge \ \mathrm{key}(c_{t,a}) = \kappa_{q_a,t} \rangle_{\uparrow}, \qquad A_t = (q_a)_{a \in I_t}, \tag{4}$$

where key returns the complete (r, v, o) key and $\langle \cdot \rangle_{\uparrow}$ retains increasing slot order. The total result is

$$\mathrm{Resolve}_t(S_t, \boldsymbol{\kappa}_t, \mathcal{C}_t) = \begin{cases} \mathsf{NoValidSnapshot}, & I_t = \varnothing, \\ \mathrm{Freeze}_t((c_{t,a})_{a \in I_t}), & I_t \neq \varnothing, \end{cases} \tag{5}$$

where Freeze_t forms an immutable snapshot from the accepted records and the forward builder's realized layout. NoValidSnapshot denotes the absence of a nonempty valid snapshot. It is an interface status rather than a fallback action: no Read is performed, and the surrounding execution policy determines the response. In our implementation, a receiver that lacks a required record, for example after a lost or delayed message, reports the missing regions, and the driver re-sends that decision's regional images once; the receiver then encodes them itself, as in re-perception. With no active region, the robot issues an ordinary single-pass planner call. Failures therefore degrade to local encoding rather than to stale state. We collect the inputs and resulting nonempty layout as

$$\begin{aligned} \mathfrak{i}_t &= \Big(M_t, S_t, \big((X_{r,t}, \mathcal{P}_{r,t}, Z_{r,t})\big)_{r \in S_t}, \boldsymbol{\kappa}_t, \mathcal{C}_t, u_t \Big), \\ \mathfrak{d}_t &= (\mathfrak{i}_t, A_t, \Lambda_t). \end{aligned} \tag{6}$$

$\mathfrak{i}_t$ is the invocation descriptor. After a nonempty resolution, $\mathfrak{d}_t$ is the realized decision descriptor. The existing forward builder derives row assignments, logical positions, masks, and attention biases Λ_t from the admitted records. Resolve tests the aligned candidate at each position against its expected complete key, retains accepted positions in the order supplied by S_t, and freezes the resulting snapshot. Different front-end or execution policies can produce different descriptors; every method guarantee compares executions with the same $\mathfrak{d}_t$. The memory interface consumes these realized choices and defines no region ordering, matching, waiting, timeout, fallback, or admission policy beyond the key test above.

A.2 Region tracking and patch assignment

Let F_t be the explored, obstacle-free cells of the shared map, and let $D_t(x)$ be the distance from $x \in F_t$ to the nearest wall or unexplored cell. The regional adapter uses local maxima c_k of D_t to seed the geodesic partition

$$\Omega_{k,t} = \left\{ x \in F_t : k = \arg\min_{k'} \mathrm{dist}_{F_t}(x, c_{k'}) \right\}. \tag{7}$$

At each step, the adapter emits one live episode-local handle for each current region. Its updates across shape changes, splits, and merges determine the current associations and invalidate records that no longer match a live handle. The emitted handles enter i_t as fixed spatial addresses for the memory layer; the memory guarantees are independent of the adapter's association procedure.

The vision encoder maps the rendered map to a 24×24 patch grid. Let C_p be the map cells covered by visual patch p, and let W_t be the set of patches kept by the WallPPR selector at step t (Section 4.1). At every decision step, region r receives

$$\mathcal{P}_{r,t} = \mathrm{Cap}_{256}\left\{p \in W_t : |C_p \cap \Omega_{r,t}| > \tfrac{1}{2}|C_p|\right\}, \tag{8}$$

Cap_{256} denotes the adapter output after limiting the qualifying patches to at most 256. Preprocessing maps this patch set to the ordered sequence $Z_{r,t}$ in Equation 6, which the memory layer consumes unchanged. The selection is made by the navigation front end, not the memory layer, and can change with robots, frontiers, and the previous goal; because $\mathcal{P}_{r,t}$ enters the version (Equation 10), a change in selection invalidates the record instead of altering an admitted message. Patch assignment is recomputed every step rather than attached permanently to the region identifier.

Let $M_{r,t}$ be the dilated region mask, J_t the target-independent map rendering, and $\boldsymbol{\gamma}$ the rendering color of explored empty floor. The exact regional input is

$$X_{r,t} = M_{r,t} \odot J_t + (1 - M_{r,t}) \odot \boldsymbol{\gamma}. \tag{9}$$

For a compatible planner configuration, the version is computed from all region-varying materialization inputs:

$$v_{r,t} = \mathrm{hash}(X_{r,t}, \mathcal{P}_{r,t}). \tag{10}$$

The rendering includes transient spatial annotations, such as frontier letters and robot arrows. Together, Equations 8 and 9 determine the version in Equation 10. A change to the mask, patch set, or any rendered content inside the masked image therefore causes the owner to rebuild the memory.

A.3 Compatibility and canonical extraction

The evaluated robots share a planner configuration comprising the checkpoint, image preprocessing, complete fixed prefix, positional convention, and cache layout. Every planner prompt begins with the same L_0 prefix tokens, ending at the begin-image token, and its layerwise prefix state is computed once. To construct a record, the owner encodes $X_{r,t}$ and retains $T_{r,t}$ tokens indexed by $\mathcal{P}_{r,t}$. It places them at canonical positions $L_0, \ldots, L_0 + T_{r,t} - 1$.

For normalized hidden states $\widetilde{H}^\ell$, let ϕ_Q^ℓ and ϕ_K^ℓ denote any position-independent query/key transformation used by the frozen model, or the identity when none is present. A layer forms

$$\bar{Q}^\ell = \phi_Q^\ell(\widetilde{H}^\ell W_Q^\ell), \qquad \bar{K}^\ell = \phi_K^\ell(\widetilde{H}^\ell W_K^\ell), \qquad V^\ell = \widetilde{H}^\ell W_V^\ell, \quad (Q_p^\ell, K_p^\ell) = \mathrm{RoPE}_p(\bar{Q}^\ell, \bar{K}^\ell). \tag{11}$$

The payload stores the actual regional rows immediately before the rotary phase, $\bar{K}_r^\ell$, together with V_r^ℓ. Construction uses only the fixed prefix and one region; no other region and no task query is present. This is why a record can later be ordered and rephased, and also why it lacks the cross-region interactions and final-position hidden context of joint encoding. Exact key rephasing requires RoPE to be the only position-dependent key transformation, applied after the stored $\bar{K}_r^\ell$. Any position- or sequence-length-dependent operation that changes regional hidden states, pre-RoPE keys, or values makes the canonical payload incompatible unless the affected states are recomputed in the final layout.

For admitted order $A_t = (r_1, \ldots, r_n)$, the receiver places branch a at $s_a = L_0 + \sum_{b<a} T_{r_b,t}$ and rephases its stored keys as

$$K_a^\ell = \mathrm{RoPE}\big(\bar{K}_{r_a}^\ell; s_a, \ldots, s_a + T_{r_a,t} - 1\big). \tag{12}$$

Task token $u_{t,k}$ then occupies $p(u_{t,k}) = L_0 + \sum_{a=1}^{n} T_{r_a,t} + k$. Rephasing changes neither the regional hidden state nor the stored pre-RoPE key content or value.

A.4 Read equations

This construction defines a canonical fork and join computation. The shared prefix forks into regional branches that are constructed independently, and the branches join only when the current task

tokens read them. Let $\mathcal{E}_\theta(P; Z_{a,t})$ denote the canonical pre-RoPE K/V extraction for branch a. The canonical fork–join graph is

$$\begin{aligned} \mathcal{B}_t^{\mathrm{FJ}} &= \bigoplus_{a=1}^{n} \rho_{s_a}(\mathcal{E}_\theta(P; Z_{a,t}))\,, \\ \pi_t^{\mathrm{FJ}} &= \mathcal{R}_\theta(P, \mathcal{B}_t^{\mathrm{FJ}}, u_t; \Lambda_t), \end{aligned} \tag{13}$$

where $\bigoplus$ follows the admitted order and $\mathcal{R}_\theta$ is the task-token recurrence.

At each layer, the receiver advances only the task tokens. Their queries attend to the prefix, every unmasked regional row in Σ_t, and earlier task tokens under the fixed snapshot layout Λ_t:

$$\mathrm{Attn}_{\Lambda_t}\left(Q^\ell_{u_t}, [K^\ell_{\mathrm{prefix}}; K^\ell_1; \ldots; K^\ell_n; K^\ell_{u_t}], [V^\ell_{\mathrm{prefix}}; V^\ell_1; \ldots; V^\ell_n; V^\ell_{u_t}]\right). \tag{14}$$

The updated task states enter the next layer, so later queries already contain evidence read from the regional rows. The regional rows remain fixed throughout the forward pass, and locally materialized and received records use the same attention interface.

A.5 Proofs of canonical-branch realization and current-state reuse

Ordinary joint encoding provides a reference graph for locating the paths removed by independent regional construction. It evaluates

$$Y_t^{\mathrm{joint}} = [P; Z_{1,t}; \ldots; Z_{n,t}; u_t] \tag{15}$$

under the same causal convention and final task-read layout. For regional token $z_{a,j}$, the joint and fork–join graphs expose different construction predecessors:

$$\begin{aligned} \mathrm{Pred}_{\mathrm{joint}}(z_{a,j}) &= [P; Z_{1,t}; \ldots; Z_{a-1,t}; Z_{a,t,<j}], \\ \mathrm{Pred}_{\mathrm{mem}}(z_{a,j}) &= [P; Z_{a,t,<j}]. \end{aligned} \tag{16}$$

Their task tokens have the same directly visible slots,

$$\mathrm{Vis}_{\mathrm{joint}}(u_{t,k}) = \mathrm{Vis}_{\mathrm{mem}}(u_{t,k}) = [P; Z_{1,t}; \ldots; Z_{n,t}; u_{t,<k}], \tag{17}$$

but the regional K/V rows in those slots can have different ancestry. Token $z_{a,j}$ is materialized at $c_{a,j} = L_0 + j$ and read at $p_{a,j} = s_a + j = c_{a,j} + \delta_a$, where $\delta_a = \sum_{b<a} T_{r_b,t}$. Receiver-side rephasing supplies the final key phase $p_{a,j}$ without recomputing the canonical hidden state, pre-RoPE key content, or value.

Let $\mathcal{B}_t^{\mathrm{joint}}$ be the layerwise regional rows produced by Equation 15, and let $\mathcal{B}_t^{\mathrm{mem}}$ be the rows from canonical materialization and receiver-side rephasing. We write $\pi_t^{\mathrm{joint}} = \mathcal{R}_\theta(P, \mathcal{B}_t^{\mathrm{joint}}, u_t; \Lambda_t)$ and define π_t^{mem} analogously.

For Property 1, fix a realized descriptor $\mathfrak{d}_t$, a compatible causal planner, and its nonempty snapshot, with every row bound to one current record. The compared constructions use identical preprocessing, regional embeddings, prefix, task tokens, final layout, masks, and attention biases. Each canonical branch uses the same construction mask and bias as its ordinary causal evaluation and places $Z_{a,t}$ immediately after P. Evaluation is deterministic and exact, non-attention operations are tokenwise, causal computation is suffix-invariant, and RoPE is the only position- or sequence-length-dependent operation after the stored pre-RoPE key. Under these assumptions, for every causal suffix W_a,

$$\Psi_{r_a,t} = \mathcal{E}_\theta(P; Z_{a,t}) = \mathcal{E}_{\theta, Z_{a,t}}([P; Z_{a,t}; W_a])\,, \tag{18}$$

where $\mathcal{E}_{\theta,Z_{a,t}}$ extracts the rows associated with $Z_{a,t}$ from the indicated ordinary causal evaluation.

Fix a regional branch a and an arbitrary causal suffix W_a. Canonical materialization $[P; Z_{a,t}]$ and the ordinary sequence $[P; Z_{a,t}; W_a]$ have identical tokens, positions, visible predecessors, masks, and attention biases through the end of $Z_{a,t}$. The initial states therefore agree. Suppose the states agree through layer ℓ. Identical query, key, and value projections give identical attention outputs. The same tokenwise output projection, residual path, normalization, and MLP then give identical states at layer $\ell + 1$. Induction gives identical regional pre-RoPE keys and values at every layer. Because this argument applies independently to every a, Equation 18 holds for all admitted regions, not only the first region in the packed read order.

Both the fork–join definition and the method then apply the same ρ_{s_a} to every branch and advance the same task-token recurrence over the resulting rows. Hence

$$\mathcal{B}_t^{\mathrm{mem}} = \mathcal{B}_t^{\mathrm{FJ}}, \qquad \pi_t^{\mathrm{mem}} = \pi_t^{\mathrm{FJ}}. \tag{19}$$

When $|A_t| = 1$, this sole regional branch is also the first block of Equation 15; suffix invariance gives $\pi_t^{\mathrm{mem}} = \pi_t^{\mathrm{joint}}$. With multiple regions, the joint graph additionally contains the construction paths in Equation 16 and therefore defines a different computation from the fork–join graph.

Proof of Proposition 1. Let c be a receiver context that reads r first, $[P; Z_{r,t}; W]$ with W the remaining regions and task tokens. Equation 18 with suffix W gives rows of r in c equal to $\mathcal{E}_\theta(P; Z_{r,t})$ before rephasing, which proves the first claim. A message exact in every context is in particular exact in c, so it equals $\mathcal{E}_\theta(P; Z_{r,t})$. It is also exact in any context c' in which other regions precede r; there it equals the joint rows of r, which therefore equal $\mathcal{E}_\theta(P; Z_{r,t})$ up to rotary phase, i.e. the preceding regions leave the rows of r unchanged. □

Selective-reuse assumptions and proof. Property 2 fixes a realized descriptor $\mathfrak{d}_t$ with a nonempty snapshot and a compatible planner configuration. Assume deterministic preprocessing and materialization, a collision-free episode-local version digest, and exact payload preservation by storage and transport. Bind stores only the output of $\mathrm{Materialize}$ for the inputs named by each record. Under the admission rule in Equation 4, replacing any subset of freshly materialized current records by resident or received records with the same complete keys, admitted order, and layout leaves $\mathcal{B}_t^{\mathrm{mem}}$, the task-token states, output logits, and π_t^{mem} unchanged.

For one admitted record, if it is freshly materialized, the claim is immediate. If it is reused, its admitted key matches the current address, version, and owner. By the binding provenance assumption, the record was produced from the inputs named by that key. Under the episode-local collision-free assumption, a matching version implies the same $(X_{r,t}, \mathcal{P}_{r,t})$; fixed deterministic preprocessing therefore yields the same regional token sequence. Planner compatibility and deterministic materialization then give exactly the payload that fresh materialization would produce. Applying this argument to every admitted record, then applying the same ρ_{s_a} and fixed row layout, yields identical regional rows; the task-token recurrence, output logits, and final distribution are unchanged.

A.6 Lifecycle transition rules

Map synchronization retains the source robot of newly explored cells. If $A_{i,t}(r)$ is the source-labelled area contributed by robot i to region r at step t, then $o_{r,t} \in \arg\max_i A_{i,t}(r)$. The ownership routine returns one current owner, and that value becomes the owner field in $\boldsymbol{\kappa}_t$. The memory contract is pointwise in this value and requires it to be the sole writer for the current regional memory.

At every decision, robots recompute $\mathcal{P}_{r,t}$ and $v_{r,t}$ from the synchronized map. A new region has no reusable memory; a changed version is rebuilt once by its owner. Peers cache a received record under $(r, v, o_{r,t})$ and request it again only when they do not hold the currently required tuple. Remote records unused for eight decisions are evicted.

Splits and merges change the region associations, and an ownership change changes the current writer. In each case, a cached record is reusable only if its address, version, and owner still match the current region metadata. The same test prevents a delayed obsolete payload from being treated as current. At the end of an episode, region identities and their cached records are cleared. Within any one planner forward, the selected memory tensors, logical positions, masks, and biases are held fixed as computational inputs. Transport-level synchronization remains an implementation choice. The target and compatibility configuration are constant in the evaluated episodes, whereas region versions and owners may change between decisions. The delayed-update case is evaluated in Section 4.4.

A.7 Fixed-memory delivery proof and accounting

Property 3 fixes one realized descriptor, compatible planner, and planner-forward memory snapshot. All execution plans start from identical task-token states and use the same ordered prefix and regional K/V rows, logical positions, masks, and attention biases. Local and remote holders form a

disjoint, complete partition of those rows at every layer, non-attention operations are deterministic and tokenwise, and the equality statement is in exact arithmetic. Each remote holder serves all layer requests in that forward from the same immutable payload.

For one query row and attention head, let m_S, l_S, and o_S denote the maximum score, shifted exponential sum, and shifted value-weighted sum over a row subset S with at least one finite, unmasked score. For disjoint subsets A and B, their stable summaries combine as

$$\begin{aligned} m_{AB} &= \max(m_A, m_B), \\ l_{AB} &= e^{m_A - m_{AB}} l_A + e^{m_B - m_{AB}} l_B, \\ o_{AB} &= e^{m_A - m_{AB}} o_A + e^{m_B - m_{AB}} o_B, \\ \mathrm{Attn}(A \cup B) &= o_{AB} / l_{AB}. \end{aligned} \tag{20}$$

Each score includes the usual scale, mask, and attention bias. Factoring the global softmax denominator and numerator around m_{AB} gives the stable union summary in Equation 20. Latent sharing evaluates Equation 14 directly. In remote recall, each owner applies the receiver's task query, logical positions, masks, and attention biases to its assigned regional rows and returns their partial-softmax summary. The receiver forms a local summary over the prefix, resident regional rows, and preceding task-token rows. These local and remote subsets form a disjoint, complete partition of the rows in Equation 14; subsets without a finite, unmasked score are omitted. For more than two holders, the receiver recursively folds Equation 20 over their summaries.

We prove Property 3 by induction over transformer layers. At the first layer, identical initial task-token states yield identical queries and local task-token K/V rows. The merge in Equation 20 equals attention over the union in exact arithmetic, so the task-token attention outputs are identical. Applying the same output projection, residual path, normalization, and MLP therefore produces identical task-token hidden states. These states yield identical queries and local task-token K/V rows at the next layer, completing the induction through all L layers. The output logits and final frontier distribution are consequently identical. In finite precision, query and memory casts, fused kernels, accumulation precision, and reduction order can produce numerical differences.

For the evaluated two-robot setting with one remote holder, let h_q and h_{kv} denote the numbers of query and key–value heads, respectively. Let d_h be the head width, $|u_t|$ the number of task tokens, and T_{miss} the total number of nonresident regional tokens held by that remote holder. The resulting byte model is

$$\hat{C}_{\mathrm{recall}} = L|u_t|\, 6 h_q d_h, \qquad C_{\mathrm{share}} = L T_{\mathrm{miss}}\, 4 h_{kv} d_h. \tag{21}$$

Remote recall pays this query-dependent traffic at every layer. Latent sharing transfers each missing valid payload once and reuses it until invalidation. The byte conventions, horizon accounting, and crossover follow.

For the evaluated two-robot setting with one nonempty remote holder, remote recall transmits bf16 queries and returns fp32 partial value accumulators, whereas latent sharing transmits bf16 keys and values. The factor 6 in Equation 21 counts two bytes for each query element and four for each returned value element; the factor 4 counts one two-byte key and one two-byte value per missing token. The recall expression omits compact request metadata, two fp32 softmax scalars per query and head, and the final-layer saving. The sharing expression counts each missing regional row once. These conventions yield the crossover in Equation 23 below; changing precision, compression, topology, or cache residency changes the boundary accordingly.

For a window of H realized planning decisions, let $T_{\mathrm{xfer},t}$ be the number of regional token positions whose complete L-layer payloads are transmitted at decision t to satisfy current receiver cache misses. Let R_t be the number of remote holders assigned nonempty, disjoint subsets of the admitted nonresident rows in the realized Σ_t, under point-to-point query replication. The byte model extends to

$$\begin{aligned} \hat{C}^{1:H}_{\mathrm{recall}} &= 6 L h_q d_h \sum_{t=1}^{H} R_t |u_t|, \\ C^{1:H}_{\mathrm{share}} &= 4 L h_{kv} d_h \sum_{t=1}^{H} T_{\mathrm{xfer},t}. \end{aligned} \tag{22}$$

Payload transfers can follow first demand, active-set entry, record creation, version or owner change, a split or merge, eviction, or an episode reset. The accounting conditions on the realized snapshot and does not select a wait, timeout, admission, or transfer-order policy.

Writing $g_{\mathrm{GQA}} = h_q/h_{kv}$, one-shot latent sharing uses less traffic than remote recall for one remote holder when

$$T_{\mathrm{miss}} < 1.5\, g_{\mathrm{GQA}} |u_t|. \tag{23}$$

With R nonempty remote holders, point-to-point query replication, and T_{miss} equal to the total missing tokens across them, the corresponding byte model is

$$\hat{C}_{\mathrm{recall}}(R) = RL|u_t|\, 6h_q d_h, \qquad T_{\mathrm{miss}} < 1.5R\, g_{\mathrm{GQA}} |u_t|. \tag{24}$$

Multicast or hierarchical aggregation changes this scaling.

B Remote Recall as a Placement Reference

Remote recall evaluates the same fixed-snapshot Read as latent sharing but leaves regional rows at their owner and sends the receiver's post-RoPE task queries at every layer. We run it to test the placement property (Property 3), not as a competing interface. Table 6 lists its cost next to latent sharing. On HM3D val-72 it moves 1.09 GB of queries and partial results per step, against 4.5 MB for latent sharing, and it needs 4.53 s of planner-server time. On a 100-Mbps link with a 20-ms round-trip time it waits 41.0 s per decision; it completed 3 of 18 val-18 episodes (16 decisions) in that setting, so we report no navigation metrics for it. Traffic counts each planner-channel message once.

Table 6: Remote recall versus latent sharing. Val-72: payload and summed planner-server time per step, and mean/p90 execution latency per decision. 100 Mbps: request-to-frontier latency and link wait per decision, and traffic per step, on HM3D val-18.

	HM3D val-72			100 Mbps, val-18		
Placement	Payload	Server-s	Exec. (s)	Dec. (s)	Wait (s)	Traffic
Remote recall	1.09 GB	4.53	4.7 / 7.8	43.7	41.0	0.50 GB
Latent sharing	4.5 MB	1.68	2.7 / 3.4	3.6	0.346	4.2 MB

C Implementation Details

The map is 480×480 at 5-cm resolution, and each masked room image is resized to 672×672 for the 24×24 token grid. The active-room rule requests rooms containing a robot or frontier, their four-neighbor-adjacent rooms, rooms whose version changed since the preceding decision, and the previous frontier's room. This rule defines membership in S_t; the navigation front end supplies its realized order, which the memory layer preserves. The per-room cap is 256 positions; caps of 64, 128, and 256 all retain 85 tokens per decision in a cap-sweep trace, so the cap does not bind. With the matched run's 83-token active set, raw bf16 memory occupies 10.2 MB before framework and allocator overhead, but only about 36 nonresident tokens cross the link per step. We enable deterministic input hashing but do not assert bitwise-deterministic int4 GPU kernels. Code, configurations, and evaluation scripts will be released upon publication.

D Deployment Resource Snapshot

Table 7 establishes deployability and memory footprint in a separate end-to-end minival harness. A separate val-72 placement run on L40S GPUs holds rooms, retention, positions, prompt, and decoding fixed. One server records 2.1 s mean execution latency and 0.653 SR; two planner processes on one GPU record 4.2 s and 0.681 SR; one GPU per robot records 2.8 s and 0.708 SR; Table 7 reports memory. Property 3 concerns one fixed snapshot; these SRs are closed-loop outcomes under non-deterministic int4 kernels, and their differences lie within the ± 15.2-point 95% interval of a difference over 72 episodes.

Latency boundaries. Latencies in this paper use different boundaries and are compared only within one table or figure. Table 2 reports closed-loop execution latency per decision on val-72. Table 5 reports end-to-end decision latency on val-72 that also includes segmentation and environment stepping, so its Rooms row (4.67 s) and the latent-sharing row of Table 2 (2.7 s) describe the same configuration under different latency boundaries, and Table 5 uses seed 0 only. Figure 4(b) and Table 3 report replayed planner-server latency per step, excluding simulator and driver work, where a step ends when all N robots have decided. Figure 4(a) reports request-to-frontier latency over an emulated link with a 20-ms round-trip time.

Table 7: Deployment footprint in a separate minival harness. VRAM is peak server memory.

Deployment	GPUs	Server VRAM
Single-server 5B, full map	1	4.8 GB
Single-server 5B, pruned	1	4.6 GB
MCoCoNav 9B	1	19 GB
Distributed 2×5B, full map	2	2×5.2 GB
Distributed 2×5B, pruned + latent sharing	2	2×5.2 GB

Table 8 reruns the two model-native cost rows from Table 2 after removing robot arrows and frontier letters from the regional input and representing them as text in the task tokens.

Table 8: Annotation-placement control under the communication-interface protocol. GPU-s is summed planner-server time per step; joint agr. is frontier agreement with joint encoding. Saving is relative to cached re-perception within each rendering condition.

Rendering	Interface	GPU-s	Joint agr.	SR	SPL	Saving
Annotations in region	Cached re-perception	3.07	0.75	0.69	0.35	—
Annotations in region	Latent sharing	1.68	0.76	0.71	0.37	45%
Annotations removed	Cached re-perception	2.14	0.74	0.69	0.35	—
Annotations removed	Latent sharing	1.24	0.75	0.70	0.37	42%