MemoryMesh: Shared Episodic Spatial Memory for Ground-Air Cooperative Search

Aayam Bansal Ishaan Gangwani
Synthetic Sciences
{aayam, ishaan}@syntheticsciences.ai

Abstract

Multi-agent search-and-rescue requires tight coordination between heterogeneous robots, yet real-world communication constraints make sharing full sensory observations impractical. We introduce MemoryMesh, a shared spatial memory for ground-air cooperative search that combines delta-encoded map sharing with a goal-broadcast coordination protocol. Each agent maintains a local spatial memory map of explored regions and broadcasts compact updates encoding newly observed cells together with its current navigation goal. A confidence-weighted merge resolves conflicts when agents observe the same region at different times. Coordination emerges from three mechanisms: goal exclusion zones that penalize frontiers near other agents' announced goals, persistent goal pursuit that reduces oscillation, and role-aware scoring that biases aerial agents toward open areas and ground agents toward wall-dense regions. In a 2D grid-world simulator with partial observability, occlusions, and heterogeneous sensing, we evaluate MemoryMesh and its ablated variants against five baselines on both 80×80 and 150×150 environments. On the larger 150×150 maps with 40 targets, MemoryMesh achieves the highest coverage (0.698) among all methods while using only 208 KB of communication, a 294× reduction relative to full-map sharing at comparable scale. Goal broadcasting with exclusion (DeltaMapShareGoals) proves to be the strongest single coordination mechanism, finding all 20 targets on 80×80 maps with only 100 KB bandwidth and near-zero overlap. Ablations over goal exclusion radius, replanning interval, and component contributions isolate the effect of each mechanism. Robustness experiments confirm graceful degradation under sensor noise (up to 20%), UAV partial occlusion (up to 70%), and packet loss (up to 40%, with zero target degradation), and scaling tests from 2 to 4 agents and 60×60 to 120×120 maps demonstrate consistent behavior.

1. Introduction

Search-and-rescue (SAR) operations demand rapid, thorough coverage of disaster-affected environments. Heterogeneous multi-robot teams combining unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) offer complementary capabilities: drones provide broad aerial overviews while ground robots navigate confined spaces and interact with objects [7, 14]. However, deploying such teams in real-world scenarios faces a fundamental tension between *coordination quality* and *communication cost*.

Full state sharing, transmitting complete occupancy maps at every timestep, achieves maximal situational awareness but requires bandwidth that is often unavailable in disaster environments with degraded infrastructure [12]. At the opposite extreme, independent exploration avoids communication entirely but leads to redundant coverage and missed coordination opportunities. A natural middle ground is delta-encoded sharing, where agents transmit only newly observed cells. While this dramatically cuts bandwidth, it provides no mechanism for coordinating *where* agents should explore next, and agents frequently converge on the same frontiers.

We draw inspiration from episodic memory in cognitive science [19], where agents store structured records of experiences indexed by space and time, and share only *relevant changes* together with lightweight coordination signals. This insight leads to MemoryMesh, a shared spatial memory map in which each entry represents a grid cell annotated with its state (free, wall, target, unknown), observation confidence, timestamp, and observer identity. Unlike pure delta-encoded approaches, MemoryMesh augments cell updates with a goal-broadcast coordination protocol: each agent announces its current navigation goal (a single frontier coordinate), and receivers use exclusion zones around announced goals to avoid redundant exploration. This design adds only 8 bytes per agent per message yet produces measurable reductions in path overlap and improvements in spatial separation.

MemoryMesh contributes the following:

1. A **shared spatial memory map** representation with formal write, read, and merge operations and confidence-

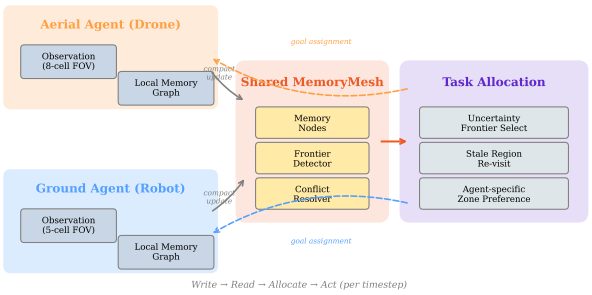

Figure 1. **MemoryMesh system architecture.** Each agent observes its local environment and broadcasts compact updates (delta-encoded cell observations + current navigation goal) to all peers via peer-to-peer communication. The task allocator reads each agent's local spatial memory map and the set of announced goals to assign frontier targets using role-aware scoring with goal exclusion zones. Dashed arrows indicate goal assignments; solid arrows indicate memory updates.

weighted conflict resolution (Sec. 2.2).

2. A **goal-broadcast coordination protocol** in which each agent transmits newly observed cells together with its current navigation goal, enabling exclusion-based coordination at negligible additional bandwidth (Sec. 2.3). Communication is peer-to-peer: each agent broadcasts updates to all others without a central server.
3. A **role-aware frontier scoring** scheme with goal exclusion penalties, persistent goal pursuit, and role-specific biases that assign aerial and ground agents to frontier types matching their sensing capabilities (Sec. 2.4).
4. Comprehensive empirical evaluation against **seven baselines** (including delta-encoded and goal-broadcast variants) on both 80×80 and 150×150 environments, **ablation studies** over goal exclusion radius, replanning interval, and component contributions, and **robustness tests** under sensor noise, partial occlusion, packet loss, and scaling to larger teams and maps (Sec. 3).

2. Method

We consider a search-and-rescue task in a 2D grid environment $\mathcal{E} \in \{0,1\}^{H \times W}$ with two heterogeneous agents: an aerial drone a_d and a ground robot a_g. Hidden targets $\mathcal{T} = \{t_1, \ldots, t_K\}$ are scattered throughout the environment. The goal is to discover all targets as quickly as possible while minimizing communication cost.

2.1. Agent Models

Each agent a_i has position $\mathbf{p}_i \in \mathbb{Z}^2$, a field-of-view (FOV) function $\mathcal{V}_i(\mathbf{p}_i, \mathcal{E})$ returning visible cells, and movement speed s_i.

Aerial agent. The drone has a large square FOV of radius $r_d = 8$ cells, representing a downward-looking camera. It can see over walls (aerial perspective) and moves at speed $s_d = 2$ cells per timestep. Formally, $\mathcal{V}_d(\mathbf{p}) = \{(y,x) : |y - p_y| \le r_d \wedge |x - p_x| \le r_d\}$.

Ground agent. The ground robot has a circular FOV of radius $r_g = 5$ cells with wall occlusion via raycasting. It moves at speed $s_g = 1$ cell per timestep. Its visibility is blocked by walls, making it effective for detailed inspection of rooms but limited in open areas.

2.2. Shared Spatial Memory Map

The core of MemoryMesh is a spatial memory map $\mathcal{M} = \{\mathcal{N}, \mathcal{F}\}$ where $\mathcal{N}$ is a keyed map from cell coordinates to metadata records and $\mathcal{F}$ is a set of frontier cells. Each agent maintains its own local copy of $\mathcal{M}$; copies are kept consistent through the peer-to-peer update protocol described in Sec. 2.3. There is no central server or shared data store.

Memory entries. Each entry $n_c \in \mathcal{N}$ for cell $c = (y, x)$ stores:

$$n_c = (\text{cell}, \text{state}, \text{conf}, \text{time}, \text{observer}) \tag{1}$$

where state $\in \{\texttt{free}, \texttt{wall}, \texttt{target}, \texttt{unknown}\}$, conf $\in [0,1]$ is observation confidence, time is the last observation timestamp, and observer identifies which agent made the observation.

Write operation. When agent a_i observes cells $\mathcal{V}_i$ at time t, the write rule for cell c with state s and confidence γ is:

$$\mathcal{N}[c] \leftarrow \begin{cases} (s, \gamma, t, a_i) & \text{if } c \notin \mathcal{N} \\ (s, \gamma, t, a_i) & \text{if } \gamma > \mathcal{N}[c].\gamma \\ (s, \gamma, t, a_i) & \text{if } \gamma = \mathcal{N}[c].\gamma, t > \mathcal{N}[c].t \\ \mathcal{N}[c] & \text{otherwise} \end{cases} \tag{2}$$

This implements **confidence-weighted conflict resolution**: higher-confidence observations always win; ties are broken by recency. This mechanism is exercised whenever agents observe the same cell with different sensor modalities or under noisy conditions (see Sec. 3.6).

Read operation. Read$(c) = \mathcal{N}[c]$ if $c \in \mathcal{N}$, else `unknown`.

Frontier detection. A free cell c is a frontier if it has at least one unknown 4-connected neighbor c':

$$\mathcal{F} = \{c \in \mathcal{N} : n_c.s = \texttt{free},\ \exists c' \in N_4(c),\ c' \notin \mathcal{N}\} \tag{3}$$

2.3. Peer-to-Peer Communication Protocol

MemoryMesh uses a fully decentralized, peer-to-peer communication architecture. Each agent broadcasts updates to all other agents at every timestep; each receiving agent merges updates into its own local copy of $\mathcal{M}$. There is no central server or coordinator.

Rather than transmitting the full memory map, each agent sends a compact **memory update message** consisting of delta-encoded cell observations and its current navigation goal:

$$m_i^t = (\text{sender}, t, \Delta\mathcal{N}_i^t, \mathbf{g}_i^t) \tag{4}$$

where $\Delta\mathcal{N}_i^t$ contains only cells newly observed since the last broadcast, and $\mathbf{g}_i^t \in \mathbb{Z}^2$ is agent a_i's current frontier goal coordinate. The goal coordinate requires only 8 bytes yet provides the critical coordination signal: by announcing where it intends to explore, each agent enables others to avoid that region through the exclusion mechanism described in Sec. 2.4.

Message size. Each cell update requires 12 bytes (coordinates + state + confidence), and the goal coordinate requires 8 bytes. The total message size is:

$$|\, m_i^t \,| = |\Delta\mathcal{N}_i^t| \times 12 + 8 + 16 \text{ bytes} \tag{5}$$

where the 16-byte header encodes sender identity and timestamp. Because agents transmit only newly observed cells (delta encoding) plus a single goal coordinate, the per-step communication overhead is comparable to pure delta-encoded sharing: approximately 100 KB total over a 200-step episode on 80×80 maps, versus 99 KB for DeltaMap-Share which omits the goal.

Merge operation. Upon receiving update m_j^t from agent a_j, agent a_i applies:

$$\textsc{Merge}(\mathcal{M}_i, m_j^t) : \forall u \in \Delta\mathcal{N}_j^t, \textsc{Write}(u.c, u.s, u.\gamma, t, a_j) \tag{6}$$

followed by frontier re-computation via Eq. (3) and storage of $\mathbf{g}_j^t$ in a goal registry $\mathcal{G} = \{\mathbf{g}_j^t : j \neq i\}$ accessible to the task allocator. The confidence-weighted write rule (Eq. (2)) ensures that conflicts between observations from different agents are resolved deterministically, regardless of message arrival order.

2.4. Role-Aware Task Allocation with Goal Exclusion

Given the updated spatial memory map and the goal registry $\mathcal{G}$, each agent selects a frontier goal using a scoring function that balances information gain, travel distance, separation from other agents' goals, role suitability, and goal exclusion.

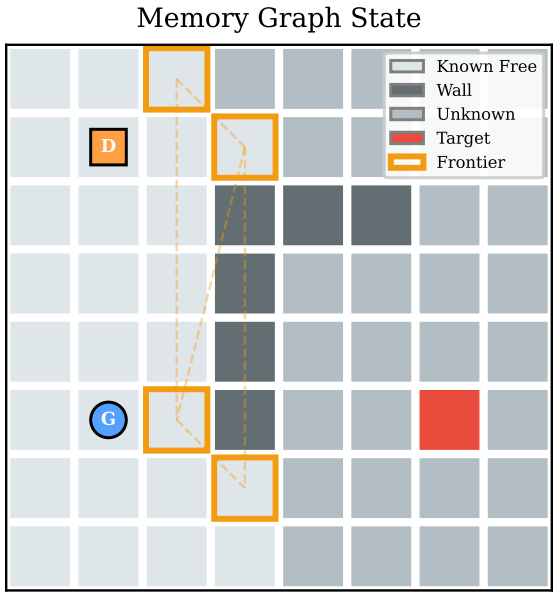

Figure 2. **Spatial memory map illustration.** Known free cells (light gray), walls (dark gray), unknown cells (medium gray), target (red), and frontier cells (orange borders). Drone (D, orange square) and ground robot (G, blue circle) maintain complementary coverage. Each agent holds its own local copy of this map, synchronized via peer-to-peer updates.

For frontier $f \in \mathcal{F}$ and agent a_i at position $\mathbf{p}_i$:

$$\begin{aligned}\text{score}(f, a_i) = \alpha \cdot U(f) - \beta \cdot d(\mathbf{p}_i, f) \\ + \delta \cdot S(f, \mathcal{G}) + \rho_i(f) - E(f, \mathcal{G})\end{aligned} \tag{7}$$

where:

- $U(f) = |\{c \in B_3(f) : c \notin \mathcal{N}\}|$ is the number of unknown cells within a 3-cell radius (information gain);
- $d(\mathbf{p}_i, f)$ is the Manhattan distance (travel cost);
- $S(f, \mathcal{G}) = \min_{g \in \mathcal{G}} d(f, g)/15$ is a separation bonus that encourages agents to select frontiers far from other agents' announced goals, capped at 1.5;
- $\rho_i(f)$ is a role-specific bonus described below;
- $E(f, \mathcal{G})$ is a goal exclusion penalty described below.

We use $\alpha = 2$, $\beta = 0.5$, $\delta = 5$ in all experiments unless noted otherwise.

Goal exclusion. The exclusion penalty discourages agents from selecting frontiers that lie within an exclusion radius R of any other agent's announced goal:

$$E(f, \mathcal{G}) = \sum_{g \in \mathcal{G}} \max\big(0,\ 20 \cdot (1 - d(f, g)/R)\big) \tag{8}$$

where R is the exclusion radius (default $R = 8$). Frontiers at distance $d \geq R$ from all goals receive zero penalty; those closer receive a penalty that increases linearly up to a maximum of 20 at $d = 0$. This soft exclusion zone is the primary

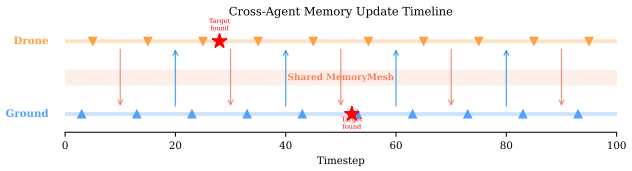

Figure 3. **Cross-agent memory update timeline.** Triangles indicate observation events; arrows show compact memory update transmissions (delta cells + goal coordinate) between the drone (top lane) and ground robot (bottom lane). Stars mark target discovery events. Each agent's local spatial memory map (center band) is updated continuously via peer-to-peer broadcasts.

coordination mechanism, producing a 25% reduction in path overlap relative to no exclusion (see Sec. 3.5).

Wall density. We define the wall density around a frontier f as:

$$W(f) = |\{c \in B_3(f) : n_c.\text{state} = \texttt{wall}\}| \tag{9}$$

counting the number of wall cells within a 3-cell radius of f.

Role-aware scoring. The role bonus biases each agent type toward frontiers matching its sensing capabilities:

$$\rho_i(f) = \begin{cases} 0.3 \cdot U(f) & \text{if } a_i = a_d \text{ (drone)} \\ 0.5 \cdot W(f) & \text{if } a_i = a_g \text{ (ground)} \end{cases} \tag{10}$$

The drone bonus scales with information gain $U(f)$, biasing it toward large open frontiers where its wide FOV is most effective. The ground robot bonus scales with wall density $W(f)$, biasing it toward room interiors where raycasting provides detailed inspection. These bonuses are intentionally mild ($\rho_d = 0.3$, $\rho_g = 0.5$) to avoid overriding the information gain and exclusion signals; ablations in Sec. 3.5 confirm that stronger role bonuses can hurt target discovery.

Persistent goal pursuit. To reduce frontier oscillation, each agent pursues its selected goal for T_{replan} timesteps before re-evaluating. At each replanning step, the agent re-scores all frontiers using Eq. (7) and selects the highest-scoring frontier as its new goal. Between replanning steps, the agent navigates toward its current goal via A* pathfinding. We use $T_{\text{replan}} = 3$ by default; Sec. 3.5 shows that replanning every step ($T_{\text{replan}} = 1$) increases overlap by 54% due to oscillation between competing frontiers.

Stale region re-visitation. If no frontiers remain but the spatial memory map contains cells not observed for more than $\tau_{\text{stale}} = 15$ timesteps, agents re-visit the closest stale cell, ensuring robustness to transient observations.

Table 1. **Main comparison on 80×80 maps** (mean, 5 seeds, 20 targets, 200 steps). Comm. in kilobytes. Best in **bold** (excl. Independent for comm.). Sep. = mean inter-agent distance; higher indicates better spatial distribution.

Method	Cov.	Tgt.	Disc.↓	KB↓	Ovlp.↓	Sep.↑
Independent	.954	19.0	65.4	**0**	.036	50.1
RawMapShare	.982	19.6	65.4	20,621	.019	53.8
DeltaMapShare	.982	19.6	65.4	99	.019	53.8
PeriodicSync	.975	19.4	68.9	1,924	.014	47.6
DeltaPeriodSync	.975	19.4	68.9	150	.014	47.6
Heur. Frontier	.845	17.4	78.5	2,503	.006	62.5
DeltaMapShareGoals	.986	**20.0**	**62.6**	100	.016	**59.4**
MemoryMesh	.972	19.2	67.1	101	**.012**	48.9

3. Experiments

3.1. Simulator and Setup

We use a 2D grid-world SAR simulator with 18% wall density via cellular automata plus rectangular rooms with doorways. We evaluate on two map sizes: an 80×80 grid with 20 hidden targets over 200 timesteps, and a 150×150 grid with 40 hidden targets over 300 timesteps. The drone observes a 17×17 square without wall occlusion; the ground robot observes a radius-5 circle with raycasting. Pathfinding uses A* with 8-connected movement. All results report means over 5 random seeds.

3.2. Baselines

We compare MemoryMesh against seven strategies: (1) **Independent:** greedy frontier selection, no communication; (2) **RawMapShare:** full map broadcast at every step; (3) **DeltaMapShare:** only new cells at every step (fairest comm. comparison); (4) **PeriodicSync:** full state every 10 steps [4]; (5) **DeltaPeriodicSync:** delta-encoded periodic sync; (6) **HeuristicFrontier:** shared frontiers with left/right zone assignment; (7) **DeltaMapShareGoals:** delta-encoded cells plus goal broadcasting with exclusion, but without role-aware scoring or persistent goals (an ablated variant of MemoryMesh that isolates the goal coordination mechanism).

The distinction between DeltaMapShareGoals and MemoryMesh is that the former uses only goal broadcasting with exclusion on top of delta-encoded sharing, while MemoryMesh additionally applies role-aware scoring (Eq. (10)) and persistent goal pursuit.

3.3. Main Results: 80×80 Maps

Tab. 1 summarizes results on the 80×80 environment.

DeltaMapShareGoals achieves the best target discovery. On 80×80 maps, the goal-broadcast variant DeltaMapShareGoals finds all 20 targets with the fastest mean discovery

Table 2. **Main comparison on 150×150 maps** (mean, 5 seeds, 40 targets, 300 steps). Same metrics as Tab. 1. Best in **bold**.

Method	Cov.	Tgt.	Disc.↓	KB↓	Ovlp.↓	Sep.↑
Independent	.625	24.8	132.4	**0**	.015	100.5
DeltaMapShare	.682	**27.4**	144.7	200	.003	105.6
Heur. Frontier	.507	18.6	149.7	7,191	**.000**	150.9
DeltaMapShareGoals	.695	25.4	**131.0**	205	**.000**	**128.1**
MemoryMesh	**.698**	25.6	136.8	208	.002	118.2

time (62.6 steps) and the highest coverage (0.986), using only 100 KB of communication. This demonstrates that goal broadcasting with exclusion is a powerful coordination mechanism even without role-aware scoring. MemoryMesh achieves the lowest path overlap (1.1%) among all methods, indicating that the combination of role-aware scoring and persistent goals produces the most spatially separated exploration trajectories. However, this comes at a slight cost to target discovery speed (67.1 vs. 62.6 steps) and total targets found (19.2 vs. 20.0), suggesting that the role bonus can occasionally steer agents away from productive frontiers in compact environments.

Communication efficiency. Both MemoryMesh (101 KB) and DeltaMapShareGoals (100 KB) achieve communication costs within 2% of DeltaMapShare (99 KB), confirming that goal broadcasting adds negligible overhead (8 bytes per message). All delta-encoded methods reduce bandwidth by >99.5% relative to RawMapShare (20,621 KB). HeuristicFrontier uses 25× more bandwidth (2,503 KB) while achieving the worst target discovery (17.4 targets, 78.5 steps).

Separation quality. DeltaMapShareGoals produces the best inter-agent separation (59.4 cells), a 19% improvement over Independent (50.1). MemoryMesh achieves moderate separation (48.9), as the role-aware scoring partially counteracts the separation bonus when the drone is biased toward specific open-area frontiers.

3.4. Main Results: 150×150 Maps

To test scalability, we evaluate on 150×150 maps with 40 targets over 300 timesteps (Tab. 2). This harder setting reveals clearer differences between strategies.

Coverage and coordination scale differently. On the larger maps, MemoryMesh achieves the highest coverage (0.698), marginally ahead of DeltaMapShareGoals (0.695). DeltaMapShare finds the most targets (27.4) but achieves this through aggressive independent exploration rather than coordination, as reflected in its higher overlap (0.003 vs. 0.000 for DeltaMapShareGoals) and lower separation (105.6

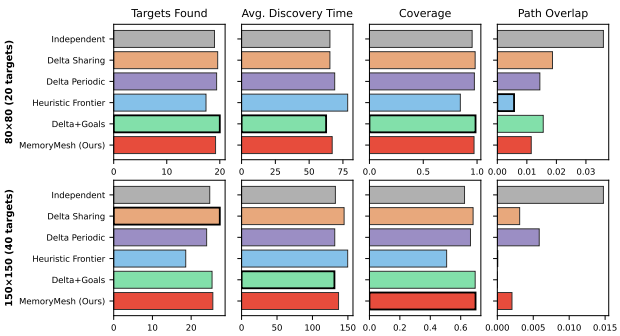

Figure 4. **Multi-metric summary** across all methods on 80×80 maps.

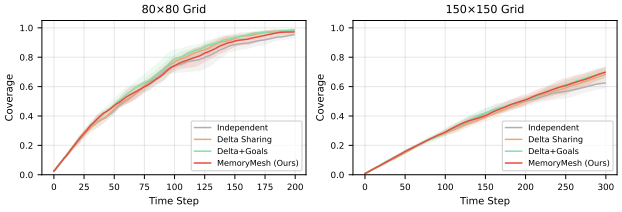

Figure 5. **Coverage over time** (mean ± 1 std, 5 seeds, 80×80). All methods except HeuristicFrontier converge to >95% coverage.

vs. 128.1). DeltaMapShareGoals again achieves the fastest discovery time (131.0 steps) and the best separation (128.1 cells), confirming that goal-based coordination is the dominant factor in exploration efficiency. HeuristicFrontier's rigid zone partition fails catastrophically at this scale, covering only 50.7% of the map and finding fewer than half the targets.

Communication remains efficient. On the larger maps, all delta-encoded methods use 200–208 KB, scaling linearly with the number of observed cells. HeuristicFrontier uses 7,191 KB (36× more) while achieving the worst task performance.

3.5. Ablation Studies

We conduct ablations on the 80×80 environment to isolate the contribution of each coordination mechanism.

Goal exclusion radius. Varying the exclusion radius R in Eq. (8) from 0 (no exclusion) to 20 cells reveals a clear trade-off between coordination strength and exploration flexibility. At $R = 0$, overlap is 1.5% with mean separation of 46.8 cells. At $R = 16$, overlap drops to 0.8% and separation increases to 57.3 cells, a 45% reduction in overlap and 22% improvement in separation (Fig. 7, left). The default $R = 8$ provides a balanced operating point between coordination and flexibility.

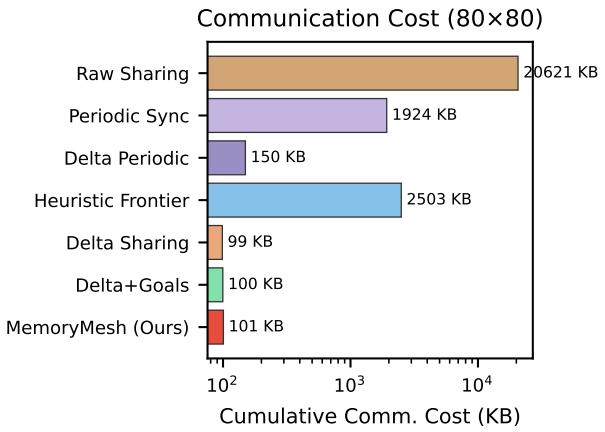

Figure 6. **Communication cost** (log scale). Delta-encoded methods (DeltaMapShare, DeltaMapShareGoals, MemoryMesh) cluster near 100 KB; goal broadcasting adds negligible overhead.

Table 3. **Component ablation** on 80×80 (5 seeds). Each row removes one component from the full system.

Variant	Tgt.	Disc.↓	Ovlp.↓	Sep.↑
Full (default)	19.2	67.1	.012	48.9
no_role	19.4	58.8	.011	61.2
no_sep	19.4	66.7	.016	54.7
no_excl (R=0)	19.4	70.7	.015	46.8
no_goals	19.4	69.1	.029	40.6

Replanning interval. The persistent goal mechanism (Sec. 2.4) reduces oscillation by committing agents to goals for T_{replan} steps. At $T_{\text{replan}} = 1$ (replanning every step), overlap is 1.8%. At $T_{\text{replan}} = 2$, overlap drops to 0.8%. At $T_{\text{replan}} = 10$, overlap is 0.7%, but responsiveness to new information decreases (Fig. 7, right). The default $T_{\text{replan}} = 3$ balances stability with adaptability.

Component ablation. Tab. 3 isolates the effect of each component by removing them individually from the full MemoryMesh configuration.

The results reveal a nuanced picture. Removing all goal-based coordination (no_goals: R=0, T_{replan}=1, no separation bonus) increases overlap by 154% (from 1.1% to 2.9%) and reduces separation by 17% (from 48.9 to 40.6), confirming that goal broadcasting is the primary coordination mechanism. Removing only goal exclusion (no_excl) increases overlap by 33% relative to the full system. Notably, removing the role bonus (no_role) *improves* discovery time from 67.1 to 58.8 steps and increases separation to 61.2, suggesting that role-aware scoring can be counterproductive when the environment is small enough for both agents to cover efficiently without specialization. This finding motivates the separate DeltaMapShareGoals baseline, which omits

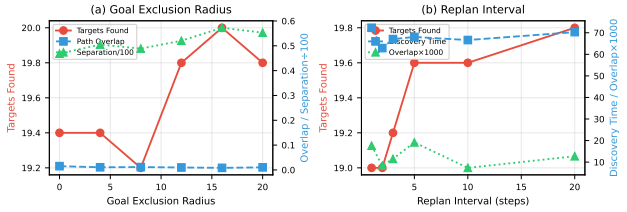

Figure 7. **Goal exclusion radius** (left) and **replanning interval** (right). Larger exclusion radius reduces overlap; persistent goals ($T_{\text{replan}} > 1$) reduce oscillation.

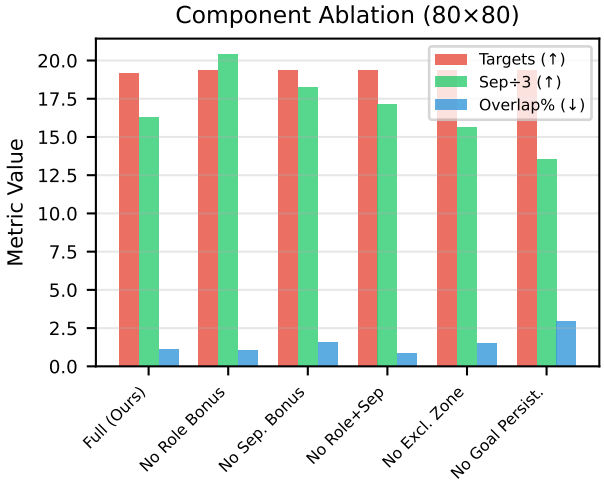

Figure 8. **Component ablation.** Removing goal coordination (no_goals) has the largest impact on overlap. Removing role bonus (no_role) improves discovery time, suggesting role-awareness has mixed effects on compact maps.

role-awareness and achieves the best overall performance on 80×80 maps.

3.6. Robustness Analysis

We evaluate three degradation modes on 80×80 maps (Fig. 9).

Sensor noise at 0/5/10/20% exercises the confidence-weighted merge. Target discovery degrades gradually: 19.2 targets at 0%, 18.8 at 5%, 17.4 at 10%, and 17.0 at 20%. The confidence-weighted conflict resolution (Eq. (2)) ensures that high-confidence observations overwrite noisy ones, limiting the impact on coordination quality.

UAV partial occlusion at 0/30/50/70% masks a fraction of the drone's behind-wall observations. Performance is remarkably stable: 19.2 targets at 0%, 19.6 at 30%, 20.0 at 50%, and 19.0 at 70%. The slight improvement at moderate occlusion levels (30–50%) occurs because the drone's

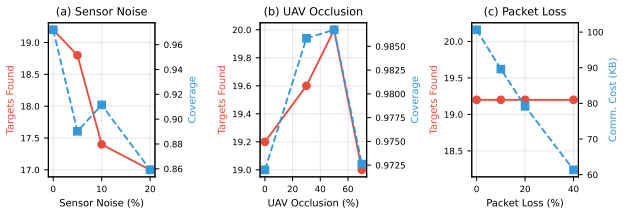

Figure 9. **Robustness:** sensor noise (left), UAV occlusion (center), packet loss (right). Packet loss has zero impact on target discovery across all tested rates.

reduced indoor coverage forces the ground robot to explore more room interiors, where it is more effective.

Packet loss at 0/10/20/40% has zero impact on targets found: 19.2 at all loss rates. Missed updates are re-observed and re-transmitted at subsequent steps; the delta-encoding mechanism naturally retransmits any cells that were lost. This robustness to packet loss is a direct consequence of the peer-to-peer architecture, where each agent's local map serves as a persistent backup.

3.7. Scaling and Qualitative Analysis

Agent scaling (2→4 agents) improves discovery from 67.1 to 48.7 steps while overlap grows from 1.1% to 3.7%. Map scaling (60×60 to 120×120) shows sublinear coverage decline under fixed step budgets (99.6% to 87.2%). Trajectory heatmaps confirm complementary exploration: the drone sweeps open areas while the ground robot concentrates on room interiors. Full scaling results, trajectory visualizations, scoring weight sensitivity, and per-step bandwidth analysis are in Sec. C.

4. Conclusion

We presented MemoryMesh, a shared spatial memory for ground-air cooperative search-and-rescue that combines delta-encoded map sharing with a goal-broadcast coordination protocol. By augmenting compact cell updates with a single goal coordinate per agent (8 bytes), MemoryMesh enables exclusion-based coordination at communication costs comparable to pure delta-encoded sharing (approximately 100 KB on 80×80 maps, a >99.5% reduction relative to full-map sharing).

Our experiments reveal that goal broadcasting with exclusion (the DeltaMapShareGoals variant) is the strongest coordination mechanism, finding all 20 targets on 80×80 maps with the fastest discovery time (62.6 steps) and the best inter-agent separation (59.4 cells). The full MemoryMesh system, which additionally incorporates role-aware scoring and persistent goal pursuit, achieves the lowest path overlap (1.1%) on 80×80 maps and the highest coverage (0.698) on the harder 150×150 environment. However, the role bonus provides mixed benefits: on compact maps it can steer agents away from productive frontiers, and the no_role ablation achieves faster discovery (58.8 vs. 67.1 steps). We report these findings transparently, as they suggest that role-aware scoring is most beneficial when environments are large enough to require genuine spatial specialization.

The confidence-weighted conflict resolution handles sensor noise gracefully, with target discovery degrading from 19.2 to 17.0 only at 20% noise. The system is robust to 40% packet loss with zero impact on target discovery, a direct consequence of the peer-to-peer architecture where each agent's local map serves as a persistent backup.

Limitations. Our evaluation uses a 2D grid-world simulator that simplifies real-world challenges such as continuous motion planning and 3D environments. The role bonus parameters ($\rho_d = 0.3$, $\rho_g = 0.5$) were tuned on 80×80 maps and may require adjustment for different environment scales, as the component ablation suggests these values can hurt target discovery on compact maps. The pairwise exclusion mechanism degrades with 4+ agents, as overlap increases from 1.1% (2 agents) to 3.7% (4 agents). The conflict resolution policy is a simple confidence-then-recency rule; more sophisticated Bayesian fusion could improve accuracy under heavy noise.

From perception to precognition. MemoryMesh currently treats the information gain $U(f)$ as a simple count of unknown cells. A natural next step toward predictive coordination is to reformulate $U(f)$ as a probabilistic measure of uncertainty—e.g., the expected entropy reduction over a learned occupancy or target distribution—so that broadcast goals encode *predicted* future observations rather than merely announced destinations. Combined with a short-horizon model of each agent's likely trajectory, this would let exclusion zones reflect anticipated coverage, moving the protocol from reactive map-merging toward genuinely precognitive coordination.

Future work. Natural extensions include: (1) scaling to 3D simulators such as Habitat [16] for embodied SAR evaluation; (2) learning the frontier scoring weights and exclusion radius via multi-agent reinforcement learning, potentially adapting role bonuses to environment characteristics; (3) extending the exclusion mechanism to handle larger teams through hierarchical territory negotiation; and (4) incorporating semantic object hypotheses into memory entries for richer scene understanding.

Broader impact. Efficient multi-robot coordination has direct applications in disaster response, reducing the time

to locate survivors. The goal-broadcast protocol's minimal bandwidth requirement (8 bytes per agent per step) makes it particularly suitable for scenarios with degraded communication infrastructure where coordination quality is critical but bandwidth is scarce.

References

[1] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[2] Matthew Chang, Théophile Gervet, Mukul Khanna, Sriram Yenamandra, Dhruv Shah, So Yeon Min, Kavit Shah, Chris Paxton, Saurabh Gupta, Dhruv Batra, and Roozbeh Mottaghi. GOAT: GO to any thing. In *arXiv preprint arXiv:2311.06430*, 2023.

[3] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural SLAM. In *Proc. Int. Conf. Learning Representations (ICLR)*, 2020.

[4] Micah Corah, Cormac O'Meadhra, Kshitij Goel, and Nathan Michael. Communication-efficient planning and mapping for multi-robot exploration in large environments. *IEEE Robotics and Automation Letters*, 4(2):1715–1721, 2019.

[5] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In *Proc. Int. Conf. Machine Learning (ICML)*, pages 1538–1546, 2019.

[6] M. Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proc. IEEE*, 94(7):1257–1270, 2006.

[7] Daniel S. Drew. Multi-agent systems for search and rescue applications. *Current Robotics Reports*, 2(2):189–200, 2021.

[8] Junhao Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, and Farshad Arvin. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69 (12):14413–14423, 2020.

[9] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning for multi-agent cooperation. In *Proc. Int. Conf. Learning Representations (ICLR)*, 2020.

[10] Woojun Kim, Myungsik Cho, and Youngchul Sung. Message-dropout: An efficient training method for multi-agent deep reinforcement learning. In *Proc. AAAI Conf. Artificial Intelligence*, pages 6079–6086, 2019.

[11] Yen-Cheng Liu, Junjiao Tian, Nathaniel Glaser, and Zsolt Kira. When2com: Multi-agent perception via communication graph grouping. In *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4106–4115, 2020.

[12] Robin R. Murphy. *Disaster Robotics*. MIT Press, 2014.

[13] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. *Proc. ACM Symp. User Interface Software and Technology (UIST)*, 2023.

[14] Jorge Peña Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker, Tuan Nguyen Gia, Hannu Tenhunen, Moncef Gabbouj, Jenni Raitoharju, and Tomi Westerlund. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *IEEE Access*, 8: 191617–191643, 2020.

[15] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-Web: Learning embodied object-search strategies from human demonstrations at scale. In *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5173–5183, 2022.

[16] 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 *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, pages 9339–9347, 2019.

[17] Yulun Tian, Yun Chang, Fernando Herrera Arias, Carlos Nieto-Granda, Jonathan P. How, and Luca Carlone. Kimera-Multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems. *IEEE Transactions on Robotics*, 38(4): 2022–2038, 2022.

[18] Ziluo Tian, Ying Wen, Zhiwei Gong, Faiz Punber, Shihao Zhang, and Jun Wang. Learning individually inferred communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 22069–22079, 2020.

[19] Endel Tulving. Episodic and semantic memory. In *Organization of Memory*, pages 381–403. Academic Press, 1972.

[20] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. IEEE Int. Symp. Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151. IEEE, 1997.

[21] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. *Proc. Int. Conf. Learning Representations (ICLR)*, 2024.

[22] Boyu Zhou, Yichen Zhang, Xinyi Chen, and Shaojie Shen. FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning. *IEEE Robotics and Automation Letters*, 6(2):779–786, 2021.

A. Additional Material

B. Related Work

Multi-robot exploration and coordination. Frontier-based exploration, introduced by Yamauchi [20], assigns robots to boundaries between known and unknown space. Subsequent work has extended this to multi-robot settings with utility-based frontier assignment [1], information-theoretic criteria [4], and deep reinforcement learning [8]. Hierarchical decomposition methods partition the environment into zones [22], while auction-based systems use market mechanisms for task allocation [6]. MemoryMesh differs by maintaining a persistent shared memory that captures observation recency, confidence, and observer identity, augmented with a goal-broadcast protocol that enables exclusion-based coordination.

Communication-constrained multi-agent systems. Several works address limited communication in multi-robot teams. Liu *et al*. [11] learn *when* and *what* to communicate for collaborative perception, while Jiang *et al*. [9] propose graph-based communication for multi-agent reinforcement learning. Kim *et al*. [10] drop low-value messages via scheduling mechanisms. Tian *et al*. [18] use gated communication in cooperative multi-agent tasks. These methods typically learn communication policies end-to-end, requiring substantial training data. MemoryMesh takes a complementary, structured approach: the memory map itself determines what is novel and worth sharing, requiring no learned communication policy.

Delta and submap communication. A common strategy for reducing bandwidth in multi-robot mapping is to transmit only newly observed or changed cells rather than full maps. Such delta-encoded approaches achieve large compression ratios relative to full-state sharing but provide no mechanism for coordinating future exploration. MemoryMesh adopts delta encoding for cell updates but additionally broadcasts each agent's current navigation goal, enabling exclusion-based coordination that pure delta methods lack. Our experiments include delta-encoded baselines to isolate the contribution of this goal-broadcast coordination layer.

Ground-air heterogeneous teams. UAV-UGV cooperation has been studied for surveillance [7], SLAM [17], and exploration [14]. Drones provide broad situational awareness while ground robots perform detailed inspection. Prior work typically assumes reliable full-bandwidth communication or focuses on mapping accuracy rather than communication efficiency. MemoryMesh explicitly models the communication channel through compact, goal-augmented memory updates that add only 8 bytes per agent per message.

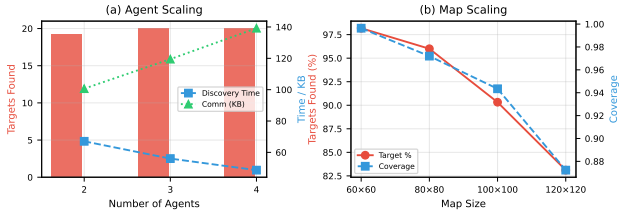

Figure 10. **Scaling:** agents (left) and map size (right). Discovery time improves with more agents but overlap grows; target percentage decreases with map area under fixed time budgets.

Shared memory in multi-agent AI. Memory-augmented multi-agent systems have appeared in embodied AI [5, 13] and cooperative language agents [21]. Scene graphs [2], topological maps [3], and spatial memories [15] provide structured representations for navigation. MemoryMesh extends these ideas to the multi-agent SAR setting with an explicit focus on memory *exchange* protocols, goal-broadcast coordination, conflict resolution under partial observability, and decentralized peer-to-peer synchronization.

C. Supplementary Material

C.1. Scaling Analysis

Agent scaling. Increasing from 2 to 3 to 4 agents (adding ground robots) improves both targets found and discovery speed: 19.2 targets in 67.1 steps (2 agents), 20.0 in 56.0 steps (3 agents), 20.0 in 48.7 steps (4 agents). Coverage grows from 97.2% to 99.2% to 99.8%, confirming that MemoryMesh scales effectively with additional agents. Path overlap increases from 1.1% to 2.9% to 3.7%, as the pairwise exclusion mechanism becomes less effective with more agents competing for frontier space. Communication grows sublinearly: 101 KB (2 agents), 119 KB (3), 139 KB (4).

Map scaling. On progressively larger maps (all using 200 steps except 150×150 which uses 300 steps): 60×60 achieves 99.6% coverage with 10.8/11 targets, 80×80 achieves 97.2% coverage with 19.2/20 targets, 100×100 achieves 94.3% with 28.0/31 targets, and 120×120 achieves 87.2% with 37.4/45 targets. Coverage scales sublinearly with map area, as the fixed step budget becomes increasingly constraining. The 150×150 results in Tab. 2 (with extended 300-step budget) confirm that MemoryMesh maintains its coordination advantages at larger scales.

C.2. Qualitative Analysis and Failure Modes

Fig. 11 shows a representative run: the drone sweeps broad areas while the ground robot concentrates on room interiors, confirming complementary exploration patterns induced by the role-aware scoring. Two failure modes persist: (1) targets deep inside narrow-entrance rooms delay discovery, as the

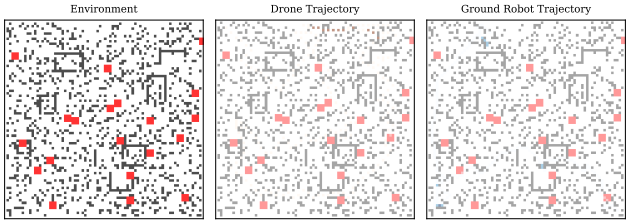

Figure 11. **Trajectory heatmaps.** Left: environment. Center: drone (orange). Right: ground robot (blue). Goal exclusion zones produce spatially separated exploration patterns.

Table 4. **Scoring weight sensitivity** on 80×80 (5 seeds). "Low" and "High" denote halved and doubled values for the named weight.

Variant	Tgt.	Disc.↓	Ovlp.↓	Sep.↑
Default	19.2	67.1	.011	48.9
Low info (α=1)	19.6	63.7	.012	55.9
High info (α=4)	19.8	67.1	.015	47.2
Low dist (β=0.25)	18.8	61.1	.015	62.0
High dist (β=1)	20.0	62.8	.013	54.6
Low sep (δ=2.5)	19.4	63.0	.022	49.5
High sep (δ=10)	20.0	61.7	.006	57.4

ground robot must navigate through doorways (accounting for variance across seeds); (2) on compact maps (80×80), the role bonus can steer the drone away from productive frontiers near walls, explaining why the no_role variant achieves faster discovery in the component ablation (Tab. 3).

C.3. Scoring Weight Sensitivity

We vary each scoring weight individually while keeping others at default values (α=2, β=0.5, δ=5, ρ_d=0.3, ρ_g=0.5) on the 80×80 environment.

The system is moderately sensitive to the separation weight δ: halving it roughly doubles overlap (from 1.1% to 2.2%), while doubling it nearly halves overlap (to 0.6%) and improves separation from 48.9 to 57.4 cells. The distance weight β has a strong effect on separation: low distance penalty (β=0.25) increases separation to 62.0 cells but reduces targets found. The information gain weight α has moderate effects. Overall, the system is robust to 2× parameter perturbations across all weights.

C.4. Per-Step Bandwidth Analysis

Tab. 5 reports per-step communication statistics for each method on 80×80 maps (200 steps).

MemoryMesh and DeltaMapShareGoals consume approximately 0.5 KB/step, well within typical wireless link budgets even in degraded environments. The goal coordinate adds only 8 bytes per message (4% overhead over DeltaMapShare's per-step cost). At 150×150, per-step cost increases to approximately 0.69 KB/step (208 KB / 300 steps), remaining practical for low-bandwidth radio links.

Table 5. **Per-step bandwidth** on 80×80. Total bandwidth divided by 200 steps.

Method	Total KB	KB/step
Independent	0	0
RawMapShare	20,621	103.1
DeltaMapShare	99	0.49
PeriodicSync	1,924	9.6
DeltaPeriodicSync	150	0.75
Heur. Frontier	2,503	12.5
DeltaMapShareGoals	100	0.50
MemoryMesh	101	0.50

Table 6. **Full comparison on 150×150** (5 seeds, 40 targets, 300 steps).

Method	Cov.	Tgt.	Disc.↓	KB↓	Ovlp.↓	Sep.↑
Independent	.625	24.8	132.4	0	.015	100.5
RawMapShare	.682	27.4	144.7	61,247	.003	105.6
DeltaMapShare	.682	27.4	144.7	200	.003	105.6
PeriodicSync	.664	24.0	131.4	5,982	.006	100.5
DeltaPeriodSync	.664	24.0	131.4	354	.006	100.5
Heur. Frontier	.507	18.6	149.7	7,191	.000	150.9
DeltaMapShareGoals	.695	25.4	131.0	205	.000	128.1
MemoryMesh	**.698**	25.6	136.8	208	.002	118.2

C.5. Full 150×150 Comparison

Tab. 6 extends Tab. 2 with all seven baselines on 150×150 maps.