QUEST: Q-Learning for Uncertainty-Guided Efficient Search Teams

Arsh Verma **Tejus Gupta** **Jeff Schneider**
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213
{arshv,tejusg,jeff4}@cs.cmu.edu

Abstract

Robot teams searching large outdoor areas must make online decisions about where each robot should sense next, with the goal of locating all objects in as little time as possible. Searching efficiently requires the robots to spread across different regions and to weigh long-term benefit over immediate gain, for example clearing a small unsearched area nearby before departing for a distant part of the map. Each decision commits a robot to a multi-step route, and poor waypoint choices waste budget: a robot revisits ground that it or a teammate has already searched, or heads early for a remote region, spending travel on the way and leaving nearby gaps that cost a return trip and erode overall coverage. We present QUEST, a decentralized graph Q-learning framework in which each robot makes independent, asynchronous decisions based on the shared belief about the environment, its remaining budget, and information about its teammates' positions and chosen routes, if available. QUEST learns the long-term value of each waypoint choice with graph-attention Q-networks trained on a standard Bellman objective, and one interface serves single robots, homogeneous teams, and heterogeneous UAV–UGV teams. In simulation, our results show that its advantage over baselines grows as more robots contribute to searching the same unsearched ground. With four ground robots it reaches an F-score of 0.910, versus 0.875 for the strongest learned baseline over $1{,}000$ episodes. This widening lead remains consistent across three independent training seeds and persists under communication outages and robot failures never seen in training. We further show that the same formulation supports a single heterogeneous policy that transfers to team compositions it never trained on, finding objects earlier with 10% less travel and 12% less redundant coverage. The gains are largest where map topology, crowding, and a tight budget make these long-term costs matter.

1 Introduction

Robot teams are increasingly deployed to search areas too large or dangerous for people to inspect: for survivors and hazards after a disaster, for wildlife and damaged infrastructure in environmental surveys, and for sparse targets such as gas leaks in public-safety missions [41, 2, 16, 34, 31, 7]. In each case the team must locate the objects as quickly as possible, with each robot deciding where to sense next as evidence arrives, under limits on travel distance and mission time.

Active search formalizes this decision problem: an agent adaptively chooses where to observe, based on its posterior over object locations, so as to find as many objects as possible within a fixed budget [15, 26, 17]. Unlike coverage-based planners, which sweep an area regardless of what is observed, active search adapts each decision to the evidence gathered so far. We study its decentralized multi-robot form, in which robots with noisy sensors choose waypoints or sensing tiles online rather than follow tours fixed before launch. We model it as shared-belief search on a cell graph (Fig. 2): each cell carries a posterior probability of containing an object, which the robots update through their observations. In this setting, each choice commits a robot to a *route* along which it senses and spends path budget, so its value depends both on the information gathered en route and on where the route

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

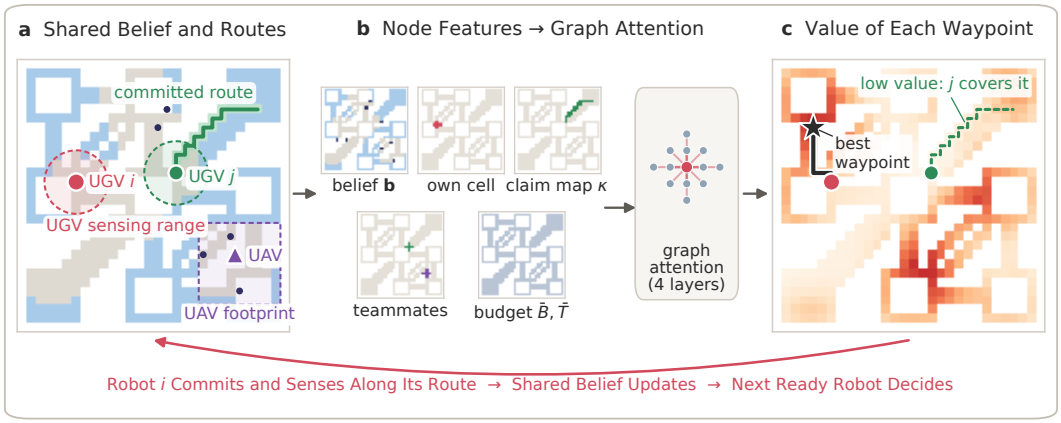


Figure 1: **QUEST scores a waypoint by its long-term consequences for the team, not just by what the route sees.** (a) Robots share one belief over the map (blue roads: not yet searched; grey: searched; dots: detected objects). UGV i (red) is ready to choose, while teammate UGV j (green) drives its committed route and a UAV (purple) senses a square footprint. (b) Robot i's inputs become node features on the cell graph, and a graph-attention network scores every feasible waypoint. (c) Darker red marks higher value. Waypoints along j's route score low, and i commits to finishing the nearby dead-end room ($\star$), which would otherwise cost a return trip through its narrow entrance later.

leaves the robot. A route that is informative now can end deep in a dead end or consume the budget needed to reach the remaining regions. On open maps and budget-rich settings such mistakes are easily recovered from. However, in constrained terrain such as forests, road networks, and collapsed buildings, early route choices determine which regions remain reachable later.

Teams make search more powerful, but also more delicate. Robots can partition the search area, avoid repeated effort, and exploit complementary platforms: UAVs sweep large regions quickly with broad but noisy sensing, while UGVs make slower, local, high-confidence observations. These benefits, however, require the robots to act as a coordinated team rather than as independent searchers. Without coordination, robots converge on the same promising region, revisit ground already searched by teammates, or leave difficult regions unresolved, even though each decision looks reasonable in isolation. This coordination must also be decentralized and asynchronous, since robots complete their routes at different times and cannot wait for one another. A good waypoint choice must therefore account not only for the robot's own future, but also for the routes its teammates have already chosen.

Existing robotic active-search methods fall short of this requirement in one or both of two ways: they plan myopically, or they offer limited support for team coordination. Greedy and path-integral planners score a route by the information it collects [16, 2, 4, 3, 18], and learned policies amortize expert planners through imitation and inherit their horizon [22, 30, 18]. Informative path planners optimize over longer horizons but target a single robot [20, 5, 34]. As a result, these approaches do not account for the long-term consequences of a route for the team.

To capture these consequences, we propose learning the value of the *post-route state*: the updated belief, together with the robot's position and remaining budget and its teammates' positions and chosen routes, at the moment a route ends. We introduce QUEST (Q-learning for Uncertainty-guided Efficient Search Teams), a decentralized framework in which each robot uses a learned estimate of this value to select waypoints independently and asynchronously, with teammates' chosen routes encoded as a *claim map* over the cell graph (Fig. 1). Training uses the standard one-step Bellman target; the novelty lies in the formulation, in which actions are variable-duration routes, robots act asynchronously over a shared belief, and teammates' routes enter each robot's state, under one decentralized interface for all platforms. Because the policy acts locally on graph nodes, the same weights transfer to new maps and to unseen team compositions.

Contributions.

1. **Formulation.** Multi-robot active search cast as asynchronous, variable-duration route decisions over a shared belief, with one decentralized interface for single robots, homogeneous teams, and heterogeneous UAV–UGV teams (§3).

2. **Learning post-route value.** A decentralized graph Q-learning method that conditions each waypoint's value on teammates' intent through a claim map, credits each robot with the reward its route earned while teammates kept acting, treats budget exhaustion as truncation rather than termination, and uses type-specific value heads for heterogeneous teams (§4).
3. **Gains that grow with team size.** The lead over the strongest learned baseline more than doubles from two to four UGVs (1,000 episodes per team size), consistently across three training seeds, and zeroing the claim map at test time degrades performance increasingly with team size; a single robot, in turn, learns to finish a region before leaving it (§5.2–§5.3).
4. **Generalization without retraining.** The same policies transfer to unseen heterogeneous team compositions, communication outages, robot failures, and a map of a real field site (§5.4–§5.6). All results are in simulation.

2 Related Work

Robotic active search. Bayesian active search and adaptive sensing choose queries by expected information gain or submodular surrogates [15, 26, 17], and informative path planning extends this to robots that sense along trajectories [20, 5, 34]. Multi-robot active search adds realistic noise and decentralized decisions: NATS and GUTS use Thompson sampling with depth-aware noise [16, 2], CAST and CDAS account for travel cost [4, 3], aerial teams have been fielded [41], and RL has been applied to multi-agent active search [21]. We keep the shared-posterior interface of these planners and use them as baselines and teachers.

Amortized active-search policies. The closest prior work amortizes expert planners into neural policies trained by imitation, often on a graph over the map or the robots [22, 30, 18, 43]. QUEST keeps this scalable representation, and the path-integral expert as a baseline, but learns the value of the post-route state instead of imitating a route score.

Multi-agent RL and communication. In MARL terms, QUEST's team policy is parameter-shared independent Q-learning [42, 10] with intent-augmented observations: robots of one type share a Q-network and act on their own observations, with no centralized critic or value mixer. Centralized training coordinates through critics, value decomposition, counterfactual credit, or teacher–student guidance [28, 35, 39, 12, 46, 6, 27, 47], and learned communication optimizes messages, recipients, or schedules [38, 11, 23, 36, 9, 25, 13, 48]. QUEST instead broadcasts what search teams already share: the posterior, teammate positions, and committed routes. Multi-agent imitation and offline MARL [37, 29, 32, 49, 24] learn from fixed behavior, whereas QUEST learns online.

Temporal abstraction and air–ground teams. Each waypoint is a temporally extended action, as in options and macro-actions [40, 1, 19]; we price its duration in the reward and discount per decision (§4.3). Multi-robot search systems, including air–ground teams, often assign explicit roles or hierarchies [31, 33, 7]. QUEST keeps the team flat, with type-specific value heads, and lets the division of labor be learned.

3 Problem Setup

Search region and belief. The search region is a square occupancy grid $\mathcal{M}$ (for the controlled maps, 30×30 cells of 30 m; Fig. 3). The *searchable cells* $\mathcal{S}$ are those some platform can observe, including cells a UGV cannot enter. Each searchable cell c independently contains an object, $\theta_c \sim \text{Bernoulli}(\rho)$. All robots share one belief $\mathbf{b} \in [0,1]^{\mathcal{S}}$ of posterior object probabilities. Observing cell c yields a noisy $o_c \in [0,1]$ whose noise σ depends on platform and range (App. A.1), and $\mathbf{b}_c$ is updated by exact Bayes' rule. Uncertainty is the total entropy $\mathcal{H}(\mathbf{b}) = \sum_c h(\mathbf{b}_c)$, with h the Bernoulli entropy.

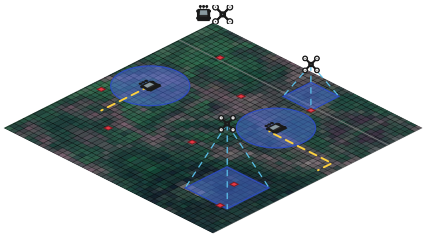


Figure 2: **Heterogeneous search on a shared grid.** UGVs sense a neighborhood of every cell on their routes (yellow); UAVs sense square tiles whose size depends on altitude. Red: hidden objects.

Platforms. A *UGV* chooses a traversable waypoint,
drives the shortest path to it on the four-connected travel graph G_{trav} of traversable cells (one budget unit per cell), and senses every cell within three cells of each cell it visits, with noise $\sigma \in \{0.05, 0.20, 0.45\}$ rising with range. A *UAV* chooses a sensing tile (i, j, k): an $f_k \times f_k$ tile with $f_k \in \{2, 4, 8\}$ at altitude $k \in \{0, 1, 2\}$ and noise $\sigma \in \{0.15, 0.25, 0.35\}$. It flies over obstacles at three cells per tick, pays Manhattan distance, and senses along its flight. A close-range UGV observation takes a cell from prior 0.1 to posterior 0.9 in about one observation, while the widest UAV footprint needs about ten (App. A.3).

Teams and budgets. We study a single UGV, homogeneous teams of M UGVs, and heterogeneous teams of U UGVs and V UAVs. Each episode has a path budget B (per team, or per platform type in heterogeneous teams) and a cap T on the number of waypoint decisions. An episode ends when a budget or the decision cap is exhausted, or when the search is *settled*: every cell for a single robot, and 95% of searchable cells for teams, has posterior below 0.05 or above 0.9.

Decisions and information. Robots act at *decision epochs*. At an epoch, a ready robot chooses an action from its action set $\mathcal{A}$ (waypoints or tiles), and the simulator executes the route over several ticks. All robots sense while moving, and every observation updates the shared belief immediately. The next epoch occurs when any robot finishes its route, so execution is event-driven and asynchronous. A ready robot observes the belief $\mathbf{b}$, its own position and remaining budget, and each teammate's position and *committed route* (the part of its chosen route not yet driven), and acts on this alone, with no central planner. The result is a decentralized decision process over beliefs whose actions have variable duration (options, 40); we index a robot's own decisions by n.

Objective. We score the prediction $\hat{\theta}_c = \mathbf{1}[\mathbf{b}_c > 0.9]$ by its F-score against θ and report it against consumed (team) path length, so a curve that rises sooner means objects found with less travel; tables use fixed checkpoints on this axis.

4 Method

QUEST scores each waypoint by the learned value of its post-route state. We describe the graph state and claim map (§4.1), the asynchronous reward (§4.2), and the learning target (§4.3); Algorithm 1 gives the training loop.

4.1 Graph state and claim map

Node features. Every searchable cell c is a node. For a single UGV at cell p, $\mathbf{x}_c = [\mathbf{b}_c,\ \mathbf{1}[c = p],\ \beta,\ \tau,\ \bar{B},\ \bar{T}]$, where β, τ are the consumed fractions of the path budget and decision cap and $\bar{B}, \bar{T}$ the normalized remaining budgets. Budgets are randomized in training, so the policy must read them to act well under any budget. Team policies add the number of teammates at c, the number whose current waypoint is c, and the claim κ_c.

Claim map. Teammate positions alone do not reveal where teammates are about to sense. The claim map does: it marks every cell on a teammate's committed route,

$$\kappa_c = \mathbf{1}\big[\exists\, j \neq i :\ c \text{ lies on the unexecuted part of robot } j\text{'s route}\big], \tag{1}$$

where i is the acting robot; for a UAV teammate the flight path is projected onto the grid. Each robot thus sees intended coverage, not just current positions. A binary claim avoids a decay constant tuned to budget and team size. The claim map informs coordination; it does not enforce it. Routes are not interrupted, and robots that become ready at the same tick choose from the same claim map, so simultaneous choices can still overlap.

Network. The policy is a four-layer graph attention network [45] (eight heads of width 64, layer normalization) on a message-passing graph G_{mp} that links each searchable cell to every searchable cell within Manhattan distance 9 (7 for policies with UAVs), with edge feature $1/(d_M + 1)$ (App. B.1). A head reads out one Q-value per action in the platform's action set. The heterogeneous policy shares the backbone but uses type-specific input projections and UGV and UAV Q-heads. Because the network is local and teammate channels are counts, the same weights run on new maps and on team compositions absent from training.

4.2 Reward

The reward pays for what the team actually learned: it follows the realized posterior, not a predicted gain. For a single robot, let $\Delta\mathcal{H}_n$ be the entropy reduction produced along the route chosen at decision n and $\ell_n > 0$ the route's length in path-budget units. The reward is

$$r_n = \Delta\mathcal{H}_n\Big(1 + \frac{\eta}{\ell_n}\Big) - c_\ell\,\ell_n + \Big[\Phi_n - \max_{m<n}\Phi_m\Big]^+. \tag{2}$$

The first term rewards information and, through η, information per unit of travel; c_ℓ prices travel. Entropy reduction drives cells toward posterior 0 or 1, and the F-score (§3) counts the cells above 0.9, so the reward pays for the resolution the metric depends on. A zero-length action (the current cell, or an unreachable target) receives $r_n = -c_r$ instead, which keeps the agent from idling. The last term pays a completion bonus for resolving the belief early:

$$\Phi_n = R_0\left[\mathrm{clip}\Big(\frac{u_n - u^\star}{1 - u^\star}, 0, 1\Big)\right]^\alpha (1 - \beta_n), \tag{3}$$

where u_n is the fraction of resolved cells ($\mathbf{b}_c \notin [0.1, 0.9]$). Φ is zero below $u^\star = 0.5$, grows quadratically ($\alpha = 2$) above it, and shrinks as budget is spent. Paying only increases over its running maximum caps the total bonus at the peak of Φ (at most R_0), so oscillation cannot farm it.

Team reward. In a team, a route's observations arrive over several decision epochs while teammates act, so its reward accumulates until the route ends. Call the time between two consecutive decision epochs an *interval* e. When robot i commits to a route, its running reward starts at $A^i = 0$; after each interval e the route spans, it is updated as

$$A^i \leftarrow w_e^i\big(A^i + \Delta\mathcal{H}_e^i - c_\ell\,\ell_e^i\big) + \frac{\eta}{N}\,\frac{\Delta\mathcal{H}_e}{\ell_e} + \frac{1}{N}\Big[\Phi_e - \max_{e'<e}\Phi_{e'}\Big]^+, \tag{4}$$

and r_n is the value of A^i when the route ends. Here $\Delta\mathcal{H}_e^i$ and ℓ_e^i are the entropy reduction from robot i's observations and its distance, $\Delta\mathcal{H}_e$ and ℓ_e the team totals (the middle term is zero if $\ell_e = 0$), and N the number of robots; for $N = 1$ a route spans one interval and this reduces to Eq. 2.

The individual term ties each robot to the true posterior: re-sensing cells a teammate has resolved earns only the small entropy drop it produces, so coordination has to be learned from the observations rather than supplied by a hand-designed team reward. Collisions are soft: $w_e^i = 0.5$ if UGV i ends the interval on another UGV's cell, halving the reward accumulated on the route so far, and $w_e^i = 1$ otherwise; robots do not block each other, and UAVs are exempt. In teams a zero-length action earns nothing but uses up a decision. Coefficients are in App. B.2.

4.3 Learning target

One transition $(s_n, a_n, r_n, s_{n+1}, d_n)$ is stored per completed route, where s_{n+1} is the robot's observation at its next decision. We use the one-step Bellman target with Double-DQN action selection [44] and clipped double-Q evaluation [14]. With online networks q_1, q_2 and target networks $\bar{q}_1, \bar{q}_2$,

$$y_n = r_n + \gamma\,(1 - d_n)\min_{j\in\{1,2\}}\bar{q}_j\big(s_{n+1}, a^\star\big), \qquad a^\star = \arg\max_{a\in\mathcal{A}(s_{n+1})}\min_{j\in\{1,2\}} q_j(s_{n+1}, a), \tag{5}$$

and both online networks minimize $\sum_j \big(q_j(s_n, a_n) - y_n\big)^2$ over a mini-batch. Here $d_n = 1$ only if the search is settled (§3). Exhausting a budget is a *truncation*: we still bootstrap, so values are not pulled to zero at an arbitrary horizon. Target networks are Polyak-averaged (rate 0.005). We discount once per decision ($\gamma = 0.95$) and price route duration in the reward; a semi-Markov target was unstable with randomized budgets (App. B.5).

Parameter sharing and heterogeneity. Robots of one type share a Q-network and a replay buffer and act on their own observations at test time. In heterogeneous teams, UGV and UAV transitions from the same asynchronous stream are sampled into per-type mini-batches, and each head is trained with Eq. 5 over its own action set.

5 Experiments

We test whether learned post-route value pays off as teams grow (§5.2), for a single robot (§5.3), in mixed fleets (§5.4), and under stress (§5.5–§5.6).

Algorithm 1 Asynchronous training of QUEST

Require: Q-networks q_1, q_2 and targets $\bar{q}_1, \bar{q}_2$ (one pair per platform type); replay buffer $\mathcal{D}$; W parallel workers

for each worker, in parallel **do**
  Reset: sample a map instance, objects, team size or composition, and budget multipliers $\sim \mathrm{Unif}(0.7, 1.3)$
  loop
    Advance the simulator tick by tick until some robot i is ready; all robots sense and update $\mathbf{b}$
    Update every robot's running reward A^j with this interval (Eq. 4)
    Build robot i's observation s: belief, own state, teammates' positions, claim map (Eq. 1)
    if robot i has a pending decision (s_i, a_i) **then**
      Add (s_i, a_i, A^i, s, d) to $\mathcal{D}$; reset $A^i \leftarrow 0$
    $a \leftarrow \epsilon$-greedy over feasible actions of $\min_j q_j(s, \cdot)$; commit robot i to the route to a
    $(s_i, a_i) \leftarrow (s, a)$; **if** the episode ended **then** flush pending decisions and reset

Learner: after every 256 new transitions, take gradient steps on mini-batches from $\mathcal{D}$ (per platform type), sampling each transition about 10 times on average; use target Eq. 5 and Polyak-update $\bar{q}_j$

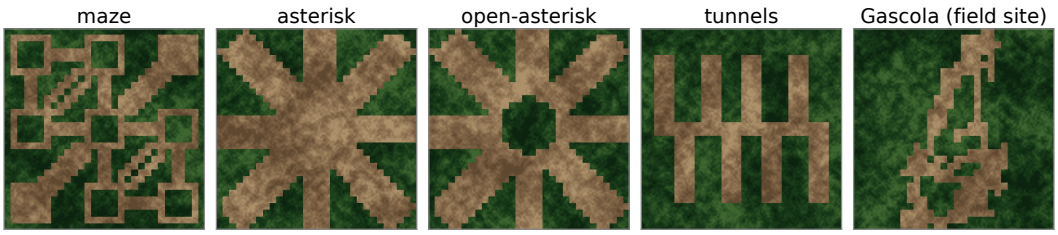


Figure 3: **The evaluation maps span constrained and open topologies.** The rightmost map is digitized from the Gascola field site. Brown: traversable by UGVs; searchable cells add a one-cell rim that UGVs can see but not enter.

5.1 Setup

Maps. The four controlled 30×30 maps vary topology (Fig. 3). The *constrained* maps are `maze`, a 3×3 grid of rooms joined by narrow bridges, and `tunnels`, a spine with eight dead-end branches. The *open* controls are `asterisk`, eight arms around a wide core, and `open-asterisk`, whose hollow core joins the arms through parallel detours. The Gascola map is digitized from the forested field site of about $75{,}000\,\mathrm{m}^2$ where we previously deployed active-search policies on an autonomous ground vehicle [18]; only its geometry is used, and objects, sensing, and motion are simulated as on the other maps.

Protocols. Every QUEST policy (QUEST-SR for a single UGV, QUEST-MR for UGV teams, QUEST-Het for UAV–UGV teams) is trained on `maze` alone, with object density $\rho = 0.1$ and budgets randomized by $\pm 30\%$. Evaluation uses sparser objects ($\rho = 0.03$), which makes the search longer and harder, and tighter budgets (App. A.2), so every result tests generalization beyond training. All methods see identical episodes. The BC baselines were trained on the Gascola map, so they too transfer zero-shot to every controlled map; the online planners are recomputed on every map. Single-UGV and UGV-team policies are trained with three seeds (figures show the first; App. C.1 reports all three); the heterogeneous policy is a single run. Team results pool 1,000 episodes per team size in 20 fixed blocks of 50; the costly online planners and the stress tests use the first block, so baseline values there differ slightly (App. A.2). Error bars and $\pm$ values are one standard error over episodes; p-values are two-sided Welch t-tests unless noted.

Baselines. All methods share belief updates, action sets, and feasibility masks. *Random* picks a feasible action uniformly. Greedy-EIG picks the action with the highest one-step expected information gain (EIG); Path-EIG accumulates EIG along the shortest path to each endpoint, with a length penalty [18]. Greedy-NATS-MR and Path-NATS-MR are team versions using the NATS score [16], with each robot replanning on the shared belief. BC-GreedyEIG and BC-PathEIG amortize Greedy-EIG and Path-EIG by imitation [18], on the QUEST backbone; in teams every robot runs the single-robot BC policy. BC-PathEIG is the strongest learned baseline in every team setting. Single-policy-as-MR and Single-policy-as-Het run QUEST-SR (and a single-UAV policy for UAVs) on every robot, with the shared belief but no teammate inputs.

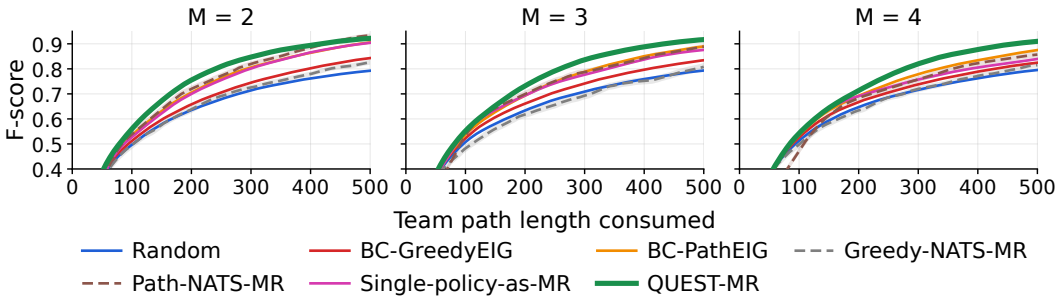


Figure 4: **As the team grows, baselines lose ground and QUEST-MR does not.** F-score versus team path length on `maze` for $M = 2, 3, 4$ UGVs. Solid: 1,000 episodes per team size; dashed (online planners): 50 episodes; bands ± 1 s.e.

	$M = 2$	$M = 3$	$M = 4$
BC-PathEIG	0.906	0.889	0.875
Single-policy-as-MR	0.904	0.876	0.839
QUEST-MR	**0.922**	**0.917**	**0.910**
Lead over BC-PathEIG	+0.016 (19/20)	+0.027 (20/20)	+0.035 (20/20)
Lead over Single-policy-as-MR	+0.018 (20/20)	+0.041 (20/20)	+0.071 (20/20)

Table 1: **The lead of QUEST-MR grows with team size.** F-score at team path length 500 on `maze`, 1,000 episodes per cell (s.e. ≤ 0.005; Random and BC-GreedyEIG score 0.79–0.84). Leads are block means; parentheses count the 20 blocks with a positive lead.

5.2 Team contention: the lead grows with team size

As more robots share a map, their routes compete for the same regions; if QUEST values the post-route state, including teammates' routes, its advantage should grow with the team. We train QUEST-MR on `maze` with team size drawn from $\{2, 3, 4\}$ in each episode and evaluate it at $M = 2, 3, 4$ with a shared team budget of 500.

QUEST-MR leads BC-PathEIG at every team size, and the lead more than doubles with the team: from $+0.016$ at $M = 2$ to $+0.035$ at $M = 4$, where it is positive in all 20 blocks (Fig. 4, Table 1; paired growth $+0.019 \pm 0.003$, $p < 0.001$). The baselines lose ground as the team grows, while QUEST-MR holds between 0.910 and 0.922. The online planner shows the same pattern: Path-NATS-MR beats QUEST-MR with two robots (0.934 vs. 0.915 on the first block) but trails by 0.056 with four. Mid-mission the lead is larger still: $+0.050$ at path length 200 for $M = 2$. The growth also holds for every training seed: the lead over BC-PathEIG is positive at every team size and grows from $M = 2$ to $M = 4$ by $+0.046$, $+0.025$, and $+0.030$ (App. C.1).

The policies rely on teammates' routes. Zeroing only the claim-map input at test time, with the same weights and episodes, costs QUEST-MR 0.039–0.060 F-score at every $M = 4$ checkpoint ($p < 0.001$); at 75% of the budget the cost is larger for four robots than for two ($+0.045 \pm 0.016$, $p = 0.004$; App. C.3).

Policies that ignore teammates' routes send several robots after the same region, and do so more often as the team grows. QUEST-MR reads teammates' routes, so it holds its F-score. Team size also changes each robot's budget share and decision rate, so the growth reflects contention together with these effects, not contention alone.

5.3 Single robot: where does looking ahead help?

BC-PathEIG already scores information along a whole route, so post-route value should add to it where a route can leave a region half-searched or force a return trip through a bottleneck, and should show up as objects found sooner. We train QUEST-SR on `maze` and evaluate it zero-shot on the other three maps (50 episodes per map, budget 300).

On the constrained training map, QUEST-SR leads BC-PathEIG at every checkpoint for all three training seeds (path length 150–300; 12/12 cells, $+0.013$ to $+0.034$; App. C.1). On the unseen maps it finds objects sooner: at path length 150 it leads for every seed on `tunnels` and `open-asterisk`,

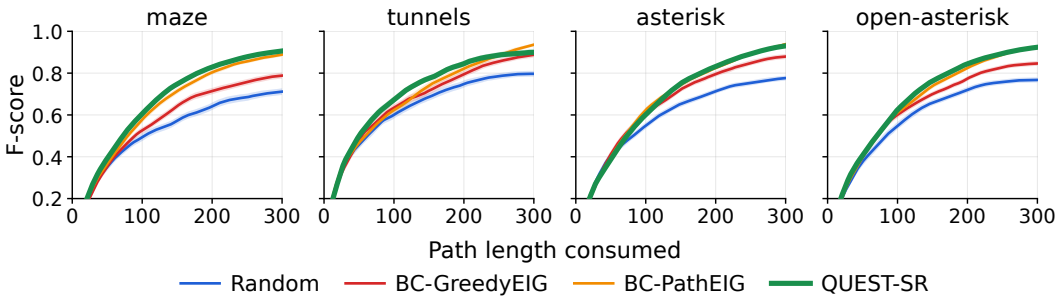


Figure 5: **A single UGV running QUEST-SR finds objects sooner.** F-score versus path length on the training map (`maze`) and three unseen maps; 50 episodes per map, bands ± 1 s.e. QUEST-SR leads throughout on `maze` and early on `tunnels` and `open-asterisk`; BC-PathEIG catches up as the budget runs out.

	Region resolved before exit (↑)			Re-entries per episode (↓)		
Map	QUEST-SR	BC-PathEIG	p	QUEST-SR	BC-PathEIG	p
`maze`	**0.843**	0.740	< 0.001	**2.12**	2.76	< 0.001
`tunnels`	**0.856**	0.824	0.004	**2.80**	4.18	< 0.001
`asterisk`	**0.858**	0.785	< 0.001	2.94	3.24	0.12
`open-asterisk`	**0.864**	0.797	< 0.001	3.28	**2.88**	0.039

Table 2: **QUEST-SR finishes a region before it leaves.** *Resolved before exit*: fraction of a region's cells with posterior ≤ 0.1 or ≥ 0.9 when the robot leaves it; regions are fixed graph communities of each map. *Re-entries*: returns to a region already left. Means over 50 episodes per policy and map; bold marks the better policy at $p < 0.05$ (s.e. in App. C.4).

and matches BC-PathEIG on `asterisk`. As the budget runs out, BC-PathEIG catches up on the open maps and finishes ahead on `tunnels` (Fig. 5).

The learned policy also searches differently. On every map, QUEST-SR resolves more of a region before leaving it ($+0.03$ to $+0.10$, all $p \leq 0.004$; Table 2). This pays where returning is expensive: on the constrained maps it also re-enters regions less often (-0.64 on `maze`, -1.38 on `tunnels`), because every return means crossing a bottleneck again. On the open maps a departed region is cheap to revisit, so completion buys little.

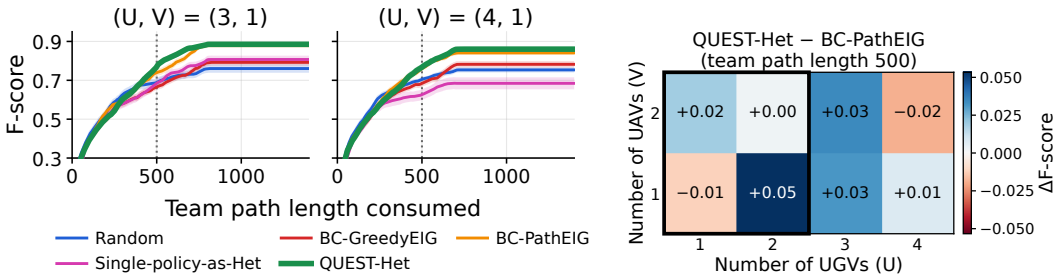
An early lead is the signature of a better searcher: given enough budget, any competent policy eventually resolves a 30×30 map, so curves converge as the budget runs out. The same pattern holds throughout the paper: the team lead peaks mid-mission at every team size (§5.2), and the mixed-fleet gain is an early-budget gain (§5.4).

5.4 Heterogeneous UAV–UGV teams

We next ask whether one policy can combine UAVs and UGVs under a binding budget, including on team compositions it never trained on. We train QUEST-Het on `maze` with $(U, V) \in \{1, 2\}^2$ (U UGVs, V UAVs) and evaluate it on all 16 compositions with $U, V \leq 4$ (Fig. 6; 25 episodes each; budgets 400 for UGVs and 1000 for UAVs). We report F-score at team path length 500, about a third of the combined budget, where most compositions are still budget-limited.

On the unseen teams of one UAV with three or four UGVs, QUEST-Het has the highest F-score at path length 500 (Fig. 6a); across the eight compositions with one or two UAVs, it leads or ties BC-PathEIG on six (Fig. 6b). Pooled over one UAV with two to four UGVs, it reaches 0.759 versus 0.728 ($p < 0.05$) while travelling 10% less with 12% less redundant coverage (both $p < 0.001$; Table 3). Once budgets saturate, BC-PathEIG catches up and leads on 14 of 16 compositions (App. C.5).

The mixed-fleet gain is coordination while the budget binds, and it requires team-level structure: Single-policy-as-Het travels as little as QUEST-Het but reaches only 0.667, because it stops early and cannot share the work.



(a) Two unseen compositions. QUEST-Het pulls ahead mid-budget (dotted line) before BC-PathEIG catches up.

(b) Gain over BC-PathEIG at path length 500. Black borders: training compositions.

Figure 6: **One heterogeneous policy coordinates team compositions it never trained on.** QUEST-Het is trained with $U, V \in \{1, 2\}$ (U UGVs, V UAVs) and evaluated without fine-tuning on `maze`; 25 episodes per composition, bands ± 1 s.e.

Method	F-score (↑)	Path length (↓)	Redundant cells (↓)
Random	0.676 ± 0.011	835 ± 14	0.328 ± 0.007
BC-GreedyEIG	0.665 ± 0.010	810 ± 18	$\mathbf{0.253 \pm 0.005}$
BC-PathEIG	0.728 ± 0.010	818 ± 13	0.298 ± 0.005
Single-policy-as-Het	0.667 ± 0.013	$\mathbf{726 \pm 30}$	0.272 ± 0.010
QUEST-Het	$\mathbf{0.759 \pm 0.010}^{*}$	$734 \pm 17^{**}$	$0.261 \pm 0.009^{**}$

Table 3: **QUEST-Het finds more objects with less travel and less overlap.** One UAV with two, three, or four UGVs on `maze`, 75 episodes per row. F-score at team path length 500; path length and redundant cells over the whole episode. Bold: best per column; $^{*}p < 0.05$, $^{**}p < 0.001$ versus BC-PathEIG.

Policy	Duplicate coverage (↓)	UAV wide	UAV narrow	Total path (↓)
Random	0.383	6%	71%	1227
BC-GreedyEIG	0.387	60%	22%	1097
BC-PathEIG	0.391	19%	71%	1033
Single-policy-as-Het	0.455	84%	16%	1082
QUEST-Het	**0.122**	39%	61%	**715**

Table 4: **QUEST-Het learns a division of labor without assigned roles.** Two UGVs and one UAV on `maze`, 25 episodes. UAV wide / narrow: share of the UAV's actions at the 8×8 / 2×2 footprint (the rest use 4×4). Duplicate coverage: fraction of sensed cells observed by more than one robot.

On two-UGV–one-UAV teams (Table 4), the gain comes from a learned division of labor: each single-agent policy's UAV favors one footprint, while QUEST-Het's UAV both screens (39% of its actions at the widest footprint) and confirms (61% at the narrowest). No role is assigned, yet QUEST-Het cuts duplicate coverage, the fraction of sensed cells observed by more than one robot, by 69% and total path by 31% relative to BC-PathEIG (App. C.6).

5.5 Robustness to outages and failures

In the field, links drop and robots break down. We test whether the lead survives such conditions, absent from training, with the four-UGV team on the first 50-episode block of §5.2. In an *outage*, three of the four robots stop sharing observations at a random time in the first 25% of the team budget, for up to 25% or 50% of the budget; a disconnected robot senses into a private belief that is merged back on reconnection. In a *failure*, one or two robots stop permanently at a random time before a given fraction of the team budget.

QUEST-MR leads in every condition (Table 5): by $+0.051$ and $+0.034$ under outages and by $+0.031$ to $+0.050$ under failures. Under failures its F-score stays within 0.907–0.915, the narrowest range of any method. The longest outage costs it 0.023 (it alone reads teammates' routes), yet it still leads by 0.034. Mixed fleets adapt as well: when a UGV fails, QUEST-Het's UAV shifts ten points of its actions (Table 4) from wide-area screening to close-range confirmation, the lost UGV's role (App. C.10). Across a sweep of outage start and duration, the lead stays positive in

		Outage (max. length)		One failure (before budget fraction)				Two failures
Method	Clean	25%	50%	0.3	0.5	0.6	1.0	1.0
Random	0.791	0.804	0.804	0.787	0.784	0.798	0.807	0.763
BC-GreedyEIG	0.815	0.811	0.808	0.819	0.826	0.829	0.823	0.819
BC-PathEIG	0.853	0.861	0.856	0.862	0.877	0.871	0.869	0.860
QUEST-MR	**0.913**	**0.912**	**0.890**	**0.912**	**0.908**	**0.915**	**0.914**	**0.907**
Lead over BC-PathEIG	+0.060	+0.051	+0.034	+0.050	+0.031	+0.044	+0.045	+0.047

Table 5: **QUEST-MR keeps its lead under outages and failures it never trained on.** Four UGVs on `maze`; F-score near the end of the team budget (path length 500; 450–495 when failures remove budget, App. C.8). 50 episodes per cell; s.e. ≤ 0.013. Clean: the first block of Table 1.

Comparison	What differs from QUEST	Setting	ΔF
BC-PathEIG	imitates a route score; no teammate inputs	$M = 4$	+0.035
		one UGV, `maze`	+0.013 to +0.034
Path-NATS-MR	online planner; no learning	$M = 2 \rightarrow 4$	$-0.019 \rightarrow +0.056$
Single-policy-as-MR	no teammate inputs; trained alone	$M = 4$	+0.071
Claim zeroed†	claim input removed at test time	$M = 4$	+0.039 to +0.060
Single-policy-as-Het	no type heads or teammate inputs	$(U, V) = (3, 1)$	+0.086

Table 6: **What each comparison isolates.** ΔF: QUEST minus the comparison at the setting shown (§5.2–§5.4); heterogeneous row at team path length 500. †Same policy and episodes, with and without its claim input.

every setting (+0.027 to +0.061; App. C.9). Outages make teammates' routes and beliefs stale and raise QUEST-MR's overlap between robots, yet under the longest outage it still overlaps less (0.511) than BC-PathEIG does with full communication (0.535); an outage here delays sharing rather than partitioning the team.

5.6 Transfer to a field-site map

The Gascola map is open terrain without rooms or dead-end branches (Fig. 3), and it is the map the BC baselines were trained on. Transferred zero-shot from `maze`, QUEST-MR ties BC-PathEIG with four UGVs (0.937 vs. 0.936), while QUEST-SR and QUEST-Het trail by 0.025 and 0.034 (App. C.7), as the open controlled maps predict.

6 Discussion and Conclusion

We have shown that robot teams search better when each decision is valued by the search state it leaves behind rather than by the route it drives. Learned this way, one policy family coordinates single robots, homogeneous teams, and air–ground fleets from information teams already share, and its advantage grows as robots contend for the same ground. The comparisons in Table 6 also mark where imitation suffices: on open maps and late in the budget, the best route and the best resulting state coincide. This suggests a simple rule for practitioners: imitate a planner when routes are easy to undo, and learn post-route value when terrain, teammates, or a binding budget make each commitment hard to reverse. Broadcasting the posterior and each robot's committed route was enough for a division of labor to emerge without learned messages, a centralized critic, or assigned roles, an interface any search team can adopt.

The next step is to carry QUEST from simulation to the field, and because every decision reads the shared posterior, each stage extends the formulation rather than redesigning it. Training across maps, densities, and team sizes, with a sparse $O(|\mathcal{S}|r^2)$ graph for wider areas (App. D.1), would replace one policy per map (App. C.11) with a general search policy. A campaign with one UAV and one UGV at our ground-vehicle site would then test the learned failure recovery on hardware. In the field, a language model could turn reports such as "survivors were last seen near the river" into belief updates, and a posterior over moving targets would take the formulation from search to tracking.

Acknowledgments and Disclosure of Funding

This work was supported by the U.S. Army Research Office and the U.S. Army Futures Command under Contract No. W911NF-20-D-0002.

References

[1] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2017.

[2] Nikhil Angad Bakshi, Tejus Gupta, Ramina Ghods, and Jeff Schneider. GUTS: Generalized Uncertainty-Aware Thompson Sampling for Multi-Agent Active Search. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[3] Arundhati Banerjee and Jeff Schneider. Cost-aware diffusion active search, 2026. arXiv preprint arXiv:2602.19538.

[4] Arundhati Banerjee, Ramina Ghods, and Jeff Schneider. Cost-awareness in multi-agent active search. In *European Conference on Artificial Intelligence (ECAI)*, 2023.

[5] Jonathan Binney and Gaurav S. Sukhatme. Branch and bound for informative path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[6] Yiqun Chen, Hangyu Mao, Jiaxin Mao, Shiguang Wu, Tianle Zhang, Bin Zhang, Wei Yang, and Hongxing Chang. PTDE: Personalized training with distilled execution for multi-agent reinforcement learning. In *International Joint Conference on Artificial Intelligence*, pages 31–39, 2024.

[7] Timothy H Chung, Geoffrey A Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics: A survey. *Autonomous robots*, 31(4):299–316, 2011.

[8] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[9] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In *International Conference on Machine Learning (ICML)*, 2019.

[10] Christian S. de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the StarCraft multi-agent challenge?, 2020. arXiv preprint arXiv:2011.09533.

[11] Jakob Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[12] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[13] Benjamin Freed, Ryan James, Guillaume Sartoretti, and Howie Choset. Sparse discrete communication learning for multi-agent cooperation through backpropagation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[14] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

[15] Roman Garnett, Yamuna Krishnamurthy, Xuehan Xiong, Jeff Schneider, and Richard Mann. Bayesian optimal active search and surveying. In *International Conference on Machine Learning (ICML)*, 2012.

[16] Ramina Ghods, William J. Durkin, and Jeff Schneider. Multi-agent active search using realistic depth-aware noise model. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[17] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

[18] Tejus Gupta, Arsh Verma, Raymond Song, David Guttendorf, Conor Igoe, Luis Navarro-Serment, and Jeff Schneider. Efficient active search via amortized path-integral policies. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

[19] Ruijie He, Emma Brunskill, and Nicholas Roy. PUMA: Planning under uncertainty with macro-actions. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2010.

[20] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014.

[21] Conor Igoe, Ramina Ghods, and Jeff Schneider. Multi-agent active search: A reinforcement learning approach. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):754–761, 2021.

[22] Conor Igoe, Tejus Gupta, and Jeff Schneider. Efficient bayesian experiment design with equivariant networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[23] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[24] Woojun Kim and Katia Sycara. B3C: A minimalist approach to offline multi-agent reinforcement learning, 2025. arXiv preprint arXiv:2501.18138.

[25] Woojun Kim, Myungsik Cho, and Youngchul Sung. Message-dropout: An efficient training method for multi-agent deep reinforcement learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2019.

[26] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research (JMLR)*, 9:235–284, 2008.

[27] Yueheng Li, Guangming Xie, and Zongqing Lu. Multi-agent guided policy optimization. In *International Conference on Learning Representations (ICLR)*, 2026.

[28] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[29] Connor Mattson, Varun Raveendra, Ellen Novoseller, Nicholas Waytowich, Vernon J. Lawhern, and Daniel S. Brown. R2BC: Multi-agent imitation learning from single-agent demonstrations, 2025. arXiv preprint arXiv:2510.18085.

[30] Quan Nguyen, Anindya Sarkar, and Roman Garnett. Amortized nonmyopic active search via deep imitation learning. *arXiv preprint arXiv:2405.15031*, 2024.

[31] Nikhil Nigam and Ilan Kroo. Persistent surveillance using multiple unmanned air vehicles. In *2008 IEEE Aerospace Conference*. IEEE, 2008.

[32] Ling Pan, Longbo Huang, Tengyu Ma, and Huazhe Xu. Plan better amid conservatism: Offline multi-agent reinforcement learning with actor rectification. In *International Conference on Machine Learning (ICML)*, 2022.

[33] Lynne E Parker, Daniela Rus, and Gaurav S Sukhatme. Multiple mobile robot systems. In *Springer Handbook of Robotics*, pages 1335–1384. Springer, 2016.

[34] Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Jen Jen Chung, Inkyu Sa, Roland Siegwart, and Juan Nieto. An informative path planning framework for uav-based terrain monitoring. *Autonomous Robots*, 44(6):889–911, 2020.

[35] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

[36] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *International Conference on Learning Representations (ICLR)*, 2019.

[37] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[38] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[39] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2018.

[40] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2): 181–211, 1999.

[41] Wennie Tabib, John Stecklein, Caleb McDowell, Kshitij Goel, Felix Jonathan, Abhishek Rathod, Meghan Kokoski, Edsel Burkholder, Brian Wallace, Luis Ernesto Navarro-Serment, et al. Decentralized Uncertainty-Aware Active Search with a Team of Aerial Robots. *arXiv preprint arXiv:2410.08507*, 2024.

[42] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaak Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE*, 2017.

[43] Mariliza Tzes, Nikolaos Bousias, Evangelos Chatzipantazis, and George J Pappas. Graph neural networks for multi-robot active information acquisition. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[44] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2016.

[45] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[46] Jian Zhao, Xunhan Hu, Mingyu Yang, Wengang Zhou, Jiangcheng Zhu, and Houqiang Li. CTDS: Centralized teacher with decentralized student for multi-agent reinforcement learning. *Transactions on Games*, 16(1):140–150, 2022.

[47] Yihe Zhou, Shunyu Liu, Yunpeng Qing, Tongya Zheng, Kaixuan Chen, Jie Song, and Mingli Song. CADP: Towards better centralized learning for decentralized execution in MARL. In *International Joint Conference on Artificial Intelligence*, 2025.

[48] Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems*, 38(4), 2024.

[49] Zhengbang Zhu, Minghuan Liu, Liyuan Mao, Bingyi Kang, Minkai Xu, Yong Yu, Stefano Ermon, and Weinan Zhang. MADiff: Offline multi-agent learning with diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

	Single UGV	Single UAV	UGV team	UAV–UGV team
Grid	30×30	32×32	30×30	32×32
Sensor noise σ	$0.05/0.20/0.45$	$0.15/0.25/0.35$	$0.05/0.20/0.45$	both
Footprint	range ≤ 3	tiles $2/4/8$	range ≤ 3	both
Speed (cells per tick)	1	3	1	1 / 3
Decision cap T	35	60	35 (team)	80 (team)
Path budget B	600	1200	600 (team)	400 (UGVs), 1200 (UAVs)
Team size	—	—	$M \in \{2, 3, 4\}$	$(U, V) \in \{1, 2\}^2$
Termination (settled)	all cells	all cells	95% of cells	95% of cells
Efficiency coefficient η	2	4	4 (team term)	4 (team term)
Step cost c_ℓ	0.02	0.01	0.02	0.01 (UGV), 0.02 (UAV)
Zero-length action reward	-1	-1	0	0
Completion bonus R_0	50	80	50	80
Resolved margin m	0.1	0.15	0.1	0.1
Collision weight (UGVs)	—	—	0.5	0.5

Table 7: **Training environments.** All settings train on `maze` with object density $\rho = 0.1$, multiply T and B by independent factors from $\mathrm{Unif}(0.7, 1.3)$ in each episode, and use $\gamma = 0.95$, $u^\star = 0.5$, and $\alpha = 2$. Evaluation settings are in Table 8. The single UAV is used only inside Single-policy-as-Het.

Appendix

- **App. A, Setting:** environments, sensor model, protocols, sensor operating points, and evaluation conventions.
- **App. B, Method details:** network and node features, reward details, training, baselines, the semi-Markov variant, and hyperparameters.
- **App. C, Additional results:** seed replicates, the 1,000-episode team evaluation, the claim-map intervention, trajectory diagnostics, heterogeneous grids and roles, the field-site map, robustness, and tuning sweeps.
- **App. D, Compute and broader impact.**

A Setting

A.1 Environments and sensing

Searchable cells and padding. The searchable region dilates the UGV-traversable mask by one cell, adding cells a UGV can see from an adjacent road but cannot occupy. Environments with a UAV use a 32×32 grid. The extra border is a passive pad with $\mathbf{b} = 0$ that lets the largest UAV tile (8×8) align with the grid; it is never sensed or scored.

Observation model. Each robot–cell observation has a noise level $\sigma \in [0, 1]$ set by the platform and geometry. The observation $o_c \in [0, 1]$ follows a truncated exponential likelihood that moves smoothly from perfect ($\sigma = 0$) to uninformative ($\sigma = 1$) measurements. With $\zeta_\sigma = \tan\big(\frac{\pi}{2}(1 - \sigma)\big)$,

$$p(o_c \mid \theta_c, \sigma) = \frac{\zeta_\sigma}{e^{\zeta_\sigma} - 1} \exp\big(\zeta_\sigma(1 - \nu_c)\big), \qquad \nu_c = \begin{cases} o_c & \theta_c = 0, \\ 1 - o_c & \theta_c = 1. \end{cases} \tag{6}$$

The likelihood ratio is $p(o_c \mid 1, \sigma)/p(o_c \mid 0, \sigma) = \exp\big(\zeta_\sigma(2o_c - 1)\big)$, so the exact Bayes update is

$$\mathbf{b}'_c = \frac{\mathbf{b}_c\, e^{\zeta_\sigma(2o_c - 1)}}{\mathbf{b}_c\, e^{\zeta_\sigma(2o_c - 1)} + 1 - \mathbf{b}_c}. \tag{7}$$

Entropy and expected information gain. Uncertainty is the total Bernoulli entropy $\mathcal{H}(\mathbf{b}) = -\sum_{c \in \mathcal{S}} \big(\mathbf{b}_c \log \mathbf{b}_c + (1 - \mathbf{b}_c) \log(1 - \mathbf{b}_c)\big)$ with $0 \log 0 = 0$. The planners use the expected information gain $\mathrm{EIG}(\mathbf{b}, a) = \mathcal{H}(\mathbf{b}) - \mathbb{E}_{o \sim p(\cdot \mid \mathbf{b}, a)}\big[\mathcal{H}(\mathbf{b}'(o))\big]$, where $\mathbf{b}'(o)$ is the posterior after the observations o of action a. The RL reward uses the realized reduction $\mathcal{H}(\mathbf{b}) - \mathcal{H}(\mathbf{b}')$ instead.

UGV sensing. At every cell p it visits, a UGV observes all cells within Euclidean distance 3 (physical range 90 m). The noise level depends on the distance $d_E(p, c)$: $\sigma_c = 0.05$ for $d_E \leq 1$, 0.20 for $1 < d_E \leq 2$, and 0.45 for $2 < d_E \leq 3$. At reset, each robot starts at a uniformly random traversable cell and receives one free observation before its first decision.

UAV actions and motion. A UAV action (i, j, k) selects the non-overlapping $f_k \times f_k$ tile with top-left corner (i, j) at altitude $k \in \{0, 1, 2\}$, with $f = (2, 4, 8)$ and noise $(0.15, 0.25, 0.35)$. The 32×32 grid has 256, 64, and 16 tiles at the three altitudes. Only tiles whose center is searchable are feasible. The UAV flies axis-aligned (first x, then y) at 3 cells per tick. A move from tile center (x, y) at altitude k to (x', y') at k' costs $|x - x'| + |y - y'| + |k - k'|$ path-budget units. During flight the camera observes the $f_{k'} \times f_{k'}$ footprint around each traversed cell, and no cell is observed twice in one flight.

Budgets and termination. Table 7 lists training budgets and Table 8 evaluation budgets. The heterogeneous setting has separate UGV and UAV team budgets. An episode is truncated when a path budget or the decision cap is exhausted. It terminates when the search is *settled*: every cell for a single robot, and for teams 95% of searchable cells with $\mathbf{b}_c < 0.05$ or $\mathbf{b}_c > 0.9$.

A.2 Training and evaluation protocols

Table 8 compares the training and evaluation protocol of every setting. Evaluation episodes are fixed per episode index, so every method sees the same object placements and start positions.

	Single UGV	UGV team	UAV–UGV team
QUEST training map	`maze`	`maze`	`maze`
BC training map	Gascola	Gascola (per robot)	Gascola (UGVs)
Team size: train → eval	—	$M \in \{2,3,4\} \to \{2,3,4\}$	$(U, V) \in \{1,2\}^2 \to \{1,..,4\}^2$
Object density: train → eval	$0.1 \to 0.03$	$0.1 \to 0.03$	$0.1 \to 0.03$
Decision cap T: train → eval	$35 \to 12$ or 15	$35 \to 35$	$80 \to 60$
Path budget B: train → eval	$600 \to 300$	$600 \to 500$	$(400, 1200) \to (400, 1000)$
Evaluation episodes per cell	50	1,000 (50 for planners)	25
Training seeds	3	3	1

Table 8: **Training and evaluation protocols.** Team budgets are shared by the team; heterogeneous budgets are $(B_{\mathrm{UGV}}, B_{\mathrm{UAV}})$ per platform type. Training multiplies T and B by independent factors drawn from $\mathrm{Unif}(0.7, 1.3)$ in every episode. Single-UGV evaluation uses $T = 12$ on `maze` and `open-asterisk` and $T = 15$ on `asterisk` and `tunnels`. The Gascola evaluation uses budgets scaled to the map (App. C.7).

A.3 Sensor operating points and platform complementarity

For a single observation of an occupied cell, the likelihood of Eq. 6 is proportional to $e^{\zeta o}$ on $[0, 1]$. Hence $\mathrm{TPR} = P(o > 0.5 \mid \theta = 1) = (e^{\zeta} - e^{\zeta/2})/(e^{\zeta} - 1)$, and by symmetry $\mathrm{FPR} = 1 - \mathrm{TPR}$. The mean observation is $\mathbb{E}[o \mid \theta = 1] = 1/(1 - e^{-\zeta}) - 1/\zeta$, which gives $\mathbb{E}[\mathrm{LLR}] = \zeta\,(2\,\mathbb{E}[o \mid \theta = 1] - 1)$. Moving the log-odds from $\ln(0.1/0.9)$ to $\ln(0.9/0.1)$ requires a total LLR of $\ln 81 \approx 4.39$. Table 9 evaluates these quantities for every noise level.

The table shows why the platforms complement each other. A close-range UGV observation resolves a cell in about one observation, but it covers only four cells and the UGV must drive there. The widest UAV footprint covers 64 cells at once and collects about $64 \times 0.43 \approx 27$ nats per observation while flying over obstacles at three times the UGV's speed, but each cell needs about ten observations. UAVs are built to screen, UGVs to confirm. Neither role is assigned by hand; the heterogeneous policy has to discover the split. The noise levels follow the operating values of our prior field deployment on this task [18]; they are not calibrated to a specific camera or range sensor.

A.4 Evaluation metrics and statistics

F-score. A cell is predicted positive if and only if $\mathbf{b}_c > 0.9$. Counting true positives (TP), false positives (FP), and false negatives (FN) over searchable cells, $F = 2\mathrm{TP}/(2\mathrm{TP} + \mathrm{FP} + \mathrm{FN})$, with

Platform	Mode	σ	Cells per obs.	TPR	FPR	$\mathbb{E}[\mathrm{LLR}]$	Obs. to 0.9
UGV	range ≤ 1	0.05	4	0.998	0.002	10.71	0.4
UGV	range $(1, 2]$	0.20	8	0.823	0.177	1.37	3.2
UGV	range $(2, 3]$	0.45	16	0.642	0.358	0.22	19.7
UAV	2×2 tile	0.15	4	0.889	0.111	2.30	1.9
UAV	4×4 tile	0.25	16	0.770	0.230	0.89	4.9
UAV	8×8 tile	0.35	64	0.693	0.307	0.43	10.3

Table 9: **Operating points of the sensor model.** TPR and FPR threshold a single observation at $o = 0.5$. $\mathbb{E}[\mathrm{LLR}]$ is the expected log-likelihood ratio of one observation of an occupied cell. *Obs. to* 0.9 is $\ln 81/\mathbb{E}[\mathrm{LLR}]$, the expected number of observations that move an occupied cell from prior 0.1 to posterior 0.9.

$F = 0$ when the denominator is zero. Curves index F-score by consumed path length (the team total in team settings). After an episode ends, its final F-score is carried forward to later checkpoints.

Checkpoints. Single-robot tables use path length 150, 200, 250, and 300 (the end of the evaluation budget). UGV-team tables use team path length 500 (the end of the team budget) unless stated otherwise. Heterogeneous tables use team path length 500, summing UGV and UAV movement cost, or the end of the budget where stated. Robot-failure columns use slightly earlier checkpoints (App. C.8).

Statistics. Error bars and $\pm$ values are one standard error over evaluation episodes. Unless stated otherwise, p-values are two-sided Welch t-tests on per-episode F-scores at the stated checkpoint. Paired tests are used where two conditions share episodes (claim-map masking, block-level differences).

B Method details

B.1 Graph attention network and node features

Attention layer. Each of the $L = 4$ layers computes, for each of $H = 8$ heads,

$$h'_v = \sum_{u \in \mathcal{N}(v)} \alpha_{uv}\, \Theta h_u, \qquad \alpha_{uv} = \mathrm{softmax}_{u \in \mathcal{N}(v)} \Big(\mathrm{LeakyReLU}\big(\mathbf{a}^\top [\Theta h_v \,\|\, \Theta h_u \,\|\, \Theta_e w_{uv}]\big) \Big), \tag{8}$$

with edge weight $w_{uv} = 1/(d_M(u, v) + 1)$. The heads are concatenated and followed by ReLU and layer normalization. The neighborhood $\mathcal{N}(v)$ contains v itself and every searchable u with $w_{uv} \geq 0.10$ (0.125 for policies that include UAVs).

Node features. Let p be the acting robot's cell, β and τ the consumed fractions of path budget and decision cap, and $\bar{B} = \max(0, B - \ell_{\mathrm{used}})/1000$ and $\bar{T} = \max(0, T - n_{\mathrm{used}})/100$ the remaining budgets in absolute units. The single-UGV features are $\mathbf{x}_c = [\mathbf{b}_c,\, \mathbf{1}[c = p],\, \beta,\, \tau,\, \bar{B},\, \bar{T}]$. The single-UAV features are $\mathbf{x}_c = [\mathbf{b}_c,\, d_M(c, p),\, k/2,\, \beta,\, \tau,\, \bar{B},\, \bar{T},\, \mathbf{1}[c = p \wedge k = 0],\, \mathbf{1}[c = p \wedge k = 1],\, \mathbf{1}[c = p \wedge k = 2]]$, where $k \in \{0, 1, 2\}$ is the UAV's current altitude. The UGV-team features are

$$\mathbf{x}_c = \Big[\mathbf{b}_c,\ \mathbf{1}[c = p],\ \kappa_c,\ \textstyle\sum_{j \neq i} \mathbf{1}[c = p_j],\ \sum_{j \neq i} \mathbf{1}[c = g_j],\ \beta,\ \tau,\ \bar{B},\ \bar{T}\Big], \tag{9}$$

where p_j and g_j are teammate j's position and current waypoint and κ_c is the claim map (Eq. 1). All teammate channels are sums over teammates, so the input size does not depend on the team size, and the features do not depend on teammate order.

Heterogeneous policy. The heterogeneous policy uses one shared attention backbone. A type-specific linear projection maps each platform's features into it, and the policy reads out a UGV Q-head and a UAV Q-head. The UGV features add counts of other UGVs and of UAVs at each cell and a platform-type bit. The UAV features mirror them and add the altitude indicators. The belief, claim map, and per-type budget channels are shared by both.

B.2 Reward details

Table 7 lists all reward coefficients. Three details complete Eqs. 2–4. First, single-robot UAV path length ℓ_n is the flight cost of App. A.1. Second, in heterogeneous teams the step cost is type-specific ($c_\ell = 0.01$ per UGV cell, 0.02 per UAV unit), and the team distance ℓ_e sums UGV cells and UAV movement cost. The completion potential uses the average of the UGV and UAV remaining-budget fractions as $1 - \beta$. Third, the resolved fraction u counts cells with $\mathbf{b}_c \notin [m, 1-m]$, with $m = 0.1$ ($m = 0.15$ for the single UAV, whose noisier sensor otherwise makes the threshold hard to reach). We use a resolved-cell criterion rather than an entropy fraction because altitude-tiered UAV noise leaves every cell with a residual entropy that an entropy threshold cannot get below.

B.3 Training implementation

Collection and updates. Sixteen parallel workers collect episodes into one replay buffer (capacity 10^6 transitions). A transition is one completed decision, of one robot in team settings. After every 256 new transitions, the learner takes $\lceil 10 \cdot 256/12 \rceil = 214$ gradient steps on mini-batches of 12, so each transition is sampled about ten times on average. Heterogeneous training keeps a UGV buffer and a UAV buffer and splits the gradient steps between the two types in proportion to their buffer sizes. Optimization uses Adam (learning rate 3×10^{-4}), gradient clipping at norm 10, and bf16 mixed precision.

Exploration and training length. Exploration is ϵ-greedy, decaying linearly from 1.0 to 0.1 over the first $50{,}000$ transitions ($80{,}000$ for the single UAV and heterogeneous teams). The first training seeds were run to $86{,}287$ decisions (single UGV), $64{,}160$ robot decisions (UGV team), and $68{,}419$ robot decisions (heterogeneous team). The additional seeds of App. C.1 were run to the same lengths.

Checkpoint selection. Every $4{,}096$ transitions, the greedy policy is evaluated for 50 episodes on the *training* environment (`maze`, $\rho = 0.1$, randomized training budgets), scored by undiscounted training return. Single-robot policies use the best such checkpoint, saved only if its score also beats the previous best on a second, independent 50-episode evaluation. Team policies use the final checkpoint. No evaluation map, evaluation budget, or F-score enters checkpoint selection.

Behavior cloning. The BC policies use the QUEST architecture. We collect $50{,}000$ to $250{,}000$ state–action pairs from the planners (Greedy-EIG or Path-EIG) with no exploration on the Gascola map, and minimize the cross-entropy between the softmax of the network outputs and the expert action. Training runs for up to $10{,}000$ epochs with early stopping on a 20% validation split. UAV BC policies are trained in the same way from UAV planners. In teams, each robot runs the single-robot BC policy of its type.

B.4 Planners

Given belief $\mathbf{b}$ and robot cell p, the path-integral planner of Gupta et al. [18] runs breadth-first search from p and estimates the reward accumulated along the shortest path to each candidate endpoint g by Monte Carlo simulation of observations (10 samples for the belief update and 25 for the reward), and chooses

$$g^\star = \arg\max_g \frac{\hat{R}(p \to g)}{1 + \xi\, d(p, g)}, \qquad \xi = 0.2, \tag{10}$$

where $d(p, g)$ is the shortest-path length. Path-EIG uses EIG as the per-cell reward and Path-NATS-MR uses the NATS score [16]. The greedy planners score only the endpoint. In teams, every robot applies the same rule to the shared belief when it becomes ready. The per-decision cost grows with $|\mathcal{S}|^2$ times the number of Monte Carlo samples, which is why the planners were evaluated on 50 episodes.

Hyperparameter	Value
Network	
Graph attention layers / heads / width	4 / 8 / 64
Normalization / activation	LayerNorm / ReLU
Neighborhood threshold on $1/(d_M+1)$	0.10 (UGV-only policies), 0.125 (with UAVs)
Remaining-budget normalizers ($\bar{B}$, $\bar{T}$)	1000, 100
Optimization	
Target	clipped double-Q with Double-DQN selection (Eq. 5)
Discount γ (per decision)	0.95
Optimizer / learning rate / mini-batch	Adam / 3×10^{-4} / 12
Loss / gradient clipping / precision	squared error / norm 10 / bf16 mixed
Target update	Polyak, rate 0.005
Data collection	
Workers / replay capacity	16 / 10^6 transitions
Update schedule	214 gradient steps per 256 new transitions
Exploration	ϵ: $1.0 \to 0.1$ over 50K transitions (80K with UAVs)
Training length (transitions)	86,287 (one UGV), 64,160 (UGV team), 68,419 (UAV–UGV)
Checkpoint	best validated (single robot); final (teams)
Validation	every 4,096 transitions, 50 episodes on the training map
Behavior cloning	
Data	50K–250K expert state–action pairs on the Gascola map
Loss / stopping	cross-entropy / early stopping on a 20% validation split

Table 10: **Hyperparameters.** Reward coefficients are in Table 7. A transition is one completed decision of one robot.

B.5 Semi-Markov variant

A waypoint is an option of variable duration ℓ_n. The semi-Markov optimality equation [40] discounts the bootstrap by the option's duration,

$$Q^\star(s_n, a_n) = \mathbb{E}\Big[\sum_{t=0}^{\ell_n - 1} \gamma^t r_{n,t} + \gamma^{\ell_n} \max_{a'} Q^\star(s_{n+1}, a') \Big], \tag{11}$$

where $r_{n,t}$ is the reward at tick t of the option. With a per-tick discount near one, long detours pay an opportunity cost before the bootstrap. For example, $\gamma = 0.998$ discounts the bootstrap of a 20-cell route by 0.96 and of a 100-cell route by 0.82.

What we tried. We discounted the bootstrap by γ^{ℓ_n} with $\gamma \in \{0.997, 0.998, 0.999\}$. With fixed budgets, the single-UGV variant learned useful policies after about 200,000 transitions and was robust to $\gamma \in \{0.997, 0.998\}$. Combined with randomized budgets in team and heterogeneous training, however, training curves collapsed.

Why the per-decision target. Route duration already enters the reward through the step cost and the efficiency term, so γ^{ℓ_n} adds a second duration preference, and randomized budgets widen the distributions of both returns and route durations. The per-decision target of Eq. 5 was stable in every setting, so all reported results use it.

B.6 Hyperparameters

Table 10 collects the network, optimization, and data-collection settings described above.

C Additional results

C.1 Training-seed replicates

The main-text results replicate across training seeds. We trained two additional seeds of the single-UGV and UGV-team configurations, identical to the first in configuration, training length, checkpoint

rule, and evaluation episodes. Table 11 reports every single-robot cell and Table 12 every team cell. All single-robot seeds and baselines are evaluated in one run on the same 50 episodes per map, the run behind Fig. 5.

Map	Seed	150	200	250	300
`maze`	0	$+0.034$	$+0.026$	$+0.022$	$+0.017$
	1	$+0.017$	$+0.027$	$+0.034^*$	$+0.031^*$
	2	$+0.013$	$+0.023$	$+0.017$	$+0.022$
`tunnels`	0	$+0.035$	$+0.023$	$+0.002$	-0.036^*
	1	$+0.021$	$+0.008$	-0.009	-0.037^*
	2	$+0.020$	$+0.000$	-0.021	-0.041^*
`asterisk`	0	$+0.007$	$+0.007$	$+0.000$	-0.008
	1	-0.004	-0.002	-0.005	-0.025^*
	2	-0.002	$+0.006$	$+0.006$	-0.002
`open-asterisk`	0	$+0.024$	$+0.018$	$+0.005$	-0.001
	1	$+0.012$	$+0.015$	-0.009	-0.032^*
	2	$+0.023$	$+0.023$	$+0.009$	-0.004

Table 11: **QUEST-SR leads on `maze` for every seed and early on the unseen maps.** ΔF (QUEST-SR minus BC-PathEIG) at path length 150–300, 50 episodes per cell; seed 0 is the policy in the main text. *Welch $p < 0.05$.

Seed	$M = 2$	$M = 3$	$M = 4$	Growth of lead, $M = 2 \to 4$
0	0.915 ± 0.008	0.922 ± 0.007	0.913 ± 0.005	$+0.046 \pm 0.014$ ($p = 0.001$)
1	0.923 ± 0.006	0.892 ± 0.010	0.900 ± 0.007	$+0.025 \pm 0.013$ ($p = 0.065$)
2	0.926 ± 0.007	0.918 ± 0.007	0.908 ± 0.009	$+0.030 \pm 0.016$ ($p = 0.061$)
BC-PathEIG	0.901	0.886	0.853	

Table 12: **Every training seed leads BC-PathEIG at every team size.** F-score of QUEST-MR at team path length 500 on the first 50-episode block. Growth: $[F(4) - F_{\mathrm{BC}}(4)] - [F(2) - F_{\mathrm{BC}}(2)]$, paired per episode.

Reading the replicates. The two results the paper relies on replicate. The first is the lead on the constrained training map ($12/12$ positive cells, $+0.013$ to $+0.034$). The second is the growth of the team lead with team size, which is positive for every seed. On the unseen maps every seed leads at path length 150 on `tunnels` and `open-asterisk` and is at parity on `asterisk`; by path length 300 BC-PathEIG has caught up on the open maps (across-seed mean ΔF of -0.012 on `asterisk` and -0.013 on `open-asterisk`) and leads on `tunnels` (-0.038). The largest per-seed drop in QUEST-MR's F-score from $M = 2$ to $M = 4$ is 0.023, versus 0.048 for BC-PathEIG.

C.2 Team evaluation over 1,000 episodes

The team lead is consistent across blocks of 50 episodes and peaks mid-mission. For each team size we evaluate Random and the four learned policies on the same 1,000 episodes and split them into 20 fixed blocks of 50. The first block is the 50-episode evaluation set used for the online planners and stress tests (e.g., Table 5). Table 13 reports the lead over BC-PathEIG at every checkpoint. After path length 100, QUEST-MR leads in at least 19 of 20 blocks at every team size, and no block reverses the $M = 3$ or $M = 4$ comparison at the end of the budget. The growth of the lead from $M = 2$ to $M = 4$ is $+0.019 \pm 0.003$ over BC-PathEIG ($18/20$ blocks, paired $p = 5.1 \times 10^{-6}$) and $+0.053 \pm 0.005$ over Single-policy-as-MR ($19/20$, $p = 7.2 \times 10^{-9}$).

Team size	100	200	300	400	500
$M = 2$	$+0.026$ (17)	$+0.050$ (20)	$+0.041$ (20)	$+0.030$ (20)	$+0.016$ (19)
$M = 3$	$+0.015$ (17)	$+0.038$ (20)	$+0.048$ (20)	$+0.038$ (20)	$+0.027$ (20)
$M = 4$	$+0.008$ (12)	$+0.021$ (19)	$+0.041$ (20)	$+0.043$ (20)	$+0.035$ (20)

Table 13: **The team lead peaks mid-mission at every team size.** Lead of QUEST-MR over BC-PathEIG, mean over 20 blocks of 50 episodes; parentheses count blocks with a positive lead.

C.3 Claim-map intervention

This intervention tests whether the trained policy actually uses its view of teammates' routes. We evaluate QUEST-MR on the same 50 episodes twice: once as trained and once with its claim input set to zero. The weights and all other inputs are unchanged. Table 14 reports the per-episode drop in F-score.

Budget checkpoint	$M = 2$	$M = 3$	$M = 4$
50%	0.039 ($p = 0.003$)	0.055 ($p = 10^{-4}$)	0.046 ($p = 10^{-4}$)
75%	0.015 ($p = 0.18$)	0.049 ($p = 4 \times 10^{-6}$)	0.060 ($p = 2 \times 10^{-6}$)
100%	0.023 ($p = 0.041$)	0.021 ($p = 0.005$)	0.039 ($p = 3.1 \times 10^{-4}$)

Table 14: **Zeroing the claim map costs more as the team grows.** Mean drop in F-score when only the claim input is zeroed at test time, by fraction of the team budget consumed; paired over 50 episodes.

The mean cost across checkpoints rises from 0.026 ($M = 2$) to 0.042 ($M = 3$) and 0.048 ($M = 4$). The difference between the $M = 4$ and $M = 2$ costs is $+0.006 \pm 0.017$ at 50% ($p = 0.70$), $+0.045 \pm 0.016$ at 75% ($p = 0.004$), and $+0.016 \pm 0.015$ at 100% ($p = 0.28$). At 50% the two team sizes pay a similar cost, and at 100% both approach their F-score ceiling. Zeroing an input measures what the trained policy relies on, not what the channel contributes during training.

C.4 Trajectory diagnostics

These diagnostics measure how a policy moves between regions of a map. Regions are the communities found by greedy modularity maximization [8] on the search graph $G_{\mathcal{S}}$ of each map (searchable cells, four-connected). We record the robot's region at every decision epoch. The robot *exits* a region when two consecutive decisions start in different regions. *Resolved before exit* is the fraction of the exited region's cells with $\mathbf{b}_c \leq 0.1$ or $\mathbf{b}_c \geq 0.9$ at that moment, averaged over the exits in an episode. A *re-entry* is a decision that starts in a region the robot has exited before. Table 15 gives the values with standard errors (50 episodes per policy and map, the evaluation protocol of §5.3).

	Resolved before exit		Re-entries	
Map	QUEST-SR	BC-PathEIG	QUEST-SR	BC-PathEIG
`maze`	0.843 ± 0.006	0.740 ± 0.008	2.12 ± 0.12	2.76 ± 0.12
`tunnels`	0.856 ± 0.009	0.824 ± 0.006	2.80 ± 0.29	4.18 ± 0.11
`asterisk`	0.858 ± 0.004	0.785 ± 0.005	2.94 ± 0.17	3.24 ± 0.10
`open-asterisk`	0.864 ± 0.005	0.797 ± 0.006	3.28 ± 0.14	2.88 ± 0.13

Table 15: **QUEST-SR resolves more of each region before leaving it on every map.** Means ± 1 s.e. over 50 episodes per policy and map; definitions in the text.

C.5 Heterogeneous composition grid

QUEST-Het wins the most compositions while the budget binds, and BC-PathEIG once it saturates. Tables 16 and 17 report every composition at team path length 500 and at the end of the budget ($B_{\mathrm{UGV}} = 400$, $B_{\mathrm{UAV}} = 1000$, $T = 60$, $\rho = 0.03$, 25 episodes per cell). At path length 500, QUEST-Het has the best F-score on 7 of 16 compositions, including every one-UAV composition with two or more UGVs. At the end of the budget, BC-PathEIG is best on 14 of 16. On the one-UAV compositions at path length 500, Welch tests of QUEST-Het against each baseline give the following ($n = 25$ per side):

- vs. Random: $\Delta = +0.106, +0.082, +0.061$ for $U = 2, 3, 4$ ($p < 0.001, 0.006, 0.008$);
- vs. BC-GreedyEIG: $\Delta = +0.093, +0.104, +0.084$ ($p < 0.001, < 0.001, 0.002$);
- vs. Single-policy-as-Het: $\Delta = +0.049, +0.086, +0.142$ ($p = 0.063, 0.002, < 0.001$);
- vs. BC-PathEIG: $\Delta = +0.053, +0.031, +0.007$ ($p = 0.044$ at $U = 2$; not significant at $U = 3, 4$).

Pooled over the three compositions (Table 3), the lead over BC-PathEIG is significant ($p < 0.05$).

UGVs U	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
UAVs V	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Random	.61	.53	**.53**	.48	.63	.57	**.58**	.54	.69	.63	.60	**.59**	.71	.68	.63	.59
BC-GreedyEIG	.59	.51	.49	**.49**	.65	.59	.54	**.54**	.67	.62	.60	.53	.68	.64	.61	.57
BC-PathEIG	**.64**	.52	.51	.48	.69	.62	.53	.52	.74	.65	.60	.58	.76	**.70**	**.66**	.60
Single-policy-as-Het	.64	**.54**	.50	.44	.69	.56	.53	.52	.68	.61	.57	.55	.63	.55	.54	.52
QUEST-Het	.63	.54	.49	.45	**.74**	**.62**	.56	.52	**.77**	**.68**	**.62**	.57	**.77**	.68	.65	**.64**

Table 16: **Heterogeneous grid at team path length 500.** Mean F-score over 25 episodes per composition (U, V) (U UGVs, V UAVs). Best per column in bold. QUEST-Het was trained on $U, V \in \{1, 2\}$.

UGVs U	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4
UAVs V	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Random	.77	.70	.71	.65	.78	.78	.76	.74	.76	.79	.77	.75	.75	.80	.79	.78
BC-GreedyEIG	.86	.77	.73	.66	.82	.83	.74	.76	.79	.80	.81	.77	.78	.82	.83	.82
BC-PathEIG	**.90**	**.82**	**.78**	**.74**	**.91**	**.88**	**.86**	**.81**	.88	**.88**	**.86**	**.82**	.84	**.88**	**.87**	**.88**
Single-policy-as-Het	.84	.61	.58	.55	.83	.71	.66	.59	.81	.79	.71	.64	.68	.63	.66	.66
QUEST-Het	.89	.75	.73	.69	.86	.81	.80	.77	**.89**	.78	.76	.76	**.86**	.81	.76	.82

Table 17: **Heterogeneous grid at the end of the budget.** Mean F-score over 25 episodes per composition (U, V). Best per column in bold.

C.6 Heterogeneous roles

QUEST-Het's UAV uses both screening and confirmation, where every single-agent policy favors one. On two-UGV–one-UAV teams (`maze`, 25 episodes), Table 4 (§5.4) measures the division of labor. *Duplicate coverage* is the fraction of sensed cells observed by more than one robot (stricter than, and not comparable to, the redundant-cell metric of Table 3). The UAV columns give the share of its actions at the widest (8×8) and narrowest (2×2) footprints; the remainder use the 4×4 footprint.

C.7 Field-site map

The Gascola map is a 30×30 digitization of the field site of our ground-vehicle deployment [18] (§5.1). The QUEST policies are transferred zero-shot from `maze`, while the BC baselines were trained on this map. Budgets are scaled to the map's size: $B = 141$ for one UGV, 282 for four UGVs, and 188 (UGVs) and 471 (UAV) for two UGVs and one UAV. Decision caps do not bind, and $\rho = 0.03$. Table 18 gives the end-of-budget F-scores and Fig. 7 the full curves.

Setting	QUEST	BC-PathEIG	ΔF	p	Episodes
Single UGV	0.927 ± 0.008	0.953 ± 0.007	-0.025	0.021	50
Four UGVs	0.937 ± 0.008	0.936 ± 0.010	$+0.001$	0.95	50
Two UGVs, one UAV	0.909 ± 0.013	0.943 ± 0.010	-0.034	0.048	25

Table 18: **On the Gascola map, QUEST-MR ties BC-PathEIG, the policy trained there.** F-score at the end of the scaled budget, QUEST transferred zero-shot from `maze`. QUEST-MR and BC-PathEIG never differ significantly at any checkpoint.

C.8 Robustness protocols

Communication outage. In each episode, $K = \min(4, M - 1)$ robots are scheduled to lose communication (three of four in Table 5). Each outage starts at a team path length drawn (stratified) from $[1, s_{\max}B]$ and lasts for a length drawn uniformly from $[1, d_{\max}B]$, with $B = 500$ the team budget. Table 5 uses $s_{\max} = 0.25$ and $d_{\max} \in \{0.25, 0.5\}$. A disconnected robot keeps moving and sensing, but into a private copy of the belief taken when the outage starts. Teammates see its last broadcast position and committed route, and it sees the same stale view of them. On reconnection, its withheld observations are applied to the shared belief. This is exact, because per-cell Bayesian updates commute, and the resulting entropy reduction is credited to the robot that collected the observations. The protocol therefore models delayed sharing, not a permanent network partition.

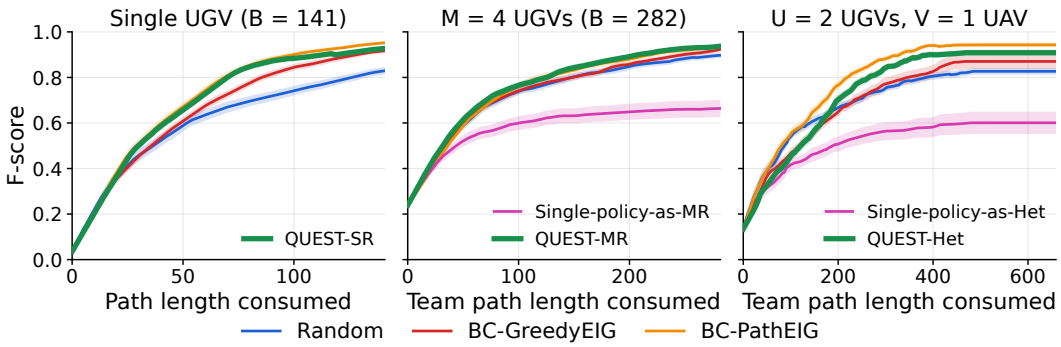


Figure 7: **On the Gascola map, QUEST-MR tracks BC-PathEIG at every budget.** F-score versus path length for the three settings of Table 18; bands ± 1 s.e.

Robot failure. At reset, K failure times are drawn in $[1, f_{\max} \cdot B]$ (team path length) and assigned to distinct robots, keeping at least one robot alive. A failed robot stops moving, sensing, and contributing to the belief at the end of the tick in which its failure time passes. Its route is truncated and it is removed from teammates' inputs and from the claim map, so teammates see it disappear. In heterogeneous teams, failures are drawn per platform type.

Checkpoints for failure columns. Failure columns report F at team path length $\min\{500,\ 0.3\,[(M-K)\cdot 500 + K\cdot(f_{\max}/2)\cdot 500]\}$, which places every checkpoint near the end of the team budget: 472.5, 487.5, 495, and 500 for one failure before $f_{\max} = 0.3, 0.5, 0.6, 1.0$, and 450 for two failures. The no-failure column uses 500. Every method is evaluated at the same checkpoint within a column. Each cell uses 50 episodes, except one failure before 0.5, which uses 75.

C.9 Communication-outage envelope and overlap

QUEST-MR leads under every outage we tested. Figure 8 sweeps the maximum outage start $s_{\max}$ and the maximum duration $d_{\max}$ for four UGVs (50 episodes per cell, F at team path length 500). The lead of QUEST-MR over BC-PathEIG is positive in every evaluated cell, from $+0.027$ (start up to 50%, duration up to 75%) to $+0.061$. Short outages barely reduce the lead, whenever they occur, because much of the coordination benefit accrues outside the outage window. Late outages barely reduce it either, whatever their length, because most informative search has happened before the link drops. The lead falls below $+0.040$ only when an early start is combined with a long duration. Table 19 shows the mechanism. Cell overlap between robots rises under outage, but QUEST-MR's overlap at the longest outage (0.511) is still below BC-PathEIG's clean overlap (0.535).

Method	Clean	Outage $\leq 25\%$	Outage $\leq 50\%$
Random	0.519 ± 0.008	0.514 ± 0.009	0.515 ± 0.010
BC-GreedyEIG	0.541 ± 0.010	0.528 ± 0.010	0.528 ± 0.010
BC-PathEIG	0.535 ± 0.009	0.542 ± 0.010	0.567 ± 0.011
QUEST-MR	$\mathbf{0.459 \pm 0.010}$	$\mathbf{0.490 \pm 0.009}$	$\mathbf{0.511 \pm 0.010}$

Table 19: **Under the longest outage, QUEST-MR still overlaps less than BC-PathEIG does with full communication.** Cell overlap between robots (lower is better); four UGVs, 50 episodes. F-scores for the same runs are in Table 5.

C.10 Robot failure in heterogeneous teams

We evaluate two-UGV–one-UAV teams ($B_{\mathrm{UGV}} = 500$, $B_{\mathrm{UAV}} = 1000$, 25 episodes) with and without the failure of one UGV. We report F-score at team decision step 20, while the budget still binds. QUEST-Het is the only learned policy whose F-score does not fall (Fig. 9), and Table 20 shows how its behavior changes: the UAV shifts from wide to narrow footprints and the surviving robots take longer routes per decision. BC-PathEIG's footprint mix changes by at most 1.3 percentage points.

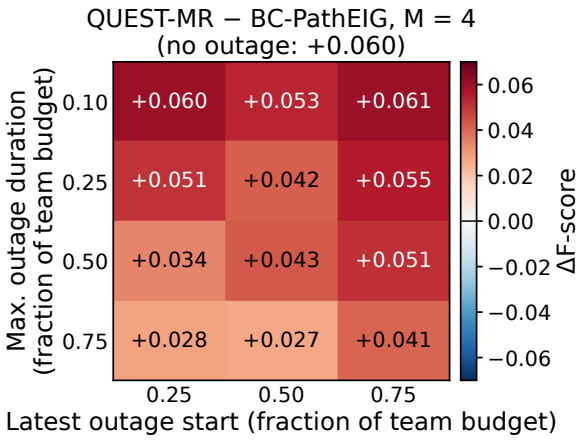


Figure 8: **Short outages barely dent the lead.** Lead of QUEST-MR over BC-PathEIG with four UGVs on `maze`, by maximum outage start (x) and maximum duration (y) as fractions of the team budget.

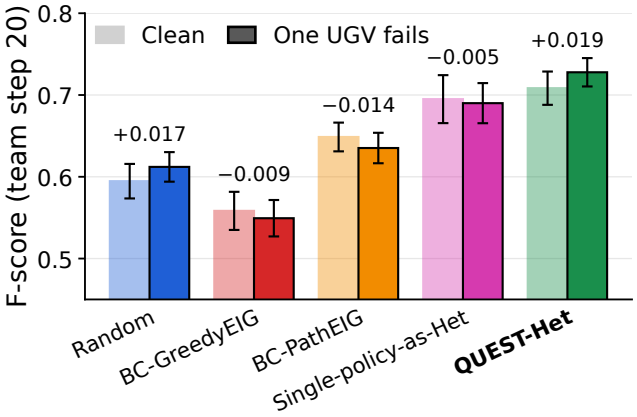


Figure 9: **One UGV fails in a two-UGV–one-UAV team.** F-score at team decision step 20, clean and with failure (25 episodes, ± 1 s.e.).

Diagnostic	QUEST-Het: clean → failure	BC-PathEIG: clean → failure
UAV actions at widest footprint	38.9% → 28.9% (−10.0)	19.0% → 20.4% (+1.3)
UAV actions at narrowest footprint	60.9% → 70.9% (+10.0)	71.0% → 69.9% (−1.1)
Path per decision (cells)	12.8 → 15.2 (+2.5)	18.5 → 17.4 (−1.1)

Table 20: **Behavior after one UGV fails** (two-UGV–one-UAV team, 25 episodes). Changes in parentheses; footprint changes are in percentage points.

C.11 Sensitivity to discount and reward coefficients

Table 21 summarizes the sweeps we ran while tuning. Each entry is a single training run per value, judged by training curves and evaluation on the training map, so these are tuning observations, not estimates of seed-level sensitivity.

D Compute and broader impact

D.1 Compute

Each training run used one GPU (NVIDIA RTX A6000, 48 GB; some sweep runs used an H100) and fits in 24 GB under bf16 mixed precision; the largest tensor is the dense distance matrix used to build the message-passing graph. A training run took about 24–36 hours for a single UGV, 36–48

Parameter	Values (setting)	Observation
Discount γ	$0.90, \mathbf{0.95}, 0.97, 0.99$ (all three)	Single UGV insensitive for $\gamma \geq 0.9$. Teams prefer 0.95; 0.97 is within noise for UGV teams and slightly worse for heterogeneous teams; 0.99 gives collapse-prone heterogeneous training.
Completion exponent α	$1, \mathbf{2}, 3$	$\alpha = 1$ settles just above the threshold $u^\star$; $\alpha = 3$ is too steep when resolution stalls below it.
UGV step cost c_ℓ	$0.005, \mathbf{0.01}, 0.02, 0.04$ (heterogeneous)	The UGV wanders far from UAV-flagged regions at 0.005 and barely moves at 0.04.

Table 21: **Tuning sweeps.** One training run per value; bold marks the value used.

hours for UGV teams, and 60–72 hours for heterogeneous teams. The heterogeneous simulator is slowest, because every event advances several platforms and rebuilds the claim map. BC training takes 30–90 minutes. Including exploratory runs and sweeps, training used about $4{,}500$ GPU-hours and evaluation about 200 GPU-hours, most of it for the Monte Carlo planners.

Scaling. The current implementation builds G_{mp} from a dense $O(|\mathcal{S}|^2)$ distance matrix before thresholding it. The retained graph is local (radius 9, at most 181 neighbors), so it can be built directly in $O(|\mathcal{S}|r^2)$ time and memory, and attention then scales with the number of edges. Much larger areas would call for hierarchical or local subgraphs.

D.2 Broader impact

Better coordination of search teams can shorten search-and-rescue operations and reduce the cost of environmental monitoring and hazard detection. The same capability could support surveillance of people. Our work addresses mission-level decision making in simulation. Any deployment would need the safety, privacy, and legal oversight appropriate to the application.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the contributions (formulation, method, empirical characterization) and their scope: a standard Bellman objective, simulated evaluation, and gains that depend on topology, team contention, and budget (§1, §5).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The scope of every claim is stated with its result: simulation only (§1), protocols and training maps (§5.1, App. A.2), seed dependence of single-robot transfer (App. C.1), open-map and saturation regimes where BC is competitive (§5.3, §5.4, §6), tested outage protocol (§5.5), semi-Markov instability (App. B.5), single-seed tuning sweeps (App. C.11), and graph-construction scaling (App. D.1).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: There are no rigorous theoretical results proposed in this paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The method, learning target, and training loop are in §4 (Algorithm 1); environments, protocols, rewards, and hyperparameters are in Apps. A–B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code is not released. The environments, sensor model, rewards, network, training loop (Algorithm 1), protocols, and all hyperparameters needed to reimplement the method are specified in §4 and Apps. A–B.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and evaluation protocols, checkpoint selection, and hyperparameters are in §5.1 and Apps. A.2, B.3, and B.6.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Error bars are ± 1 standard error over evaluation episodes (25–1,000 per cell, stated per result), with Welch or paired t-tests. Headline single- and multi-robot configurations are replicated over three training seeds (App. C.1).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: GPU type, memory, per-run training time, and total compute are in App. D.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: App. D.2 discusses benefits (search and rescue, monitoring) and the risk of use for surveillance of people.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We don't anticipate any such risks associated with this work.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All code, models, and maps were created by the authors; experiments use simulated environments, and the field-site map is a digitized occupancy grid of a site used in the authors' prior work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing experiments and research was conducted for this work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This question is not relevant for our work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: This paper doesn't have any LLM use in its core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.