2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Efficient Active Search via Amortized Path-Integral Policies

Tejus Gupta*,[1], Arsh Verma*,[1], Raymond Song[1], David Guttendorf[1], Conor Igoe[1], Luis Navarro-Serment[1] and Jeff Schneider[1]

***Abstract*— This work presents amortized path-integral policies that enable efficient and real-time active search for robotic systems. We model search as an active sensing problem where agents select actions to maximize information about target locations. Unlike previous approaches that only consider information gain at final waypoints, our method accounts for observations along entire paths. To address the computational expense of path-integral policies, we amortize costs through Graph Neural Network (GNN) policies trained via behavior cloning. GNNs provide equivariance to spatial transformations and generalize across diverse maps. We validate our approach through field experiments in a 75,000 m² forested environment using an autonomous ground vehicle, along with simulated testing. Our experiments demonstrate successful policy amortization, cross-map transfer, and improved search efficiency.**

I. INTRODUCTION

Robotic systems are increasingly being deployed for search and monitoring operations [1], [2], [3], [4], [5], [6], [7]. These systems offer significant advantages: they can operate more efficiently than human-operated alternatives, enable coordination between large numbers of heterogeneous agents, and can be deployed in situations too dangerous for human operators. This work presents amortized path-integral policies that enable real-time and efficient active search.

We formulate search as an active sensing problem where the agent selects sensing actions to maximize information about target locations. The agent maintains a belief over object locations and updates it using observations. Following previous work [8], [9], we model the agent's observation model as heteroscedastic, i.e., the robot's perception system becomes less accurate as objects are farther away. The agent then selects a waypoint by optimizing the information gain among a candidate set of waypoints. However, previous approaches suffer from a key limitation in this step: they only consider information gain at the final waypoint. This neglects observations made along the path, leading to suboptimal search performance.

Although path-integral policies have been studied in prior work [10], [11], they are difficult to deploy in robotic systems due to their computational expense. We address this challenge by amortizing the computational cost through learning neural network-based policies through behavior cloning. This approach enables efficient, real-time decision-making for active search. We parameterize our active search policies using Graph Neural Networks (GNNs) since they offer several unique benefits: (i) they are equivariant to rotations and translations of the search domain; (ii) spatial relationships can be naturally encoded as graphs; and (iii) they generalize across diverse maps.

Fig. 1: Testing setup: (Top) Aerial view of our field testing site; (Bottom left) Top-down view of the site with the search polygon in red, object of interest (OOI) locations shown as yellow stars and the robot's starting location shown in blue; (Bottom right) Ground robot and an example detection of an OOI (mannequin).

We empirically validate our approach through both simulation and field experiments. We conduct field tests in an unstructured forested environment with a search area of $\approx 75,000m^2$ using an autonomous ground vehicle and mannequins serving as objects of interest, as shown in Figure 1. Our results demonstrate that we can effectively amortize information-gain-based policies using GNNs, that these policies transfer well across different maps, and that they improve efficiency in field experiments. Thus, our methodology offers a significant improvement for real-time and efficient active search operations.

In summary, our contributions are threefold: (1) we pro-

* Equal contribution

[1]All authors are with the Robotics Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213

Corresponding authors: Tejus Gupta, Arsh Verma and Jeff Schneider {tejusg, arshv, jeff4}@cs.cmu.edu

pose amortized path-integral policies for real-time active search; (2) we design GNN-based policies that are equivariant to transformations of the search domain, allowing transfer across maps; and (3) we validate our approach in both simulations and large-field experiments, demonstrating improved efficiency and generalization.

II. RELATED WORK

Coverage search [12], [13] is a widely used strategy for robotic search that selects waypoints to systematically cover the entire search region, often following regular "lawnmower" patterns [14]. While coverage methods guarantee complete area exploration, they treat all regions equally regardless of the likelihood of finding objects. This uniform sampling can be inefficient when objects are sparsely distributed or when the belief state suggests certain areas are more promising. Consequently, informative action selection methods that adaptively focus sensing effort on regions of high uncertainty or high expected information gain have been found to outperform coverage planning and random exploration [14], [15].

Bayesian active-search methods [16], [6], [17] offer an effective way to adaptively select informative waypoints. They balance exploring high-uncertainty regions and exploiting promising locations. These methods readily incorporate prior knowledge about the search space and the robot's observation model. Recent work, such as NATS [18], has extended this line of work by integrating more realistic depth-aware observation noise modeling and decentralized multi-robot systems. GUTS [9] follows up by deploying a multi-robot search system involving drones and ground robots in unstructured environments. However, despite these advances, the computational complexity inherent in these Bayesian approaches presents significant challenges for practical deployment.

These methods remain computationally expensive in practice. GUTS, for instance, requires a very coarse map discretization (60×60m grid cells) for larger experiments, which limits precision in waypoint selection. Even with this coarse discretization, waypoint planning takes approximately 30 seconds, consuming valuable search time. The system must further subsample candidate waypoints on drones due to limited onboard computing resources. These costs arise from just evaluating information gain at the final waypoints; integrating information gain across entire paths would be computationally prohibitive for online operations. This highlights a fundamental tension between real-time decision-making requirements and the decision quality achievable by current Bayesian active-search methods.

Another prominent approach to informative action selection is to optimize informative paths using sampling-based planners [19], [20]. The main advantage of this approach is that it inherits sampling-based planners' ability to plan in high-dimensional spaces and handle non-trivial sensor constraints. However, the computational cost of optimizing information-theoretic objectives over sampled path trees is substantial, limiting these methods to generating open-loop

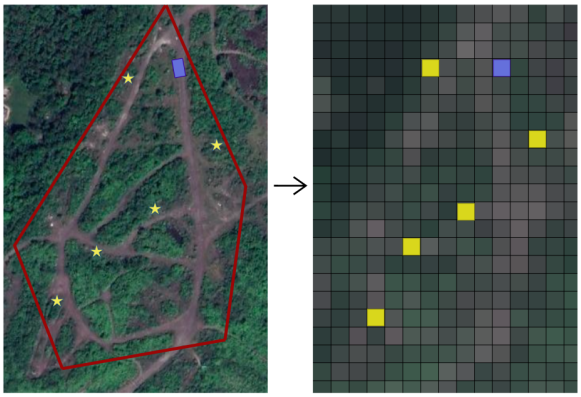

Fig. 2: Map discretization: (Left) top-down view of search area with yellow stars showing OOIs and blue rectangle showing robot position; (Right) discretized map with OOIs marked in yellow and the robot position in blue.

global plans that cannot be efficiently updated as new observations are received. This inability to dynamically replan based on online observations leaves significant performance on the table.

Amortization through behavior cloning has proved to be a promising approach to overcoming the high computational cost of online decision-making in several robotic tasks [21], [22], [23]. This strategy generates large datasets from computationally expensive "expert" algorithms during offline training, then learns neural network policies that approximate the expert's decision-making process. The key advantage is shifting the computational burden from online execution to offline training: the resulting learned policy can execute in real-time at a fraction of the original cost, enabling high-quality decision-making for computationally demanding tasks like active search. Recent work [24] shows that the choice of policy network architecture strongly affects sample efficiency and generalization in active-search tasks. In particular, GNNs perform well due to their spatial equivariance inductive bias, which allows policies to transfer across maps with different layouts.

Our work follows a rich set of contemporary research on learning robotic search policies [25], [26], [27]. These works leverage methods like reinforcement learning and multi-agent training to learn efficient active-search policies. While these approaches show promise in controlled simulation environments, they face significant challenges when deployed on physical robotic systems at scale, such as the sim-to-real gap in sensor modeling, computational constraints of onboard processing, and the need for policies that generalize across diverse search environments. Our work focuses on these practical deployment challenges by combining expert imitation learning with extensive real-world validation, bridging the previous gap in translating learned search policies from simulation to practice.

III. PROBLEM FORMULATION

We formulate active search as a Bayesian experimental design problem where we optimize sensing actions to reduce uncertainty about object locations. We discretize the search region Ω into a grid $M \times N$, where each cell (i, j) may

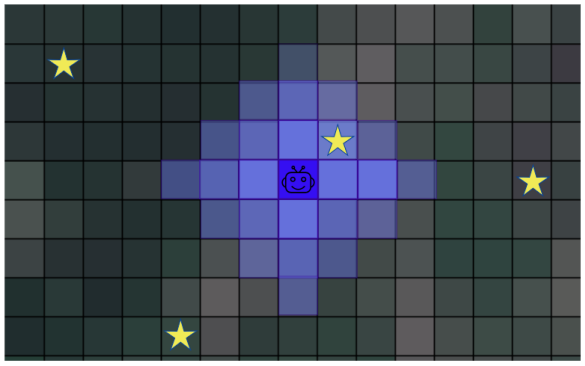

Fig. 3: The robot makes sensing actions to identify objects of interest (yellow stars). The robot can sense in all directions at from any given location. The noise in observations increases with distance from the robot.

contain an object of interest (OOI). Our variable of interest $\theta \in \{0,1\}^{(M\times N)}$ is a binary matrix where $\theta_{i,j} = 1$ if cell (i,j) contains an OOI and 0 otherwise.

Observation model: At sensing location x, the robot observes cells within its field of view (FOV) - an omni-directional region with Manhattan radius of r cells. For each observed cell (i,j), the robot receives a noisy observation $y_{i,j}$ following the same distance-dependent heteroscedastic model as NATS and GUTS [18], [9]. This noise model matches our on-robot perception system, whose errors increase with distance from the robot.

Action space and dynamics: The robot selects waypoints x from traversable grid cells. The movement between waypoints follows shortest paths, during which the robot passively observes cells within field of view (modeling realistic sensor behavior during navigation).

Bayesian Inference: The robot starts with a sparse and cell-wise independent Bernoulli prior $p_0(\theta)$, sequentially chooses actions $\{x_1, ..., x_t\}$, receives observations $\{y_1, ..., y_t\}$, and computes the posterior over object locations $p_t(\theta|\{x_i, y_i\}_{i=1}^t)$ analytically since conjugacy is preserved in our setup. We declare an object found when $p_t(\theta_{i,j} = 1) \geq \tau$, where τ balances detection sensitivity with false positives.

Objective: We seek policies $\pi(x|p_t(\theta), s_t)$ mapping belief states $p_t(\theta)$ and robot positions s_t to informative actions, evaluated by average recall over path length during policy rollouts.

IV. System Description

This section describes the autonomy components for the custom ground robot we use for testing our search methods.

Perception system: Our perception system employs two identical RGB cameras with different field-of-view configurations: a wide field-of-view (WFOV, 63°) camera and a narrow field-of-view (NFOV, 16°) camera with 4× optical zoom. Both cameras provide 16.2 MP native resolution (5320 × 3032 px). To reduce computational load, images are downsampled by a factor of two before processing, yielding an effective input resolution of 8.1 MP (2660 × 1516 px).

Object Detector: We employ a YOLOv11 [28] model fine-tuned on our custom-labeled dataset to detect objects of interest in the captured images.

Object Tracker: The tracking module uses Range-Parameterized Cubature Kalman Filtering (RPCKF) [29], [30]. For object association, we employ the Hungarian Algorithm [31], [32], where the distance metric is the angular difference between predicted and observed object positions.

UGV Path Planner: The robot's navigation system utilizes the SBPL planner[33] which accounts for terrain traversibility and generates plans between waypoints while avoiding obstacles.

Simulation Environment: We developed a custom Gym [24] environment for data collection and evaluation of search policies. The simulator models our robot's omni-directional sensing capabilities, where the robot observes all cells within a fixed Manhattan distance from its current position. The simulation incorporates realistic observation noise that increases with distance from the robot, matching our physical sensor characteristics.

The objects of interest in our experiments are mannequins positioned throughout the search area to simulate search and rescue scenarios. While our detection system can identify various environmental objects, our field experiments specifically focus on localizing these target mannequins.

V. Methods

This section describes the main components of our approach: (i) baseline search policies used for comparison, (ii) a path-integral (route-aware) search policy that evaluates information gathered along entire trajectories, and (iii) amortized policies that mimic these experts via behavior cloning with Graph Neural Networks (GNNs).

Throughout, the agent maintains a belief $p_t(\theta)$ over the binary occupancy variable $\theta \in \{0,1\}^{M\times N}$ (Sec. III), and s_t denotes the robot's state (current grid cell). Sensing outcomes are generated by the heteroscedastic observation model introduced earlier, and Bayes updates yield the posterior $p_{t+1}(\theta)$ after taking action x and observing y.

A. Baselines

We compare against three standard baselines that select the next waypoint x_{t+1} from the set of traversable cells $\mathcal{X}_{\text{trav}}$.

Random: Select x_{t+1} uniformly at random from $\mathcal{X}_{\text{trav}}$.

Coverage: Visit all cells in a predetermined order (e.g., lawnmower pattern) until the budget is exhausted. Coverage treats all regions equally, providing completeness under sufficient budget but wasting effort when the belief state concentrates mass.

One-step lookahead (greedy): Choose the next waypoint by maximizing the expected utility of sensing at that waypoint given the current belief. Let $r(\cdot)$ be a scalar utility functional of the posterior. We consider expected information gain (EIG) by default, though other utilities (e.g., expected recall) can be substituted. The one-step policy is

$$\pi(x_{t+1}|p_t(\theta), s_t) = \arg\max_{x\in\mathcal{X}_{\text{trav}}} E_{y_{t+1}\sim x_{t+1}} r(p_{t+1}(\theta|x_t, y_t)) \quad (1)$$

When r is EIG, we can write

$$r(p_{t+1}) \;=\; H[p_t(\theta)] \;-\; \mathbb{E}_{y\sim p(y|x,p_t)}[\,H[p_{t+1}(\theta)]\,], \quad (2)$$

where $H[\cdot]$ is the Shannon entropy of the belief over θ.

B. Path Integral Search

Greedy one-step lookahead treats waypoints in isolation and ignores any observations accrued while traveling to the chosen destination. In practice the robot must traverse a sequence of cells to reach a waypoint, collecting measurements along the route. We therefore evaluate the *path* to each candidate waypoint rather than isolated destinations in the Path Integral search method.

Let $\mathcal{X}_{\text{trav}}$ denote the traversable cells and let a planner (SBPL) return the shortest collision-free path from the current state s_t to any $x \in \mathcal{X}_{\text{trav}}$:

$$Path(s_t, x_{t+1}) \;=\; (z_1, z_2, \ldots, z_K), \quad z_1 = s_t,\; z_K = x_{t+1}$$
$$z_{i+1} \text{ is reachable from } z_i \forall i$$

At each step k along $Path(s_t, x_{t+1})$, the robot obtains an observation $y_k \sim p(y \mid z_k, p_t)$ under the heteroscedastic sensor model.

The utility of following the route $Path(s_t, x_{t+1})$ equals the expected reduction in entropy of the map occupancy after assimilating all en-route observations:

$$U_{\text{path}}(x_{t+1}; p_t, s_t) \;=\; \sum_{z_k \in Path(s_t, x_{t+1})} r(p_{t+1}|z_k, y_k). \quad (3)$$

Path-length weighting: To balance information gain against travel effort, we apply a mild length penalty via $w(L) = \frac{1}{1+w_\ell L}$ with $L = |Path(s_t, x_{t+1})|$ and a constant $w_\ell > 0$:

$$x_{t+1} \in \arg\max_{x_{t+1}\in\mathcal{X}_{\text{trav}}} w(|Path(s_t, x_{t+1})|) \cdot U_{\text{path}}(x_{t+1}; p_t, s_t) \quad (4)$$

In our experiments we use $w_\ell = 0.2$; other normalizations (e.g., none, or reward-per-cost) are possible but can lead respectively to excessively long exploratory routes or myopic short hops when nearby rewards are plentiful.

C. Amortized Policies using Behavior Cloning

To enable real-time decision making with path-integral search, we amortize the greedy and path-integral experts using Graph Neural Network (GNN) policies trained by behavior cloning. GNNs provide useful inductive biases (e.g., equivariance to spatial transforms) and generalize across diverse map geometries, while their inference latency is effectively independent of the analytical complexity of the expert they mimic.

Formally, let G_t denote a graph encoding the belief $p_t(\theta)$, the robot state s_t, and traversability structure at time t. We parameterize a stochastic policy $\pi_\phi(x \mid G_t)$ over traversable cells $x \in \mathcal{X}_{\text{trav}}$ and train it to imitate an expert action $x^\star$ with a cross-entropy loss:

$$\mathcal{L}(\phi) = -\,\mathbb{E}_{(G_t, x^\star)\sim\mathcal{D}}\big[\log \pi_\phi(x^\star \mid G_t)\big], \quad x^\star \sim \pi_{\text{expert}}(x_{t+1}; p_t, s_t). \quad (5)$$

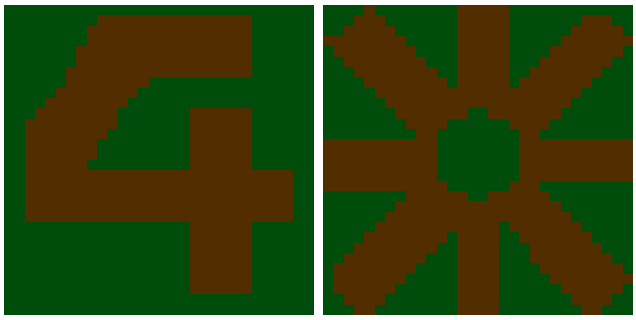

Fig. 4: Synthetic maps generated for training and evaluation. Brown and green depict the traversable and non-traversable areas, respectively.

Method	Waypoint Selection Time
Greedy-EIG	5.30
Path-Integral EIG	122.00
GNN-based Policy	0.48

TABLE I: Run time (per decision) of various experts and GNN policies, averaged over 100 timesteps. All results are for 30×30 maps.

We choose Graph Attention Networks [34] as our choice of GNN. The GNN's input graph, G_t, is constructed from the discretized search map, where each cell (i, j) serves as a node. The nodes are represented by features for the current belief state $p_t(\theta)$ and the traversability of the cell. The edges of the graph are weighted by the inverse of the distance between cells. This structure allows the GNN to naturally encode spatial relationships and generalize across diverse maps. The GNN policy, $\pi_\phi(x|G_t)$, then learns to map this graph representation to a distribution over traversable cells, mimicking the expert's decisions.

VI. Experiments and Results

The core of our experiments is to empirically validate our central claims: that GNN policies can (i) faithfully imitate strong, computationally expensive expert policies and (ii) generalize across diverse search maps, all while preserving real-time decision latency. We achieve this by first evaluating our experts and baselines in simulation, training GNNs via behavior cloning, and then rigorously testing their robustness to various environmental shifts.

We simulated evaluations on four maps: the ice cream cone (ICC, matching our field test environment, Fig. 2), an all-traversable map (AT), and two synthetic maps shown in Figure 4: the open asterisk (OA) and figure four. By default, each map has 30×30 grid cells, with each cell corresponding to $30m \times 30m$ regions. Shaded regions denote the standard error of the mean over 20 independent evaluation trajectories.

A. Imitating Greedy Expert

We begin by *imitating the one-step lookahead greedy policy*. We first compare the performance of our baseline policies, then train GNN policies by behavior cloning on greedy EIG expert rollouts, and finally compare the learned policies to the expert policy for each map.

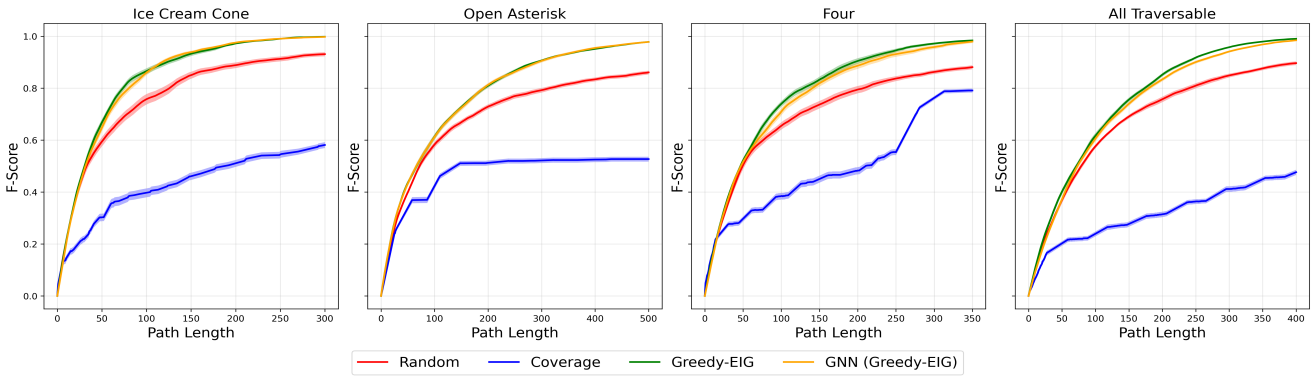

Fig. 5: GNN-based policies (yellow) track the performance of the Greedy-EIG expert (green) for all four maps, demonstrating effective behavior cloning.

Baseline policies' performance: To assess expert performance, we run each policy in a simulator and plot the F-score vs. path length (Fig. 5). As expected, coverage is the most budget-inefficient baseline; random exploration does well at the start of the trial (when most of the map is unseen) but quickly plateaus; and the greedy-EIG policy, while slower per-decision than the former two, adapts online and achieves substantially better recovery given a fixed budget. While both random and greedy achieve an F-score of $\sim$0.6 around the same path length, they diverge significantly from there, with greedy continuing to improve steadily while random exploration quickly saturates.

Training the GNN policies (behavior cloning): We construct a dataset by rolling out the chosen expert in a simulator and recording $(G_t, x^\star)$ pairs for each search map. For the 30$\times$30 setting, a dataset of $\sim$10k samples suffices to clone the experts; each trajectory is capped at 75 timesteps with 20% object density, with hyperparameters scaling with environment size and task complexity.

Evaluation of the learned policies: We train a GNN to imitate the greedy expert and evaluate it on the four maps in Fig. 5 (Ice-Cream-Cone, All-Traversable, Open-Asterisk, Four). The GNN policies closely match the expert, attaining an F-score within 1.5% of the expert at each timestep across all maps while being substantially more efficient to run, confirming the effectiveness of our behavior cloning approach. This improvement underscores why amortization is attractive in practice.

Runtime analysis on the Ice-Cream-Cone map in Tab. I demonstrates the computational advantages of our approach. The GNN policy achieves a per-decision latency of only 0.48s, delivering an 11× speedup compared to the Greedy expert (5.3s) and a substantial 254× speedup over the computationally expensive Path-Integral expert (122s) (Table I).

B. Policy Robustness

We evaluate the robustness of our GNN policies under two types of distribution shifts: cross-map generalization and object density variation. These experiments assess whether policies trained on specific conditions can maintain performance when deployed in different environments. This is crucial for real-world deployments where exact environmental conditions, such as object density or map geometry, may not be known a priori.

Cross-Map Generalization: Figure 6 evaluates the ability of GNN policies to generalize across different map geometries and scales. We train policies on the 30 $\times$ 30 variants of each of the four maps and evaluate all policies on each 60 $\times$ 60 variants. The results demonstrate excellent generalization capabilities: GNN policies track their experts nearly perfectly across all map shapes, confirming that a policy trained on one map can effectively transfer to others without retraining. This result highlights the GNN's ability to learn underlying topological and spatial relationships, rather than memorizing a specific map layout.

Robustness to Object Density Shifts: Figure 7 illustrates the performance of our policies under changes in the density distribution of objects. To simulate a realistic deployment scenario where conditions may not match training environments, we trained policies on 30 $\times$ 30 maps with a 20% ground truth density and evaluated them on the same maps with only 1% density.

All GNN policies demonstrated robust generalization under this shift, closely tracking their respective experts across all four test maps. We further quantified this generalization and the ability to transfer between maps with changes in density in Table II using the area under the F1 curve metric. The policies trained on the ICC and AT maps performed marginally better when evaluated on their original training maps, exceeding their expert performance by 0.5% and 1.5%, respectively. In contrast, the policies trained on the OA and Four maps saw a performance drop of 2% and 3% relative to their experts.

Crucially, the policy trained on the All-Traversable (AT) map achieved the best generalization, with an average performance drop of less than 1% across all maps, outperforming other policies that experienced a 2-3% average drop. This superior performance suggests that training on a generic, less-biased map fosters a more robust and versatile policy, capable of adapting to a wider range of novel environments.

These results establish that our GNN-based policies are not only faithful imitators of their expert teachers but also robust to environmental variations commonly encountered in real-world deployments.

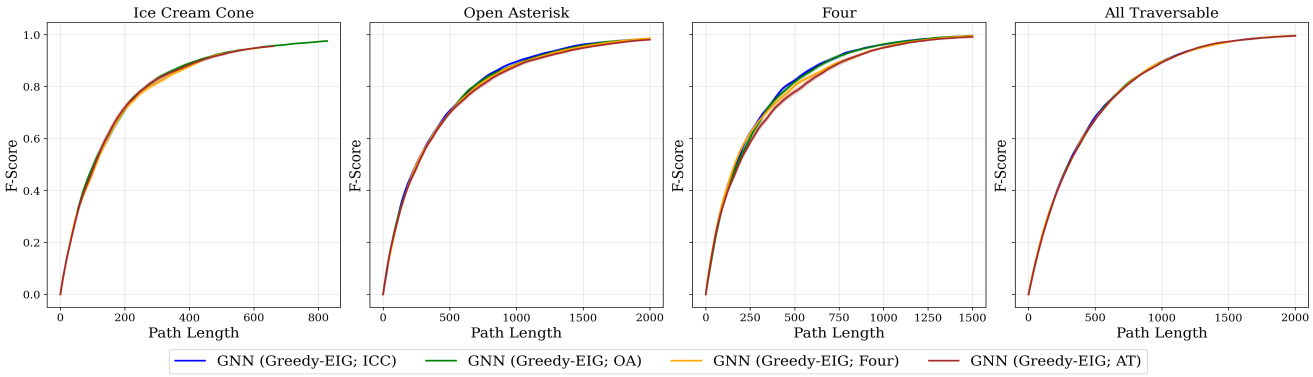

Fig. 6: Cross-map generalization of GNN policies. All policies are trained using the Greedy-EIG expert on 30×30 maps and evaluated on scaled-up 60×60 versions of the same maps. GNN policies closely track expert performance at the new scale without retraining. ICC: Ice Cream Cone, OA: Open Asterisk, Four: Figure Four, AT: All Traversable.

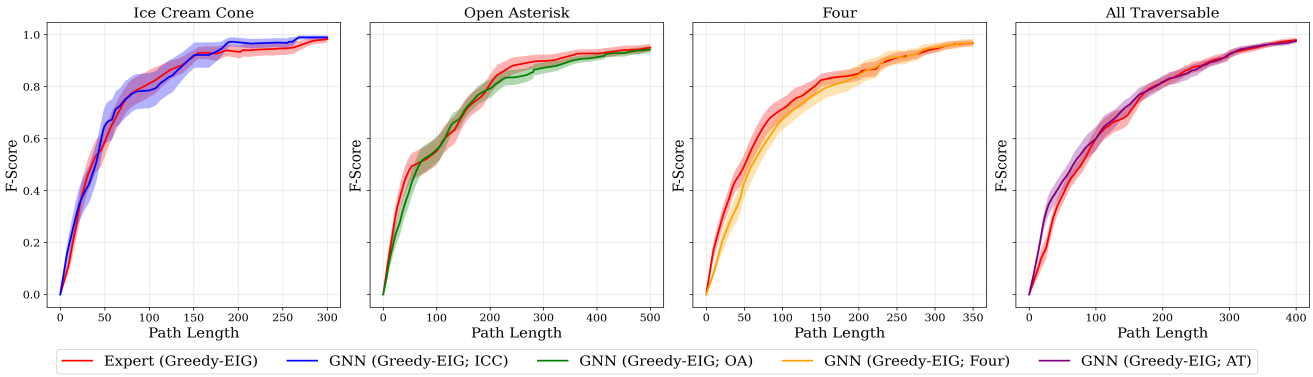

Fig. 7: GNN policies are robust to distribution shifts in object density between training and evaluation maps. Policies are trained with 20% object density and evaluated at 1% density on 30×30 maps. ICC: Ice Cream Cone, OA: Open Asterisk, Four: Figure Four, AT: All Traversable.

Method	ICC	OA	Four	AT	Mean
Expert (Greedy-EIG)	0.7960	0.7634	0.7579	0.7212	0.7596
GNN (Greedy-EIG; ICC)	0.8056	0.7425	0.7262	0.6790	0.7383
GNN (Greedy-EIG; OA)	0.7612	0.7446	0.7333	0.7186	0.7394
GNN (Greedy-EIG; Four)	0.7334	0.7655	0.7292	0.6831	0.7278
GNN (Greedy-EIG; AT)	0.7794	0.7635	0.7356	0.7366	0.7538

TABLE II: Area under the F1 curve for plots in Fig. 7. Analysing the performance of policies under distribution shift. Green signifies best for a given evaluation map (columns). ICC: Ice cream cone, OA: Open Asterisk, AT: All Traversable, Mean: Average value over each row.

C. Path EIG Expert and Policies

We examine the performance of our proposed path-integral expected information gain (Path-EIG) search method, which constitutes the core contribution of this work. Unlike the greedy approach, which only considers information gain at destination waypoints, Path-EIG evaluates the cumulative information gathered along entire trajectories (Eq. 4).

Figure 8 highlights the differences between greedy and Path-EIG experts at timesteps 2 (top) and 9 (bottom). Path-EIG favors longer, previously unvisited trajectories in early iterations and eventually converges to selecting proximal unexplored regions, which yields sparse, minimally overlapping paths. In contrast, the greedy expert lacks this bias and may repeatedly traverse the same paths, as demonstrated by the relatively cluttered trajectories.

We further visualize the evolution of the belief space as the policies progress. After nine timesteps, both methods cover approximately equal pathlengths have confirmed the locations of all OOIs, but the Path-EIG expert achieves certainty over significantly more locations, demonstrating a substantial gain in search efficiency.

In Figure 9, we present the performance of Path-EIG GNN policy trained on a 30 × 30 ICC map, evaluated against all experts evaluated on each of the four maps. While the Path-EIG policy demonstrates marked improvement over greedy expert and almost matches the expert on the ICC and OA maps, it is not able to generalize as well on the other two maps - it just tracks the greedy expert on the figure four map and is worse on the AT map. This result is not surprising since the GNN is learning a complex function and may need more parameters and data to be able to generalize universally.

VII. Field Tests

In this section, we describe the results of our field experiments with an autonomous ground vehicle in the search area shown in Fig. 1. Our goal with these field experiments is to (i) evaluate the generalization of our search policies from simulation to the real-world, (ii) compare policies specifically trained for the target search map and a policy trained on a

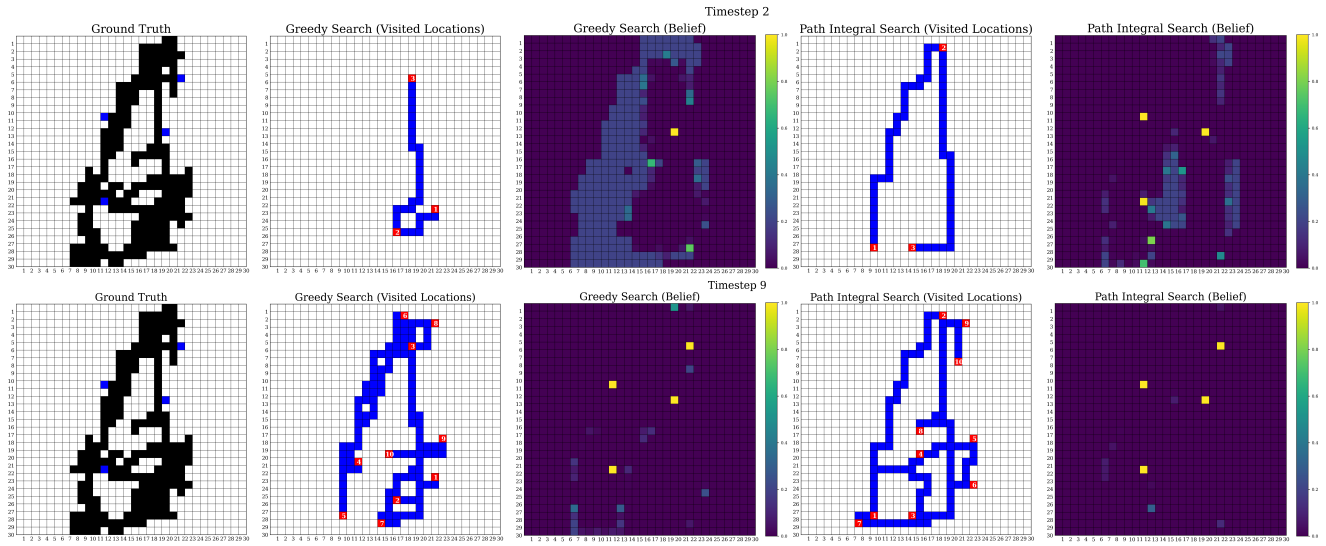

Fig. 8: Qualitative comparison of Greedy and Path-Integral experts at timesteps 2 (top) and 9 (bottom) on a shared ground truth. The leftmost column shows the search area, with traversable cells in black and ground-truth target locations in blue. For each policy and timestep, we visualize the traversed path (blue cells) and the resulting belief state; numbered red boxes indicate waypoints in selection order. Yellow indicates high object probability and purple confirmed absence. Path-Integral search produces less redundant trajectories that achieve greater information gain within the same path budget.

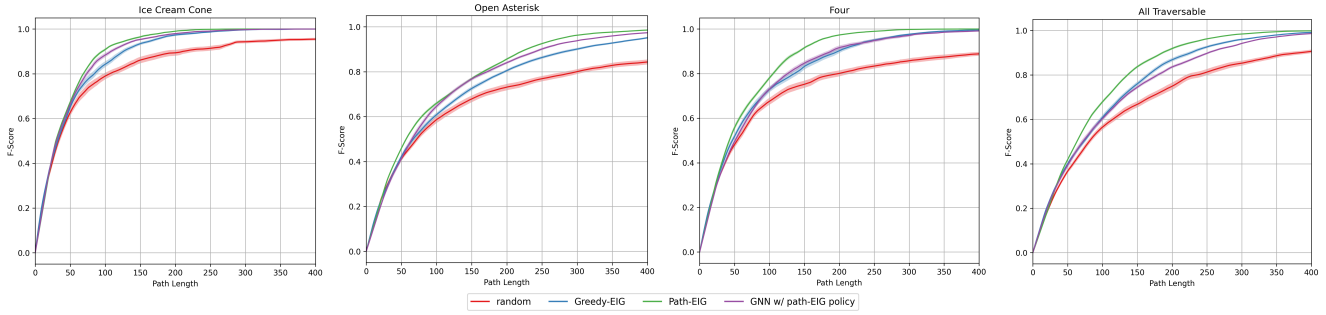

Fig. 9: Performance of the Path-Integral EIG expert and its GNN-amortized policy, compared against the Greedy-EIG expert and random and coverage baselines, across all four map geometries. The GNN policy is trained on the Ice Cream Cone map and evaluated zero-shot on the remaining maps.

generic all-traversable map, and (iii) compare our policies to the baseline NATS algorithm [18].

We begin by specifying a search polygon; the robot assumes that the prior OOI probability outside the search polygon is 0. We then randomly place 5 OOIs (mannequins) inside this search polygon. We initialize the robot randomly inside the search polygon, and run search missions for up to 15 minutes with different search policies. If the robot finds all the OOIs before the budgeted time, we terminate the mission early. To reduce variance from the testing setup, we tried to keep the OOI locations and initial robot location constant for one run of each method. Details about the robot's navigation and sensors are available in section IV.

We evaluated four policies: the NATS baseline, our GNN policies trained to imitate the greedy-EIG policy on the ice cream cone map (GNN (Greedy-EIG, ICC)) and on a generic all-traversable map (GNN (Greedy-EIG, AT)), and our GNN policy trained to imitate the path-EIG policy on the ice cream cone map (GNN (Path-EIG, ICC)).

Table III presents compelling evidence for the effectiveness of our approach. Across all three configurations (A, B, C), both GNN policies significantly outperformed the NATS baseline in terms of search completion time. The GNN (Greedy-EIG; ICC) policy, trained specifically on the target map geometry, achieved the most consistent performance with speedups ranging from 1.29× to 2.86× over NATS. Notably, in Configuration A, GNN (Greedy-EIG; ICC) com-

Config	Method	Time [s]	(mm:ss)	speedup vs. NATS
A	NATS	>900	(15:00)	1.00
	GNN (Greedy-EIG, ICC)	315	(05:15)	2.86
	GNN (Greedy-EIG, AT)	460	(07:40)	1.96
	GNN (Path-EIG, ICC)	340	(05:40)	2.64
B	NATS	663	(11:03)	1.00
	GNN (Greedy-EIG, ICC)	350	(05:50)	1.89
	GNN (Greedy-EIG, AT)	495	(08:15)	1.34
	GNN (Path-EIG, ICC)	275	(04:35)	2.41
C	NATS	425	(07:05)	1.00
	GNN (Greedy-EIG, ICC)	330	(05:30)	1.29
	GNN (Greedy-EIG, AT)	385	(06:25)	1.10
	GNN (Path-EIG, ICC)	260	(04:20)	1.63

TABLE III: Wall-clock time to achieve full recall (all 5 OOIs found) for each search policy across three experimental configurations (A, B, C) in the field test environment, and the speedup vs NATS for each configuration.

pleted the search in just 5 minutes and 15 seconds while NATS failed to find all objects within the 15-minute budget.

A key finding is the strong performance of GNN-AT, which was trained on a simple all-traversable map but deployed on the more complex test map. Despite this domain gap, GNN-AT achieved substantial speedups of 1.10× to 1.96× over NATS across all configurations. This demonstrates that our GNN policies can generalize effectively across different map geometries without requiring environment-specific training, a crucial capability for practical deployment where exact environmental conditions may not be known a priori.

VIII. Conclusion

We presented amortized path-integral policies for active search, distilling computationally expensive information-gain experts into GNN policies that run in real time. Simulation results confirm generalization across map geometries and object densities at orders-of-magnitude lower inference cost, and large-scale field trials on an autonomous ground vehicle demonstrate consistent speedups of up to 2.86× over existing baselines. Future work includes extending to multi-robot search and incorporating multi-step reinforcement learning.

IX. Acknowledgements

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. We would also like to thank Eric Haywiser for his continuous support during this project.

References

[1] C. Mouradian, J. Sahoo, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A coalition formation algorithm for multi-robot task allocation in large-scale natural disasters," in *International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2017.

[2] D. S. Drew, "Multi-agent systems for search and rescue applications," *Current Robotics Reports*, vol. 2, no. 2, pp. 189–200, 2021.

[3] K. Nagatani, Y. Okada, N. Tokunaga, S. Kiribayashi, K. Yoshida, K. Ohno, E. Takeuchi, S. Tadokoro, H. Akiyama, I. Noda *et al.*, "Multirobot exploration for search and rescue missions: A report on map building in robocuprescue 2009," *Journal of Field Robotics*, vol. 28, no. 3, pp. 373–387, 2011.

[4] J. L. Baxter, E. Burke, J. M. Garibaldi, and M. Norman, "Multi-robot search and rescue: A potential field based approach," in *Autonomous robots and agents*. Springer, 2007, pp. 9–16.

[5] R. R. Murphy, "Human-robot interaction in rescue robotics," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 138–153, 2004.

[6] O. M. Cliff, R. Fitch, S. Sukkarieh, D. L. Saunders, and R. Heinsohn, "Online localization of radio-tagged wildlife with an autonomous aerial robot system," in *Robotics: Science and Systems*, 2015.

[7] M. Dunbabin and L. Marques, "Robots for environmental monitoring: Significant advancements and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 24–39, 2012.

[8] R. Ghods, W. J. Durkin, and J. Schneider, "Multi-agent active search using realistic depth-aware noise model," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9101–9108.

[9] N. A. Bakshi, T. Gupta, R. Ghods, and J. Schneider, "GUTS: Generalized Uncertainty-Aware Thompson Sampling for Multi-Agent Active Search," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[10] S. Toscano-Palmerin and P. I. Frazier, "Bayesian optimization with expensive integrands," *SIAM Journal on Optimization*, vol. 32, no. 2, pp. 417–444, 2022.

[11] R. Marchant and F. Ramos, "Bayesian optimisation for informative continuous path planning," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6136–6143.

[12] T. M. Cabreira, L. B. Brisolara, and F. J. Paulo R, "Survey on coverage path planning with unmanned aerial vehicles," *Drones*, vol. 3, no. 1, p. 4, 2019.

[13] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1, pp. 113–126, 2001.

[14] S. Frolov, B. Garau, and J. Bellingham, "Can we do better than the grid survey: Optimal synoptic surveys in presence of variable uncertainty and decorrelation scales," *Journal of Geophysical Research: Oceans*, vol. 119, no. 8, pp. 5071–5090, 2014.

[15] H. Zhu, J. J. Chung, N. R. Lawrance, R. Siegwart, and J. Alonso-Mora, "Online informative path planning for active information gathering of a 3d surface," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1488–1494.

[16] A. Arora, P. M. Furlong, R. Fitch, S. Sukkarieh, and T. Fong, "Multi-modal active perception for information gathering in science missions," *Autonomous Robots*, vol. 43, no. 7, pp. 1827–1853, 2019.

[17] Z. W. Lim, D. Hsu, and W. S. Lee, "Adaptive informative path planning in metric spaces," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 585–598, 2016.

[18] R. Ghods, W. J. Durkin, and J. Schneider, "Multi-agent active search using realistic depth-aware noise model," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[19] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.

[20] B. Moon, S. Chatterjee, and S. Scherer, "Tigris: An informed sampling-based algorithm for informative path planning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 5760–5766.

[21] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on robot learning*. PMLR, 2020, pp. 66–75.

[22] A. Tagliabue, "Efficient imitation learning for robust, adaptive, vision-based agile flight under uncertainty," Ph.D. dissertation, Massachusetts Institute of Technology, 2024.

[23] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.

[24] C. Igoe, T. Gupta, and J. Schneider, "Efficient bayesian experiment design with equivariant networks," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[25] C. Igoe, R. Ghods, and J. Schneider, "Multi-agent active search: A reinforcement learning approach," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 754–761, 2022.

[26] N. Xu, "Active object searching on mobile robot using reinforcement learning," in *International Conference on Computing and Data Science (CDS)*, 2021.

[27] X. Ye, Z. Lin, H. Li, S. Zheng, and Y. Yang, "Active object perceiver: Recognition-guided policy learning for object searching on mobile robots," in *International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[28] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[29] I. Arasaratnam and S. Haykin, "Cubature kalman filters," *Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009.

[30] N. Peach, "Bearings-only tracking using a set of range-parameterised extended kalman filters," *Control Theory and Applications*, vol. 142, no. 1, pp. 73–80, 1995.

[31] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[32] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[33] M. Likhachev, "Search-based planning library (sbpl) including a collection of graph searches and planners that utilize these graph searches," November 2018.

[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.