Reinforcement Learning with Non-uniform State Representations for Adaptive Search

Sandeep Manjanna[1], Herke van Hoof[2] and Gregory Dudek[1]

***Abstract*— Efficient spatial exploration is a key aspect of search and rescue. In this paper, we present a search algorithm that generates efficient trajectories that optimize the rate at which probability mass is covered by a searcher. This should allow an autonomous vehicle find one or more lost targets as rapidly as possible. We do this by performing non-uniform sampling of the search region. The path generated minimizes the expected time to locate the missing target by visiting high probability regions using non-myopic path generation based on reinforcement learning. We model the target probability distribution using a classic *mixture of Gaussians* model with means and mixture coefficients tuned according to the location and time of sightings of the lost target. Key features of our search algorithm are the ability to employ a very general non-deterministic action model and the ability to generate action plans for any new probability distribution using the parameters learned on other similar looking distributions. One of the key contributions of this paper is the use of non-uniform state aggregation for policy search in the context of robotics.**

We compare the paths generated by our algorithm with other accepted spatial coverage techniques such as distribution independent boustrophedonic coverage and model dependent spiral search. We present a proof showing that rewarding for clearing probability mass instead of locating the target does not bias the objective function. The experiments show that the learned policy outperforms several well-known baselines even in scenarios different from the one it has been trained on.

I. INTRODUCTION

This paper addresses the discovery and synthesis of efficient search trajectories based on learned domain-specific policies and probabilistic inference. We also explore the use of non-uniform state space representations to enhance the performance of policy search methods.

Searching for a lost target or a person in the wilderness can be tedious, labor-intensive, imprecise, and very expensive. In some cases, the manual search and rescue operations are also unsafe for the humans involved. These reasons motivate the use of autonomous vehicles for such missions. In many search and rescue problems, timeliness is crucial. As time increases, the probability of success decreases considerably [1], and every hour the effective search radius increases by approximately 3 km [2]. We present an active sampling algorithm that generates an efficient path in real-time to search for a lost target, thus making an autonomous search and rescue mission realistic and more beneficial.

[1]Sandeep Manjanna and Gregory Dudek are with Mobile Robotics Lab (MRL), Center for Intelligent Machines, McGill University, Montreal, Canada (msandeep, dudek)@cim.mcgill.ca

[2]Herke van Hoof performed this work while at McGill, but is now with the Amsterdam Machine Learning Lab (AMLAB), University of Amsterdam, Amsterdam, Netherlands h.c.vanhoof@uva.nl

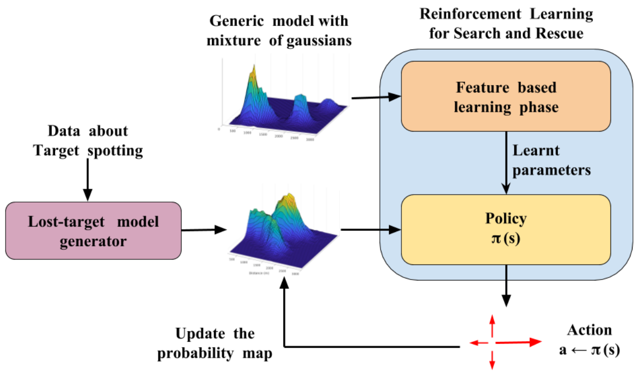

Generic model with mixture of gaussians
Reinforcement Learning for Search and Rescue
Feature based learning phase
Data about Target spotting
Learnt parameters
Lost-target model generator
Policy π (s)
Update the probability map
Action a ← π (s)

Fig. 1: Overview of our search and rescue approach.

Given an approximate region of interest, the first task is to decide where to search. Within the region of interest, some areas have higher probability of containing the target than the others. Data shows that more than 90% of searches based on a probability distribution over the region have been resolved successfully within a short duration [3]. We model the spatial field of search using classic Gaussian mixture models based on sightings of the lost target. The modeled probability distribution map over the search region is used as an input for planning the search paths. The traditional way to assign probabilities to search regions is with a distance ring model: a simple bulls-eye formed by the 25%, 50%, 75%, and 95% probability circles [3]. There are more advanced methods that propose a Bayesian model that utilizes publicly available terrain data to help model lost person behaviors enabling domain experts to encode uncertainty in their prior estimations [4]. Since modeling of the search region probability distribution is not the focus of this paper, we use a simpler approach depending only on location and time of the target sightings. Our search algorithm would perform efficiently independent of the underlying probability distribution model.

Once the probability distribution of the search region is mapped, the task is to search this region efficiently such that the regions with high probability of finding the target (*hotspots*) are searched in the early phase of the exploration. Many complete coverage approaches use line-sweep based techniques [5]–[7] which are not suitable for search and rescue tasks because of their non-timeliness. Cooper *et al.* discuss the relationship between area coverage and search effort density. They claim that any solution to the optimal search density problem is also a solution to the optimal coverage problem [8]. We present a proof showing that coverage of the search region to reduce probability mass is equivalent to searching for the target. We have previously demonstrated an algorithm to selectively cover a region based on an underlying reward distribution [9], [10]. This technique, however, does not scale up smoothly for larger regions because of the computational complexity.

In this paper, we present an adaptive sampling technique that generates efficient paths to search the missing target as fast as possible by performing non-uniform sampling of the search region. The path generated minimizes the expected time to locate the missing target by visiting high search probability regions using non-myopic path planning based on reinforcement learning. A non-myopic plan is one that accounts for possible observations that can be made in the future [11]. Fig. 1 presents an overview of the whole path generation system. Our algorithm gets trained with a generic model of the probability distribution, which can be a map generated by a Gaussian mixture. These learnt parameters are used on the generated lost target probability distribution map to come up with an action plan (*policy* π). For a given state, an action is chosen according to the probability distribution π and thus a path is generated for the searcher robot. Training the system with discounted rewards helps the planner to achieve paths that cover hotspots at the earlier stages of the search task. A major contribution of this paper is the use of non-uniform state aggregation for policy search in the context of robotics.

The key feature of our search algorithm is the ability to generate action plans for any new probability distribution using the parameters learnt on other similar looking distributions, i.e. an action planner trained on generic search and rescue distribution models is capable of planning high rewarding paths with a new probability distribution map without being trained again.

II. PROBLEM FORMULATION

The search region is a continuous two-dimensional area of interest $\mathcal{E} \subset \mathbb{R}^2$ with user-defined boundaries. The spatial search region is discretized into uniform grid cells, such that the robot's position $\mathbf{x}$ and the target's position $\mathbf{y}$ can be represented by two pairs of integers $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^2$. Each grid-cell (i, j) is assigned a prior probability value $\mathrm{q}(i, j)$ of the target being present in that cell.

The aim is to find the target in as short a time as possible. Formally, we specify this objective as maximizing $\mathbb{E}[\exp(-T/c)]$, where T is the time elapsed until the target is found and c is a condition-dependent constant. This objective reflects the assumption that the probability of the target's condition becoming bad is constant, and the aim is to locate the target in good condition.

We will specify the robot's behavior using a parametrized policy. This is a conditional probability distribution $\pi_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) = p(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})$ that maps a description of the current state $\mathbf{s}$ of the search to a distribution over possible *actions* $\mathbf{a}$. Our aim will be to automatically find good parameters $\boldsymbol{\theta}$, after which the policy can be deployed without additional training on new problems. The maximization objective then becomes:

$$\max_{\theta} J(\theta) = \max_{\theta} \mathbb{E}_{\tau_{\boldsymbol{\theta}}}[\exp(-T/c)], \tag{1}$$

where the expectation is taken with respect to trajectories $\tau_{\boldsymbol{\theta}} = (s_0, a_0, s_1, a_1, ...)$ generated by the policy $\pi_{\boldsymbol{\theta}}$.

III. MODELING OF THE SEARCH AREA

We model the spatial field of search using a classic Gaussians mixture model. The map q is initialized according to these generated prior probabilities. An example map is illustrated in Fig. 2. Our search algorithm can take any map q in accordance to the prior belief over where the target is. In this work, we will test our method on randomly generated Gaussian mixtures, but the algorithm could be trained on any type of distribution.

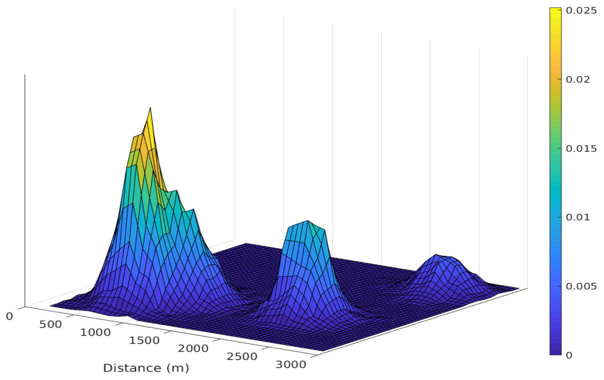


Fig. 2: Probability distribution modeled with Gaussian mixtures.

IV. POLICY GRADIENT BASED SEARCH

Finding a sequence of actions that maximizes a long-term objective could be done using dynamic programming. However, in our formulation the system state is described using a map containing the per-cell probability of the target being present and this map changes as areas are visited by the agent. The result is an extremely large state space where dynamic programming is impracticable - especially if the time to solve each particular problem is limited.

Instead, we turn to methods that directly optimize the policy parameters $\boldsymbol{\theta}$ based on (simulated) experience. To apply these methods, we will first formalize the search problem as a Markov Decision Process (MDP).

A. *Formalizing search as MDP*

A Markov Decision Process is a formal definition of a decision problem that is represented as a tuple $(S, A, T(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a_t}), r(\mathbf{s}_t, \mathbf{a}_t), \gamma)$, where S and A are the state and action space, T models transition dynamics based on the current state and action, and r defines the reward for the current state-action pair. γ is a discount factor that reduces the desirability of obtaining a reward t time-steps from now rather than in the present by γ^t. The objective is then to optimize the expected discounted cumulative reward $J = \mathbb{E}_{\tau}[\sum_{t=0}^{H} \gamma^t r_t(s_t, a_t)]$, where H is the optimization horizon.

In the proposed approach, we take the state $\mathbf{s}$ to include the position of the robot $\mathbf{x}$ as well as the map q containing the per-location presence probability for the target, $\mathbf{s} = (\mathbf{x}, \mathrm{q})$. The actions we consider are for the robot to move to and scan the cell North, East, South or West of its current location. Transitions deterministically move the agent in the desired direction. When scanning does not reveal the target, the probability mass $\mathrm{q}(i, j)$ of the current cell (i, j) is then reduced to 0[1].

The most intuitive definition of the reward function corresponding to (Eq. 1) would give the reward of 1 for recovering the target, coupled with a discount factor $\gamma = \exp(-1/c)$. However, this reward function has a high variance, as with the exact same search strategy the target could be found quickly, slowly, or not at all due to chance. Instead, we reward the robot for scanning cells with a high probability of containing the target. This does not introduce bias in the policy optimization, while reducing statistical variance[2]. A lower statistical variance typically allows optimal policies to be learned using fewer sampled trajectories.

[1]More complex models specify a probability of detection (POD) given that robot and target are in the same area [8]. For now, our work assumes a probability of detection (POD) of 1. A more realistic POD could easily be included in our approach by updating the probability mass in the cell accordingly.

Furthermore, note that as probability mass is cleared, the numbers in q no longer sum up to 1, so q is an unnormalized probability distribution.

[2]Proofs are given in the Appendix.

B. *Policy Gradient Approach*

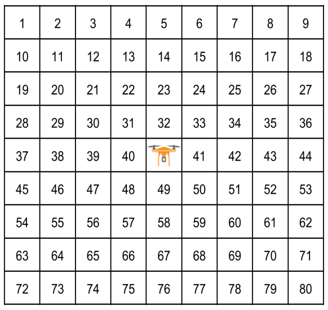
Policy gradient methods use gradient ascent for maximizing the expected return J_θ. The gradient of the expected return $(\nabla_\theta J_\theta)$ guides the direction of the parameter (θ) update. The policy gradient update is given by,

$$\theta_{k+1} = \theta_k + \eta \nabla_\theta J_\theta,$$

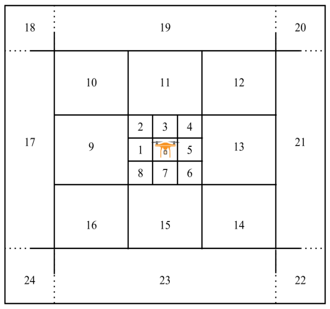
where η is the learning rate. The likelihood ratio policy gradient [12] is given by,

$$\nabla_\theta J_\theta = \int_\tau \nabla_\theta p_\theta(\tau) R(\tau) d\tau \tag{2}$$

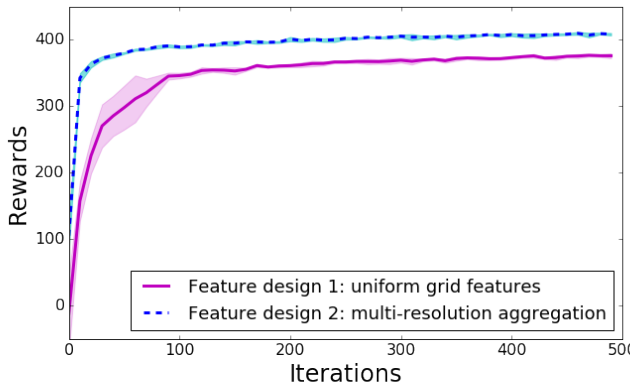
However, this expression depends on the correlation between actions and previous rewards. These terms are 0 in expectation, but cause additional variance. Ignoring these terms yields lower-variance updates, which are used in the Policy Gradient Theorem (PGT) algorithm and the GPOMDP algorithm [13]–[15]. Accordingly, the policy gradient is given



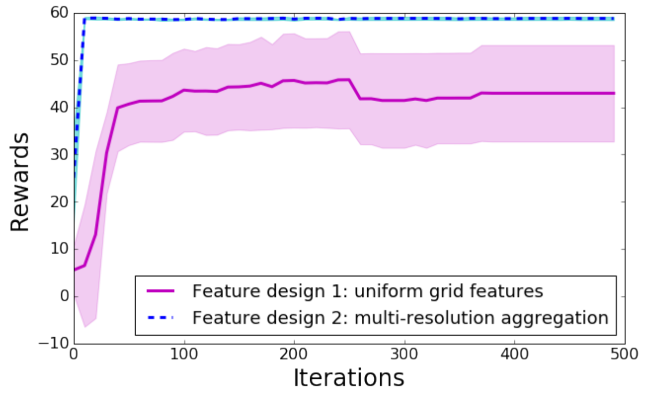
(a) Feature design 1 - centered around the robot with a map of size 9×9. This is all-grid uniform feature design.



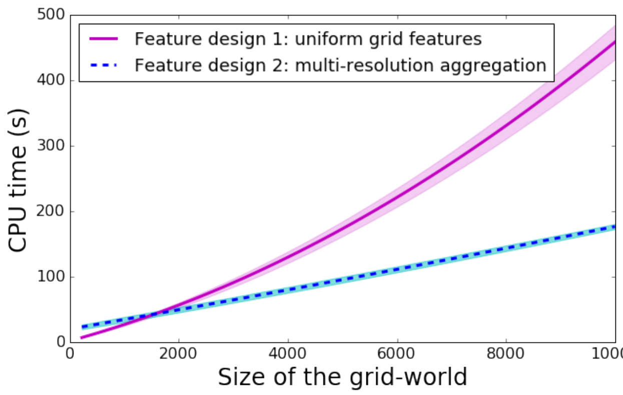
(b) Feature design 2 - centered around the robot with a map of size $n \times n$, with $n \geq 10$. This is a multi-resolution feature design.



(c) Average accumulated rewards with random starting locations



(d) Average discounted rewards with a fixed starting location



(e) CPU-time to obtain a path vs. growth in the size of the search world

Fig. 3: Results from evaluation of two different kinds of feature designs.
The shaded region indicates the standard deviation over five trials on three different sized worlds.

by,

$$\nabla_\theta J_\theta = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) \left(\sum_{j=t}^{H-1} r(s_j^{(i)}, a_j^{(i)}) - b(s_t^{(i)}) \right). \quad (3)$$

In this equation, the gradient is based on m sampled trajectories from the system, with $\mathbf{s}_j^{(i)}$ the state at the j^{th} time-step of the i^{th} sampled roll-outs. Furthermore, b is a variance-reducing baseline. In our experiments, we set it to the observed average reward.

C. Policy design

We consider a policy that is Gibbs distribution in a linear combination of features given by

$$\pi(\mathbf{s}, \mathbf{a}) = \frac{e^{\theta^T \phi_{sa}}}{\sum_b e^{\theta^T \phi_{sb}}}, \qquad \forall s \in S; a, b \in A, \quad (4)$$

where ϕ_{sa} is an l-dimensional feature vector characterizing state-action pair (s, a) and θ is an l-dimensional parameter vector. This is a commonly used policy in reinforcement learning approaches [14]. The final feature vector ϕ_{sa} is formed by concatenating a vector $\phi'_s \delta_{aa'}$ for every action $a' \in \{North, East, South, West\}$, where $\phi'_s \subset \mathbb{R}^k$ is a feature representation of the state space, and $\delta_{aa'}$ is the Kronecker delta. Thus, the final feature vector has $4 \times k$ entries, 75% of which corresponding to non-chosen actions will be 0 at any one time step.

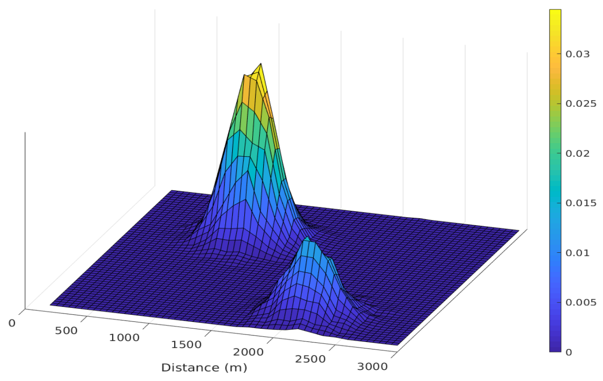
We consider two types of feature representations (ϕ'_s) for our approach. The first feature construction is to consider a vector that represents all rewards in q robot-centric, as illustrated in Fig. 3a. This feature vector grows in length as the size of the search region increases, resulting in higher computation times for bigger regions. In the second design we consider a multi-resolution feature aggregation resulting in a fixed number (24) of features irrespective of the size of the search region. In this case, features corresponding to larger cells are assigned a value equal to the average of values of $\mathrm{q}(i, j)$ that fall in that cell. In multi-resolution aggregation, the feature cells grow in size along with the distance from the robot location as depicted in Fig. 3b. Thus, areas close to the robot are represented with high resolution and areas further from the robot are represented in lower resolution. The intuition behind this feature design is that the location of nearby target probabilities is important to know exactly, while the location of faraway target probabilities can be represented more coarsely. The multi-resolution feature design is also suitable for bigger worlds as it scales logarithmically to the size of the world.

Both these feature designs were tested with the policy gradient based searching algorithm and we found that the multi-resolution aggregated features produce higher accumulated rewards and better discounted rewards (Fig. 3c, 3d). Also, the computation gets quadratically expensive as the size of the grid world increases (Fig. 3e). Based on these results, a non-uniform, multi-resolution, robot-centric feature design is more beneficial for efficient searching. These results further strengthen our belief that the immediate actions are influenced by the nearer rewards and the farther low-resolution features enhance non-myopic planning of the whole path. Theoretically, replacing the individual cell values by their averages causes some information loss. However, to plan for far-away target probabilities, a coarse location is enough. In fact, the results in Fig. 3c and Fig. 3d show that, perhaps due to the regulating effect of averaging, multi-resolution grids perform better than the uniform representation even after extensive training.

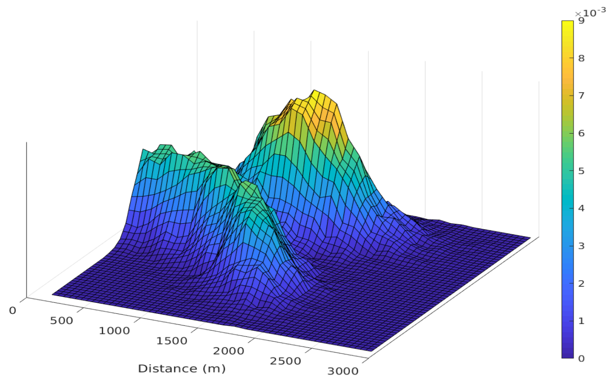
Further in this paper we will only consider the multi-resolution robot-centric feature design in our policy gradient search algorithm. The aggregated feature design is only used to achieve better policy search, but the robot action is still defined at the grid-cell level.

V. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we will introduce the experimental set-up and baseline methods, before presenting and discussing the experimental results.

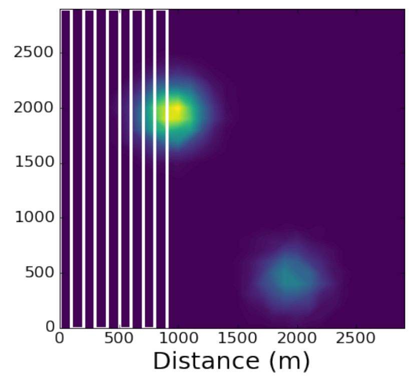


(a) test-scenario1 - mixture of Gaussians based probability distribution

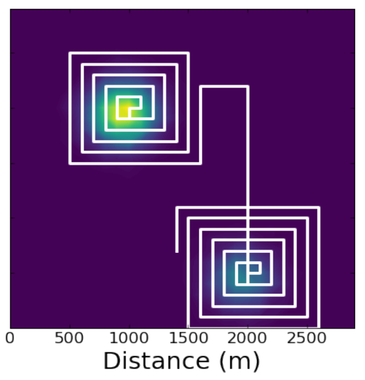


(b) test-scenario2 - non-gaussian probability distribution

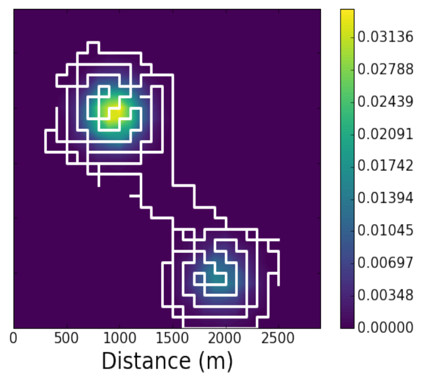
Fig. 4: Probability distributions used for testing. Colorbars indicate the probability of finding the lost target.



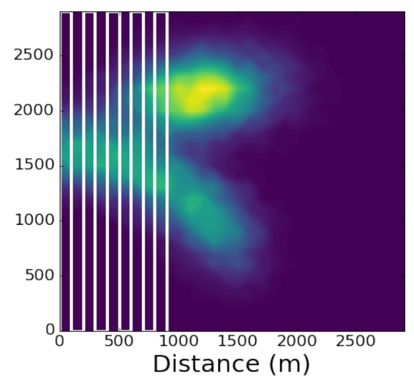
(a) Boustrophedonic coverage over test scenario 1



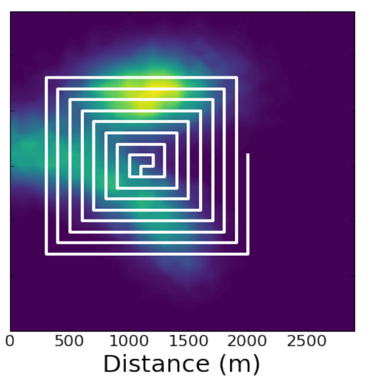
(b) Informed spiral search on test scenario 1



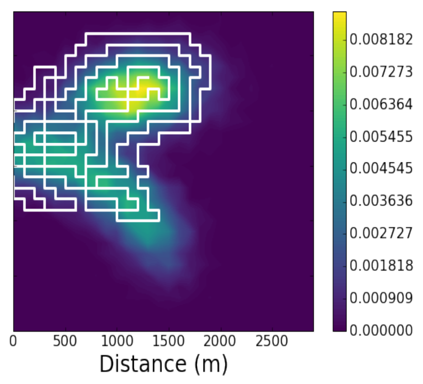
(c) Action plan generated by our method for test scenario 1



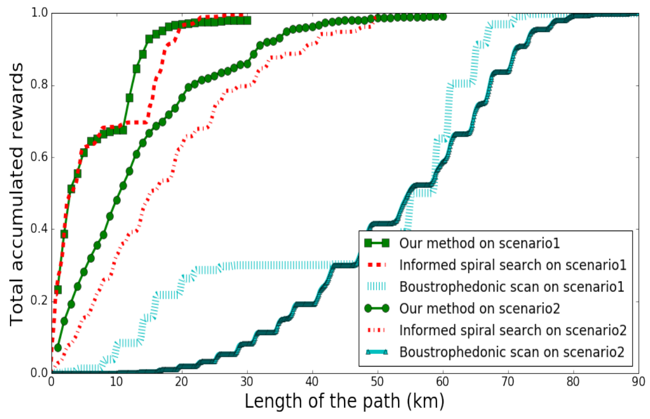
(d) Boustrophedonic coverage over the test scenario 2



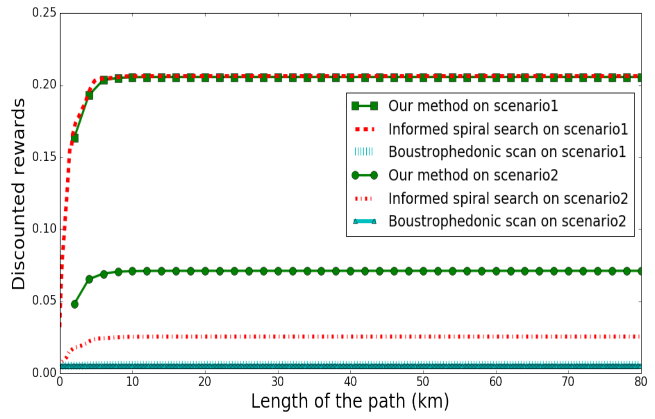
(e) Informed spiral search on test scenario 2



(f) Action plan generated by our method for test scenario 2



(g) Total accumulated rewards vs. length of the path



(h) Discounted reward vs. length of the path. For our experiments, we used a discount factor $\gamma = 0.9$

Fig. 5: Comparison of three search methods. The trajectories shown are from running the three search methods for 300 time steps.

A. Setup

We generated a generic training scenario with probability distributions for a lost target using Gaussian mixtures. We model the search space of a simulated aerial search vehicle as a 30×30 grid world with each grid-cell spanning $100m \times 100m$. We used 20 roll-outs in every iteration of the training phase. A discount factor of 0.9 was used in these experiments. During the test phase, an action with maximum probability is chosen at a given state. Fig. 2 illustrates the probability distribution grid used for training our policy based searcher. However, as mentioned in Section III, the searcher algorithm could be trained on any other type of distribution too. Two significantly different test-scenarios are presented to evaluate and compare the search algorithms. The first test-scenario (Fig. 4a) comprises of two Gaussians imitating the probability distribution of a lost target. The second test-scenario (Fig. 4b) cannot be represented as a mixture of a few Gaussians.

B. Baselines

Traditionally, exhaustive sampling of a partially observable, obstacle-free region employs a boustrophedon path [16]. The *boustrophedon* or *lawnmower path* is the approach a farmer takes when using an ox to plow a field, making back and forth straight passes over the region in alternating directions until the area is fully covered. Another efficient search pattern reported in the robotic search literature is spiral, which minimizes the time to find a single stationary lost target in several impressive idealized scenarios [17], [18].

We compare our search algorithm with these two search techniques. Fig. 5a-5f illustrate the paths generated by the three coverage algorithms overlaid on top of the test scenario probabilities. The paths shown are generated by running the search algorithms for 300 time-steps.

C. Results and discussion

Spiral search can generate efficient, or even optimal, paths under suitable conditions such as the *first test scenario* (Fig. 5b). Its satisfactory performance is only assured for a restricted class of unimodal distributions (as opposed to that in Fig. 5e). We compare these algorithms based on the rewards collected by removing the probability mass in the search region, corresponding to the (discounted) probability of finding the target. Our goal is to reduce the probability mass as fast as possible by visiting regions with high probability mass (*hotspots*). We use discounted rewards as a metric to measure the timeliness of an algorithm.

The plots in Fig. 5g illustrate how our proposed method is the fastest to cover the target probability mass on both test scenarios, even though all methods eventually visit all states of potential interest. Spiral search performs on par with our algorithm in terms of total rewards accumulated. Nonetheless, policy based searcher out performs both, spiral and boustrophedon search techniques in total discounted rewards on the *second test scenario* by a significant margin. Thus the policy based searcher exhibit a higher order of timeliness, which is a key requirement for any search algorithm to be applicable in search and rescue missions.

It is important to note that despite the dissimilarity between the training scenario and the test scenario, the policy-based search algorithm achieves better performance than the other methods.

VI. CONCLUSIONS

We presented an optimization algorithm that results in a policy that can generate non-myopic search plans for novel environments in real time. Our policy gradient based search algorithm is well suited for applications in search and rescue because of its ability to come up with a search plan on-the-go, based on the new probability distribution of the lost target, with no time wasted on retraining. Other time-critical applications like aerial surveys after a calamity, water surface surveys to monitor and contain algal blooms, and searching for remains under ocean after an accident, can use the presented algorithm to explore the region of interest.

In the near future, we would like to enhance the performance of the search algorithm by exploring different feature aggregation techniques and designing an adaptive aggregation that can combine features in real-time according to the observations made during the survey.

ACKNOWLEDGMENT

We would like to thank the the Natural Sciences and Engineering Research Council (NSERC) for their support through the Canadian Field Robotics Network (NCFRN).

APPENDIX

First, we want to show that giving a reward for clearing probability mass instead of locating the target does not bias the objective function.

Proposition 1:

$$\mathbb{E}_{\tau_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{H-1}\gamma^t r(\mathbf{s}_t,\mathbf{a}_t)\right] = \mathbb{E}_{\tau_{\boldsymbol{\theta}},\mathbf{y}}[R(\tau)]$$

where the discount factor $\gamma = \exp(-1/c)$, $\mathbf{y}$ is the target's location, and the proxy reward $r(\mathbf{s}_t,\mathbf{a}_t)$ is equal to the probability of the target being at the robot's location $\mathbf{x}$ according to q[3]. $R(\tau) = \exp(-T/c)$ with T is the time until the target is found if the target is found within H time steps, or 0 otherwise[4].

Proof: For any trajectory,

$$\begin{aligned}\mathbb{E}_{\mathbf{y}}R(\tau) &= \begin{cases}\mathbb{E}_{\mathbf{y}}\left[\exp\left(-1/c\right)^T\right] & \text{if } T \le H\\ 0 & \text{otherwise}\end{cases}\\ &= \mathbb{E}_{\mathbf{y}}\left[\sum_{t=0}^{H-1}\gamma^t\delta(\mathbf{x}_t,\mathbf{y})\right]\\ &= \sum_{t=0}^{H-1}\gamma^t\mathbb{E}_{\mathbf{y}}[\delta(\mathbf{x}_t,\mathbf{y})] = \sum_{t=0}^{H-1}\gamma^t r(\mathbf{s}_t,\mathbf{a}_t),\end{aligned}$$

where δ is the Kronecker delta. Since this equality holds for any trajectory, it must hold for a linear combination of trajectories. ■

Now, we want to show that the variance of gradient estimates using the proxy reward for clearing probability mass is lower than that of the gradient estimates using the original objectives.

Proposition 2:

$$\begin{aligned}&\mathrm{Var}\sum_{t=0}^{H-1}\nabla_\theta\log\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)\sum_{j=t}^{H-1}\gamma^j r(\mathbf{s}_j,\mathbf{a}_j)\\ &\qquad\le \mathrm{Var}\,\mathbb{E}_X\sum_{t=0}^{H-1}\nabla_\theta\log\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\sum_{j=t}^{H-1}\gamma^j\delta(\mathbf{x}_j,\mathbf{y})\end{aligned}$$

Proof: We use the definition of the variance and rearrange terms as in the GPOMDP method [13] to reformulate

[3]The reward map q is not normalized, yet after clearing a fraction g or probability mass, the probability that the target has not been found yet is $1-g$. If the target were not found yet, the normalized probability that the target is in cell (i,j) is $q(i,j)/(1-g)$. So the probability of finding the target in (i,j) indeed equals $(1-g)q(i,j)/(1-g) = q(i,j)$.

[4]The proof here assumes a static target for notational simplicity, but can be generalized to dynamic targets by making both r and $\mathbf{y}$ time-step dependent, and calculating expected values over all $\mathbf{y}_1,\ldots,\mathbf{y}_H$ jointly.

the proposition. We introduce the shorthand $r_j = r(\mathbf{s}_j, \mathbf{a}_j)$ make the equations more readable, and obtain

$$\mathbb{E}_\tau\left[\left(\sum_{j=0}^{H-1}\gamma^j r_j \sum_{t=0}^{j}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)\right)^2\right] \tag{5}$$

$$-\,\mathbb{E}_\tau\left[\sum_{j=0}^{H-1}\gamma^j r_j \sum_{t=0}^{j}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)\right]^2 \tag{6}$$

$$\leq\ \mathbb{E}_\tau\left[\left(\mathbb{E}_{\mathbf{y}}\sum_{j=0}^{H-1}\gamma^j \delta(\mathbf{x}_j,\mathbf{y}) \sum_{t=0}^{j}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)\right)^2\right] \tag{7}$$

$$-\,\mathbb{E}_\tau\left[\mathbb{E}_{X}\sum_{j=0}^{H-1}\gamma^j \delta(\mathbf{x}_j,\mathbf{y}) \sum_{t=0}^{j}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t)\right]^2 . \tag{8}$$

Note that (6) and (8) are expectations of unbiased estimates of the respective objective, and so equal the gradient of the expected value of the respective objective. By Proposition 1, these expected values are the same, so (6) and (8) cancel each other out. Writing out the squares in (5) and (7) and combining like terms yields

$$\mathbb{E}_\tau\left[\sum_{j=0}^{H-1}\sum_{i=0}^{H-1}\gamma^{i+j}\left(r_j r_i - \mathbb{E}_{\mathbf{y}}\left[\delta(\mathbf{x}_i,\mathbf{y})\delta(\mathbf{x}_j,\mathbf{y})\right]\right)S_j S_i\right] \leq 0$$

where we introduced the shorthand

$$S_j = \sum_{t=0}^{j}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\mathbf{a}_t|\mathbf{s}_t),$$

and used the fact that S_j, S_i are independent of the target location $\mathbf{y}$. Note that r_i is just the probability that $\delta(\mathbf{x}_i,\mathbf{y}) = 1$, and since the agent cannot find the target twice, $\delta(\mathbf{x}_i,\mathbf{y})\delta(\mathbf{x}_j,\mathbf{y})$ is non-zero only if $i = j$. Thus, the proposition is equivalent to

$$\mathbb{E}_\tau\left[\sum_{j=0}^{H-1}\sum_{i=0}^{H-1}\gamma^{i+j}\left(r_j r_i - \delta(i,j) r_i\right)S_j S_i\right] \leq 0. \tag{9}$$

The left-hand side of this inequality can be expressed and upper-bounded as follows:

$$\mathbb{E}_\tau\left[\sum_{j=0}^{H-1}\sum_{i=0}^{H-1}\gamma^{i+j}\left(r_j r_i - \delta(i,j) r_i\right)S_j S_i\right] \tag{10}$$

$$= \sum_{i=0}^{H-1}\sum_{j=0}^{H-1}\gamma^{i+j} r_i r_j S_i S_j - \sum_{i=0}^{H-1} r_i \gamma^{2i} S_i^2$$

$$\leq \left(2 - \sum_{k=0}^{H-1} r_k\right)\sum_{i=0}^{H-1}\sum_{j=0}^{H-1}\gamma^{i+j} r_i r_j S_i S_j - \sum_{i=0}^{H-1} r_i \gamma^{2i} S_i^2$$

$$= -\mathbb{E}_\tau\left[\sum_{i=0}^{H-1} r_i\left(\gamma^i S_i - \sum_{j=0}^{H-1}\gamma^j r_j S_j\right)^2\right]. \tag{11}$$

The inequality is due to the sum of rewards always being smaller than 1 (as the rewards denote probabilities of finding the target, and would sum up to 1 if and only if the robot visits all cells where probability mass was initially present in this trajectory). Note that the expected value in (11) is non-negative: it is a weighted sum of squared terms, where all weights are non-negative (again, due to their interpretation as probabilities). Thus, (11) is non-positive, so (10) must ≤ 0, confirming (9). ■

References

[1] L. D. Pfau, "Wilderness search and rescue: Sources and uses of geospatial information," *Master's thesis, Pennsylvania State University*, 2013.

[2] T. J. Setnicka, *Wilderness search and rescue*. Boston, US: Appalachian Mountain Club, 1980, no. 363.348 S495w.

[3] R. J. Koester, *Lost Person Behavior: A search and rescue guide on where to lookfor land, air and water.* Charlottesville, VA: dbS Productions LLC, 2008.

[4] L. Lin and M. A. Goodrich, "A Bayesian approach to modeling lost person behaviors based on terrain features in wilderness search and rescue," *Computational and Mathematical Organization Theory*, vol. 16, no. 3, pp. 300–323, 2010.

[5] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, vol. 1, 2001, pp. 27–32.

[6] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Optimal complete terrain coverage using an unmanned aerial vehicle," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011, pp. 2513–2519.

[7] C. Berger, M. Wzorek, J. Kvarnström, G. Conte, P. Doherty, and A. Eriksson, "Area coverage with heterogeneous UAVs using scan patterns," in *Proc. IEEE Int. Symp. Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 342–349.

[8] D. C. Cooper, J. R. Frost, and R. Q. Robe, "Compatibility of land sar procedures with search theory," Potomac Management Group Alexandria VA, Tech. Rep., 2003.

[9] S. Manjanna, N. Kakodkar, M. Meghjani, and G. Dudek, "Efficient terrain driven coral coverage using Gaussian processes for mosaic synthesis," in *Conf. Computer and Robot Vision (CRV)*. IEEE, 2016, pp. 448–455.

[10] S. Manjanna and G. Dudek, "Data-driven selective sampling for marine vehicles using multi-scale paths," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, September 2017, pp. 6111–6117.

[11] A. Singh, A. Krause, and W. J. Kaiser, "Nonmyopic adaptive informative path planning for multiple robots." in *IJCAI*, vol. 3, 2009, p. 2.

[12] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Reinforcement Learning*. Springer, 1992, pp. 5–32.

[13] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, vol. 15, pp. 319–350, 2001.

[14] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[15] M. P. Deisenroth, G. Neumann, J. Peters *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[16] H. Choset and P. Pignon, *Coverage Path Planning: The Boustrophedon Cellular Decomposition*. London: Springer London, 1998, pp. 203–209.

[17] S. Burlington and G. Dudek, "Spiral search as an efficient mobile robotic search technique," in *Proceedings of the 16th National Conf. on AI, Orlando Fl*, 1999.

[18] M. Meghjani, S. Manjanna, and G. Dudek, "Multi-target rendezvous search," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), 2016*, 2016, pp. 2596–2603.