Reconstructing a Spatial Field with an Autonomous Robot Under a Budget Constraint

Azin Shamshirgaran Stefano Carpin

***Abstract*— In this paper we consider the information path-planning problem for a single robot in a stochastic environment with static obstacles subject to a preassigned constraint on the distance it can travel. Given a set of candidate sampling locations, the objective is to determine a path for the robot that allows to visit as many sampling locations as possible to accurately reconstruct an unknown underlying scalar field while not exceeding the assigned travel budget. Starting from the assumption that the phenomenon being measured can be modeled by a Gaussian Process, our algorithm balances exploration and exploitation to determine a sequence of locations ensuring that a preassigned final site is reached before the budget is consumed. Using mutual information as a reward criterion, as well as a generative model to predict consumed energy, the algorithm iteratively determines where to sample next, and when to end the mission. Our findings are validated in simulation in various scenarios and lead to a better reconstruction with less failures when compared with other methods.**

I. Introduction

Autonomous robots are finding more and more applications in domains such as environmental monitoring and agriculture [1], [16]. In these applications it is often necessary to frequently collect data to monitor parameters such as water, nitrates, nutrients, or to determine soil quality, or crop estimates. These samples must be collected over vast spatial domains and the use of robots allows a more efficient use of the dwindling agricultural workforce. A key problem in this area is how to handle the tradeoff originating by the necessity to collect numerous samples while being constrained by how far a robot can travel before its battery is depleted and needs recharging. In this context, planning a path to ideally collect *the best samples* while not running out of fuel is of fundamental importance. More precisely, *best samples* means the set of samples leading to the most accurate estimate of the quantity being measured. This family of problems is known as informative path planning (IPP). Informally speaking, the goal of IPP is to define a path through a set of sampling locations allowing for the best estimate of the parameter being measured. Depending on the application, sampling locations may be preassigned (and then the robot has to select a subset of them), or may be determined on the fly. Owing to the intrinsic computational complexity of the problem, one has to settle for collecting a sub-optimal subset of samples. A distribution of spatial data can be modeled by a Gaussian Process (GP), and this approach is often used in agriculture [8], [9]. Combined with this model, metrics such as mutual information (MI), entropy, or variance can be used to guide the process of collecting samples to estimate a GP field. While the IPP problem has been extensively studied in the past, in this paper we consider a harder version where we also manage an assigned energy budget that is influenced by stochastic travel costs. This makes this problem different and harder than the orienteering problem that we have recently studied in a similar, yet simpler setting emerging in precision agriculture [11]–[14].

Our proposed solution is off-line and relies on two alternating steps to iteratively add more vertices to the path being computed. The first determines the next sample location based on the last added position, the remaining budget, and the estimated quality of the possible sample locations. Mutual information is used to rank the value of a possible location where a sample could be collected. The second step uses a rapidly exploring random tree (RRT) planner to determine a path in the environment with known obstacles to reach and visit the next location. Using a given model for stochastic energy consumption, this planned path is used to estimate the energy consumed.

The main contribution of this paper is to develop a planning based method to find the sequence of sampling locations that is more likely to determine an accurate reconstruction of the GP and to determine when to end the exploration and move to the final location or docking point. The key idea of the proposed method is managing the budget, which means that after various off-line simulations the robot will learn how to manage the traveling cost between locations and visit the more important and informative locations. Extensive simulations outline the validity of our method when compared with other alternatives and show that the produced plan allows the reconstruction of the spatial field while not exceeding the energy budget.

The rest of this paper is organized as follows. Section II discusses related works and Section III formally introduces the problem. In Section IV, we discuss the proposed method, while simulation results are given in Section V. Finally in section VI we summarize our results and discuss possible venues for future work.

II. Related Work

As path planning is an integral part of the IPP problem, sampling based path planning methods such as rapidly ex-

A. Shamshirgaran and S. Carpin are with the Department of Computer Science and Engineering, University of California, Merced, CA, USA. A. Shamshirgaran is the supported by USDA-NIFA under award # 2021-67022-33452 (National Robotics Initiative). S. Carpin is partially supported by the IoT4Ag Engineering Research Center funded by the National Science Foundation (NSF) under NSF Cooperative Agreement Number EEC-1941529. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the view of the U.S. Department of Agriculture or the National Science Foundation.

ploring random trees (RRT-RRT*), rapidly exploring random graphs (RRG*), probabilistic roadmaps (PRM*), and rapidly exploring information gathering (RIG*) have been adapted to tackle the IPP problem. In [2], the authors proposed a RIG-based algorithm for incremental information gathering. They used mutual information as an information criteria to maximize the information gathering with respect to the budget constraint. One of the limitations of these methods is the assumption of the availability of a suitable near point for graph expansion which in some cases may result in task failure. On the other hand, sampling-based methods have the advantage that they do not require to discretize the environment as they work on continuous space. In our work we use RRT for planning a path to the next selected sampling location.

An alternative approach to solving the problem of IPP is to use the model of the environment to place the sensors and plan a path for a robot. This approach is more similar to what we study in this paper, where we assume that a set of potential sampling locations is given upfront. GP is a powerful Bayesian approach used to model different phenomena in natural environments [9]. Different kernels can be used to model prior assumptions about the underlying function being estimated [7]. After modeling the spatial field, one can then plan the path based on the model. In [9], the authors study how to balance between the amount of sensing resources (e.g., number of deployed robots, energy consumption, mission time) and the quality of data collected. To this end, they formulated a constrained optimization problem imposing a bound on the variance of the estimated field. The problem is then solved finding measurement locations, planning a tour for a single robot to visit those measurement locations and finally planning tours for multiple mobile robots. GP is used to model the spatial parameter being estimated and later the traveling salesperson with neighborhoods (TSPN) is used to find the best path for a single robot to visit the location of interest. Finally, the multi robot version of the algorithm is proposed to save operating time. This solution however, does not consider energy constraints.

Another approach to solving IPP is learning based methods. In [17] the authors used Reinforcement Learning (RL) to solve the IPP problem. The problem is defined as visiting a set of sample locations with a robot with a travel budget. The nearest sample location with shortest path is selected and the reward is assigned considering the budget. However, the authors did not consider a stochastic environment and exclusively choose the shortest path for the next sampling location. In our problem, we address the selection of next sampling location with learning methods where the robot itself learns to choose the next sample point based on assigned budget, consumed and remaining energy, the information and reward it gets by visiting each location and the stochastic noise of the environment.

III. Background and Problem Formulation

A. Markov Decision Process

Markov Decision Process (MDP) is a standard modeling tool for tackling planning problems when the state is fully observable, the environment is stochastic, and rewards are additive. An MDP is defined by set of states $s \in S$, set of actions $a \in A$, a transition function $p(s'|s,a)$ and a reward function $R(s,a,s')$. The goal is to find the optimal policy $\pi^* : S \rightarrow A$ to maximize the expected return. Different return metrics have been proposed in literature. In our work we deal with an *episodic* task, i.e., the task always ends after a finite amount of time, either because the robot reaches the final location, or because it runs out of fuel. In this case it is typical to define G_t as a function of reward sequence

$$G_t = r_t + \lambda r_{t+1} + ... + \lambda^{T-t} r_T;\ 0 \leq \lambda \leq 1 \qquad (1)$$

where λ is a factor discounting future rewards and T is the time of the last action. The objective is then to produce realizations that in expectation maximize G_t. When the model of the environment is known, classic methods such as the Bellman equation and dynamic programming can be used to find the optimal policy. It is worth noting that this problem is unconstrained, i.e., the objective is to maximize G_t only, while in the problem studied in this paper we are also subject to a constraint on the traveled budget.

In model-free or direct reinforcement learning complete knowledge of the environment is not assumed. In our case, while we assume that the transition probabilities are known, the rewards are instead not known upfront (their specific formulation is described later). In such scenarios, methods such as Q-learning, SARSA, and others can be used to determine the policy [10]. Q-learning is often used because of its off-policy nature, i.e., the ability to estimate the value of taking a certain action a from state s while the system is following a preassigned policy that may be different from the optimal one. Key to Q-learning is the following update equation

$$\begin{aligned} Q(s_t,a_t) \leftarrow & Q(s_t,a_t) + \alpha[R_{t+1} \\ & + \lambda \operatorname*{argmax}_a Q(s_{t+1},a) - Q(s_t,a_t)] \end{aligned} \qquad (2)$$

where α is the learning parameter and s_t and a_t are the state and action at time t. In this case, the learned action value function directly approximate the optimal action value function Q_* irrespective of the policy being followed [10]. In our work, as detailed in section IV, we implement a similar update equation to estimate the value of visiting a certain location.

B. Informative Path Planning

Let $\mathcal{U} \subset \Re^2$ be the environment of interest. We hypothesize the presence of random disturbances characterized by a time-invariant, Markovian transition kernel that depends on the position, but not on the underlying field being sampled. The current location of the robot is defined by $s = (s^x, s^y)$. We assume the robot starts from a pre-assigned start location s_s (e.g., the point where the robot is deployed), and must

terminate its mission at the final location s_f, where it will either be retrieved or recharged. The robot is provided with n different sample locations of interest denoted by the set $\mathcal{V} = \{s_1, s_2, \ldots, s_n\}$. The robot is subject to a travel budget B limiting the set of sample locations it can visit. This constraint models, for example, the limited energy provided by the battery onboard the robot. We denote with x_g the scalar reading collected by the robot when sampling location $s_g \in \mathcal{V}$, and will denote with χ_g the random variable modeling x_g. The goal is to visit a subset of locations of $\mathcal{V}$ such that the overall travel budget is not exceeded and the accuracy of field reconstructed using the collected samples is maximized. More formally, assuming ρ is a path starting at s_s visiting a subset of locations of $\mathcal{V}$ and ending in s_f, let $f(\rho)$ be a generic function measuring the quality of the reconstructed field and $C(\rho)$ be the travel cost associated with traversing ρ. The informative path planning problem (IPP) can then be expressed as the problem of solving the following constrained optimization problem

$$\rho^* = \underset{\rho \in \psi}{\operatorname{argmax}} f(\rho) \; s.t. \, C(\rho) \leq B$$

where ψ is the set of all paths from s_s (start point) to s_f (final point). It is immediate to note that the orienteering problem is a special instance of IPP, and IPP is therefore NP-hard [9].

IV. Proposed Method

A. Mutual information

Let $\mathcal{V} \subset \mathcal{U}$ be the subset of coordinates in the domain where the robot can collect a sample. We assume that $\mathcal{V}$ is given, e.g., it may be provided by a human with expertise about the domain of operation. We associate a random variable χ_g to each location $s_g \in \mathcal{V}$. χ_g models the sample collected at s_g. Let $\mathcal{A} \subset \mathcal{V}$ be the subset of already visited locations and let $\chi_\mathcal{A}$ denote the set of random variables associated with the elements of $\mathcal{A}$. Initially $\mathcal{A} = s_s$ (the starting location), and the role of the planning algorithm is to iteratively add locations to $\mathcal{A}$ so that eventually the underlying spatial field can be accurately estimated, while at the same time ensuring that the robot does not overrrun its assigned travel budget. To avoid trivial instances, we assume that with the assigned budget the robot cannot visit all elements $\mathcal{V}$.

To asses the quality of the information gathered at a given location, and drive the sampling selection process we use mutual information (MI). MI expresses the expected reduction of entropy of the sample locations $\mathcal{V} \setminus \mathcal{A}$ which the robot has not visited yet, after considering the measurements obtained at visited locations. This is an instance of the regression problem, where we use the measured data from the visited sample locations to predict values at yet to be visited sample locations. MI can be defined as follows by using entropy and conditional entropy. The classical definition of continuous (or differential) entropy of a continuous random variable χ_g according to [6] is

$$H(\chi_g) = -\int P_{\chi_g} \log(P_{\chi_g}) \, dx$$

This definition can be naturally extended to the case of a set of variables $\chi_\mathcal{A}$. For a sample location $s_g \notin \mathcal{A}$ and a set $\mathcal{A} \subset \mathcal{V}$ we define the conditional entropy of s_g with respect to $\mathcal{A}$ as:

$$H(\chi_g|\chi_\mathcal{A}) = -\int P(\chi_g, \chi_\mathcal{A}) \log(P(\chi_g|\chi_\mathcal{A})) dx_g dx_\mathcal{A}$$

MI is then defined as

$$\mathcal{MI}(\chi_\mathcal{A}; \chi_{\mathcal{V}\setminus\mathcal{A}}) = H(\chi_{\mathcal{V}\setminus\mathcal{A}}) - H(\chi_{\mathcal{V}\setminus\mathcal{A}}|\chi_\mathcal{A})$$

Therefore, the problem of selecting the best set of sample locations $\mathcal{A}$ is equivalent to solving the following optimization problem

$$\underset{\mathcal{A}\subset\mathcal{V}}{\operatorname{argmax}} [H(\chi_{\mathcal{V}\setminus\mathcal{A}}) - H(\chi_{\mathcal{V}\setminus\mathcal{A}}|\chi_\mathcal{A})]$$

while ensuring that the robot keeps the consuming energy under the travel budget as it visits the locations in $\mathcal{A}$. In our work, $\mathcal{A}$ is not determined *globally*, but rather incrementally, i.e., by adding one sample location at the time we aim to maximally increase MI. More formally, each candidate sample location s_g will give the following contribution [3]:

$$\begin{aligned} \mathcal{MI}(\chi_\mathcal{A} \cup \chi_g) - \mathcal{MI}(\chi_\mathcal{A}) = \\ H(\chi_g|\chi_\mathcal{A}) - H(\chi_g|\chi_{\mathcal{V}\setminus\{\mathcal{A}\cup g\}}) \end{aligned} \tag{3}$$

It is well known that this approach yields suboptimal results, but this is inevitable in the general case due to the intrinsic computational complexity of the problem. To formally define how we compute Eq. 3, we resort to the assumption that the underlying field can be modeled using a GP. To make predictions at location s_g, we consider the conditional distribution $p(\chi_g = x_g|\chi_\mathcal{A} = x_\mathcal{A})$, where we condition on all sample locations $x_\mathcal{A}$ visited by the robot [4].

The multivariate normal distribution over a set $\chi_\mathcal{V}$ of random variables associated with n locations in $\mathcal{V}$ is defined as:

$$P(\chi_\mathcal{V} = x_\mathcal{V}) = \frac{1}{(2\pi)^{n/2}|\Sigma|} e^{-\frac{1}{2}(x_\mathcal{V}-\mu)^T \Sigma^{-1}(x_\mathcal{V}-\mu)} \tag{4}$$

where every location in $\mathcal{V}$ corresponds to one particular sampling location. The multivariate normal distribution is fully specified by providing a mean vector μ and a covariance matrix Σ. If we know the values collected at $\mathcal{A} \subset \mathcal{V}$, we find that for the sample location $s_g \in \mathcal{V} \setminus A$ the conditional distribution $p(\chi_g = x_g|\chi_\mathcal{A} = x_\mathcal{A})$ is a normal distribution, where mean $\mu_{g|\mathcal{A}}$ and variance $\sigma^2_{g|\mathcal{A}}$ are given by [6]:

$$\begin{aligned} \mu_{g|\mathcal{A}} &= \mu_g + \Sigma_{g\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}(x_\mathcal{A} - \mu_\mathcal{A}), \\ \sigma^2_{g|\mathcal{A}} &= \sigma^2_g - \Sigma_{g\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\Sigma_{\mathcal{A}g} \end{aligned} \tag{5}$$

where $\Sigma_{g\mathcal{A}} = \Sigma^T_{\mathcal{A}g}$ is a row vector of the variances χ_g with all variables in $\chi_\mathcal{A}$, similarly for $\Sigma_{\mathcal{A}\mathcal{A}}$. μ_g and σ^2_g are the mean and variance of χ_g.

For any finite subset $\mathcal{A} = \{s_1, s_2, ..., s_m\}$, $\mathcal{A} \subset \mathcal{V}$ of location variables, the covariance matrix $\Sigma_{\mathcal{A}\mathcal{A}}$ of the variables χ_A is obtained by

$$\Sigma_{\mathcal{A}\mathcal{A}} = \begin{bmatrix} \mathcal{K}(s_1, s_1) & \mathcal{K}(s_1, s_2) & \cdots & \mathcal{K}(s_1, s_m) \\ \vdots & \vdots & & \vdots \\ \mathcal{K}(s_m, s_1) & \mathcal{K}(s_m, s_2) & ... & \mathcal{K}(s_m, s_m) \end{bmatrix} \quad (6)$$

where $\mathcal{K}$ is the Mattérn kernel [8], which is the generalization of RBF Kernel and is parameterized by length scale $l \geq 0$, smoothness scale ν and Euclidean distance $d(.,.)$ which assumes the distance between two random variables (isotropy) and is independent of their locations (stationary). Once all the mean and covariance functions have been estimated, we can evaluate the MI criterion in Eq. 3. Using Eq. 5 and the fact that the differential entropy of a Gaussian random variable χ_g conditioned on some set of variables $\mathcal{A}$ is a monotonic function of its variance

$$\begin{aligned} H(\chi_g|\chi_\mathcal{A}) &= \frac{1}{2}\log(2\pi e\sigma^2_{g|\mathcal{A}}) \\ &= \frac{1}{2}\log(\sigma^2_{g|\mathcal{A}}) + \frac{1}{2}(\log(2\pi) + 1) \end{aligned} \quad (7)$$

we can define the value information of each candidate sample location as

$$R_g = \frac{\sigma^2_g - \Sigma_{g\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\Sigma_{\mathcal{A}g}}{\sigma^2_g - \Sigma_{g\bar{\mathcal{A}}}\Sigma^{-1}_{\bar{\mathcal{A}}\bar{\mathcal{A}}}\Sigma_{\bar{\mathcal{A}}g}} \quad (8)$$

where $\bar{\mathcal{A}} = \mathcal{V} \setminus \{\mathcal{A} \cup \{g\}\}$. We have to make sure $\sigma^2_g - \Sigma_{g\bar{\mathcal{A}}}\Sigma^{-1}_{\bar{\mathcal{A}}\bar{\mathcal{A}}}\Sigma_{\bar{\mathcal{A}}g}$ is always nonzero to guarantee the R_g is the finite number.

B. Proposed Algorithm

Starting from the framework established in the previous subsection, to select the subset of sample locations $\mathcal{A}$ while considering the travel budget, we propose a two stages algorithm[1]. The outer planning algorithm determines the next location to add to the path, while the inner algorithm plans a collision-free path to it and estimates the travel cost while accounting for the stochastic nature of the environment. Algorithm 1 shows the outer planning loop and works as follows.

The algorithm takes as input the set of candidate sampling locations $\mathcal{V}$, the start and final locations s_s and s_f, the budget B, a number of iterations M, the locations of obstacles, and parameter r (radius), ε, γ, and α.

The idea behind the algorithm is to develop a plan using an approach inspired by Q-learning, whereby for each state s we estimate $Q(s, a)$, i.e., the expected return of executing action a while in state s. In our problem, s is a robot location, i.e., one of the elements of $\mathcal{A}$, and action a represents a possible next sampling location in $\mathcal{V}\setminus\mathcal{A}$, i.e., a is an unvisited location. The reward associated with a potential sampling location (i.e., an action) is given by R_g defined in Eq. (8). These values are stored in an $n \times n$ matrix U such that $U[i, j]$ represents the estimated value of moving from s_i to s_j and collecting a sample at s_j. Assuming the robot is currently positioned at location s_i, it will pick the next location randomly among the locations in $\mathcal{V} \setminus \mathcal{A}$ within a radius r with probability ε, and with probability $1-\varepsilon$ it will pick location s_g with g defined as

$$g = \arg\max_{j \in \mathcal{V}\setminus\mathcal{A}} U[i, j]. \quad (9)$$

Constraining the choice of the next location to be within radius r ensures that the robot does not switch too frequently between far away locations, thus helping to make a more efficient use of the available budget. Parameter ε instead balances exploration and exploitation and is decreased during the execution of the algorithm to favor exploration in the beginning and then exploitation. After a location s_g is selected, in line 11 the second step of the algorithm (inner algorithm) is executed. Based on the current location s and the selected next location s_g, the RRT algorithm [5] computes an obstacle free path and estimates the energy consumed (e_T). In our implementation we use RRT, but other motion planners could be used as well. To estimate the consumed energy e_t, the algorithm uses the motion model for the robot to simulate multiple times the execution of the returned path and obtain an estimate for e_T. The

Algorithm 1 Sample Location Selection Planner

1: **Input**: $\mathcal{V}$, s_s, s_f, B, N, obstacles-list, r, γ, ε, α
2: **Output**: R_T, E_T, $\mathcal{A}$
3: Initialize matrix U with zeros
4: **for** N **iterations do**
5: $R_T \leftarrow 0$
6: $E_T \leftarrow 0$
7: $\mathcal{A} \leftarrow \{s_s\}$
8: $s \leftarrow s_s$
9: **for each** episode **do**
10: $s_g \leftarrow \begin{cases} \text{random el. in } \mathcal{V} \setminus \mathcal{A} & \text{in r with prob } \varepsilon \\ \text{element as per Eq. 9} & \text{with prob } 1-\varepsilon \end{cases}$
11: $e_T \leftarrow$ RRT(env, s, s_g, obstacles-list)
12: $E_T \leftarrow E_T + e_T$
13: **if** $E_T > B$ **then**
14: $R_T \leftarrow R_T - \Delta$
15: **else if** $E_T \leq B$ & $s_g = s_f$ **then**
16: $R_T \leftarrow R_T - (B - E_T)/K$
17: **else**
18: $R_T \leftarrow R_T + R_g$ (see Eq. 8) /dist (s, s_g)
19: **end if**
20: update U based on Eq. 10
21: $\mathcal{A} \leftarrow \mathcal{A} \cup \{s_g\}$
22: update ε
23: $s \leftarrow s_g$
24: **end for**
25: **end for**
26: **return** $\mathcal{A}$ and navigation path to travel along $\mathcal{A}$.

[1]This problem is related to orienteering, but it is however different because in orienteering one must know the value of each vertex before, while in this scenario this is not the case. This aspect will be further discussed in the experimental section.

concatenation of the routes returned by the successive calls of the inner planning loop are eventually returned by the overall planner to define how to navigate through the sequence of selected locations $\mathcal{A}$.

In the next step, the matrix U is updated as follows. First, for location s_g, we will find the best next goal based on Eq. 9 and then U is updated as follows:

$$U[i,g] = (1-\alpha)U[i,g] + \alpha(R_T + \gamma \operatorname*{argmax}_{j \in \mathcal{V}\setminus\mathcal{A}} U[g,j]) \quad (10)$$

where i is the index fo the current location and g varies over the set of all unvisited locations. The idea behind this update is to consider not only the immediate reward, but also a one-step lookahead. In Eq. (10) the current return value R_T is used. As seen in the pseudocode, R_T accumulates the rewards, but incurs a penalty when the consumed energy E_T exceeds the budget. The algorithm also penalizes runs where the robot ends with too much unused budget (line 18). One could of course use a deeper lookahead, but this would come at an increased computational cost. Studying the impact of the lookahead horizon is part of our future work. In each iteration, the algorithm assumes the robot starts from beginning location s_s and in each episode, it will add one sample location s_g. Each iteration consists of number of a number of episodes depending on the budget and will end if the consumed energy goes beyond the Budget or the robot visits the final location s_f with in a budget. Through repeated iterations, the contents of the matrix U are updated and model the value of visiting a certain location while averaging all past executions, aligned with the spirit of Q-learning.

V. Results and Discussion

To investigate the effectiveness of the proposed method, we simulate it in different scenarios with various underlying distributions of the field being measured, number of sample locations, and obstacles. Comparisons are made with three alternatives; heuristic greedy-random point selection; a Q-learning method; and an orienteering method.

The heuristic greedy-random strategy, M_{GRRRT}, selects at each stage either a random unvisited location or the closest unvisited location. More precisely, with probability $\beta > 0$ it selects a location in $\mathcal{V} \setminus \mathcal{A}$ using a uniform distribution (random option), and with probability $1-\beta$ it selects the location in $\mathcal{V} \setminus \mathcal{A}$ that is closest to the current robot location (greedy option). To account for the budget constraint, in both instances the next sample is rejected if there is not enough budget left to reach it and then move to the final location s_f. When a location is discarded, the selection method is iterated until either a valid location is found or no more candidate locations are left. In this last case, the algorithm selects s_f and tries to reach the final location. After the sample selection process, the RRT method finds an obstacle free path to the next location. The Q-learning method, M_{LQL}, chooses the next location based on Algorithm 1, and afterwards runs an offline Q-learning method to find the best obstacle free path to reach the location of choice. In this method the robot receives the penalty once it enters one of the locations considered as obstacles. Finally, the orienteering methods, M_{Or} and M_{OrwP}, determine the path that visits the most points in $\mathcal{V}$ without exceeding the preassigned budget. Note that in this case it is necessary to assign a value to each element of $\mathcal{V}$ in advance, consistently with the fact that in orienteering one must know the rewards of the vertices beforehand. In M_{Or} we set all rewards to 1 and in M_{OrwP} we set the rewards to the expectation of the GP prior at the point.

To compare the quality of the solutions produced by the different methods we use two criteria. First, we use the mean squared error (MSE) error between the predicted model and the underlying model (ground truth, unknown to each of the algorithms). To predict the model using the values collected at the sample locations we use the *scikit-learn* Python library and its GP regression library with Mattern Kernel with length scale of 1 and smoothness parameter of 1.5. The choice of the kernel and of the parameters was made after having experimentally evaluated different alternatives. The same kernel and parameters were used for all algorithms. The second criterion to evaluate the effectiveness of the algorithm is the ability to remain within the assigned budget B. Indeed, from a practical perspective it is important for the robot to reach the final docking station before running out of energy.

In the first scenario, we consider a 2D stochastic environment with obstacles spread uniformly on a lattice pattern. This choice is informed by our practical problem, i.e., collecting samples in an orchard with equally spaced fruit trees. In all cases, the start location s_s is on the bottom left (0,0) and the final location s_f is on the top right, (50,50). Four different budgets were used, namely 500, 750, 1000 and 2000. For this scenario, the set $\mathcal{V}$ consists of 100 locations, half of which are randomly dispersed in the environment, while the other half are chosen by hand in proximity of the peaks of the underlying ground truth distribution (see figure 1 (a)-(b)). The random locations are used to test robustness to inaccurate information, while the locations placed by hand are consistent with our initial assumption that a human with domain expertise would select *interesting* places to sample. With reference to algorithm 1, we set Δ to 1, K to 100, γ to 0.9, α to 0.5, r to 10 and ε starts from 0.9 and decreases to 0.2. Also for M_{LQL} and M_{GRRRT}, we set α to 0.5, λ to 0.9 and β to 0.5.

Table I summarizes the results of the three algorithms for the different budgets. For this first scenario, we do not consider the orienteering method. For each test case we present the average results produced by each of the algorithms (M_{LRRT} is the algorithm discussed in this paper, M_{GRRRT} is the heuristic greedy-random algorithm and M_{LQL} is the learning sampling selection method following by Q-learning [10]). The values displayed for N_A, E_T, R_T and MSE are limited to the cases where the robot does not exceed the assigned budget B. N_f shows how often the algorithms violate the budget constraint B (out of 25 runs). The trial counts as a failure when the robot runs out of the energy while

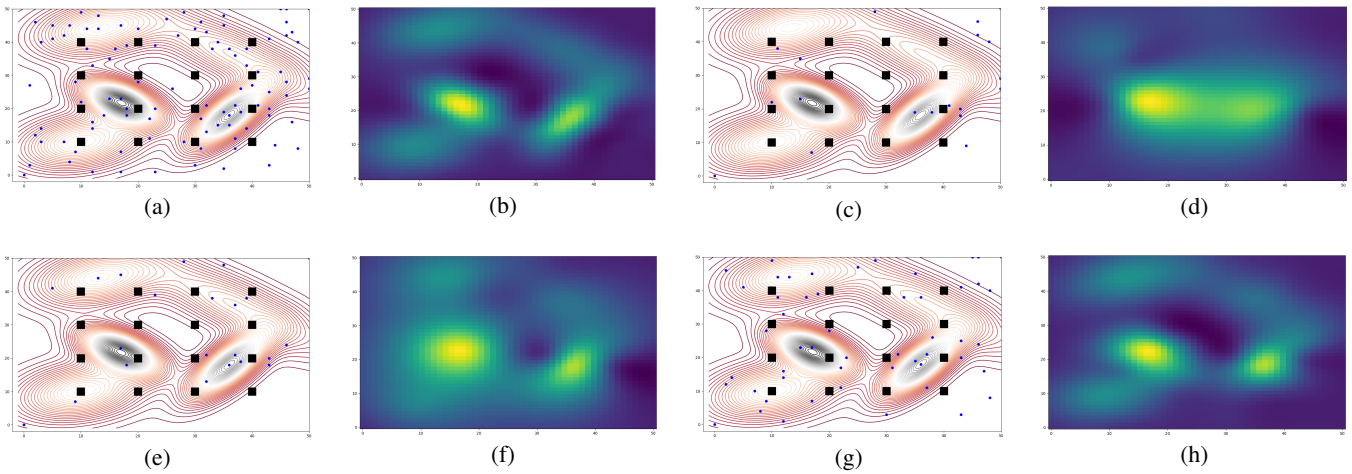

Fig. 1: Figures (a)-(b) show the underlying distribution with all sample locations and predicted distribution with all locations. Figures (c)-(d), (e)-(f) and (g)-(h) show the selected sample locations and reconstructed underlying distribution with M_{GRRRT} ($B = 750$), M_{LRRT} ($B = 750$) and M_{LRRT} ($B = 2000$).

it has not visited the final location s_g. $N_{\mathcal{A}}$ is the number of locations visited by each algorithm, E_T is the consumed energy, R_T is total reward and MSE is the mean square error. The last column provides the running time of each algorithms (with the format of minute:seconds.milliseconds.) The results illustrate that our proposed algorithm M_{LRRT} is generally faster and achieves better MSE comparing to M_{GRRRT} despite visiting a smaller number of locations on average. Importantly, M_{LRRT} consistently manages the assigned budget, while the other two algorithms have a higher failure rate. Overall, the reward column indicates the M_{LRRT} method visits the locations providing better information (in term of GP regression) which is another crucial factor in IPP problem. In cases with tight budgets ($B = 500$), this issue becomes increasingly important where the M_{LRRT} method would be able to reach better MSE and reward with fewer failures.

Fig. 1 (a)-(b) show the underlying model of the spatial data in the environment with all sample locations and the predicted model computed with all locations. Fig. 1 (c)-(d) show the selected points with the heuristic greedy-random method. Despite the robot visiting more locations, the reconstructed scalar field is less accurate. Furthermore, it runs out of energy more frequently. Subfigures (e)-(f) show the selected and visited locations with our proposed method with budget 750. It can be seen that the proposed approach selects the points widely to cover the whole area and predicts the model more accurately by visiting a sufficient number of sample locations while maintaining the budget. Subfigures (g)-(h) show the predicted model with the visited locations in M_{LRRT} method with budget 2000. It is clear from the picture that the predicted models resemble the underlying model especially it has the ability of predicting each peaks of Gaussian distribution despite the fact that it will only visit half of the sample locations.

The path generated by the M_{LRRT} algorithm with a budget 750 is shown in Figure 2, while Figure 3 shows the path generated by M_{GRRRT} under the same conditions. The figures show that the path produced in the first case more effectively balances between covering the entire space and focusing on the peaks of the underlying distribution. Moreover, the path is more regular with less switches between different sides of the environment.

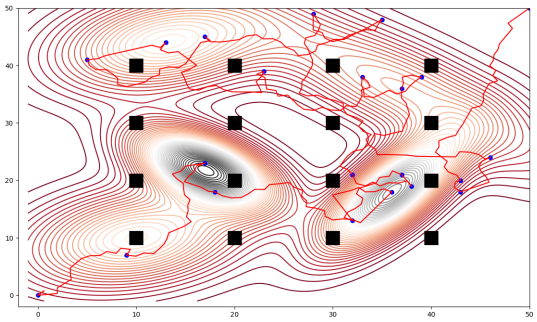

Fig. 2: Path followed by the robot running the M_{LRRT} algorithm with budget of 750.

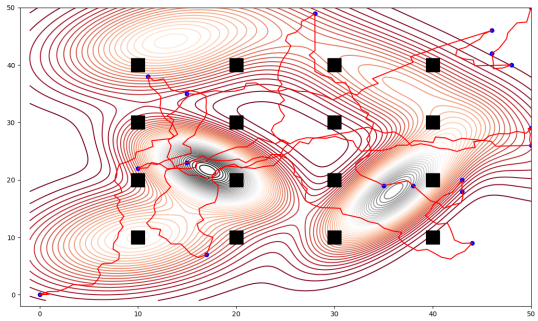

Fig. 3: Path followed by the robot running the M_{GRRRT} algorithm with budget of 750.

As a second scenario, we consider the same environment, but this time without obstacles and with a different underlying distribution (see Figure 4).

In this case our method is compared with the orienteering

Budget	methods	$N_{\mathcal{A}}$	E_T	R_T	MSE	N_f	time
500	M_{LRRT}	14.4	437.93	1.53	**0.001734**	1	0:17.23
	M_{LQL}	15.2	456.12	0.54	0.001803	1	7:62.13
	M_{GRRRT}	17.4	468.95	0.44	0.001748	3	0:28.74
750	M_{LRRT}	16.4	645.14	2.74	**0.001494**	0	0:23.38
	M_{LQL}	22.25	690.22	2.28	0.001671	1	10:45.12
	M_{GRRRT}	22.6	686.35	1.14	0.001621	3	0:35.42
1000	M_{LRRT}	25	801.13	2.69	**0.001144**	1	0:35.55
	M_{LQL}	30.7	847.24	2.47	0.001372	0	14:25.18
	M_{GRRRT}	34.5	780.90	1.40	0.001468	2	0:45.87
2000	M_{LRRT}	45.9	1764.85	3.14	0.0007221	0	01:34.69
	M_{LQL}	55.3	1842.84	3.63	**0.0005320**	0	22:38.10
	M_{GRRRT}	53.2	1743.23	2.17	0.0007980	0	01:40.74

TABLE I: Summary of results for 25 runs with different budgets. The iteration number is set to 20.

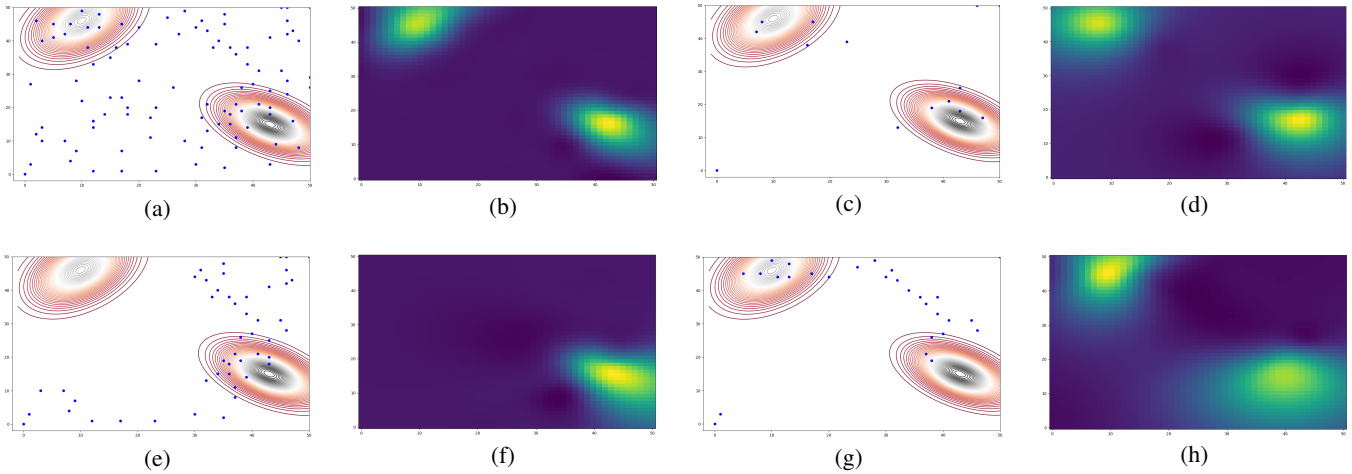

Fig. 4: Figures (a)-(b) show the underlying distribution with all sample locations and predicted distribution with all locations. Figures (c)-(d), (e)-(f) and (g)-(h) show the selected sample locations and reconstructed underlying distribution with M_{LRRT} ($B = 200$), M_{Or} ($B = 200$) and M_{OrwP} ($B = 200$).

method. M_{LRRT} represents the proposed method, while M_{Or} and M_{OrwP} are the orienterering methods formerly introduced. The orienteering problem is solved using a general purpose heuristic known as S-algorithm (the exact solution based on mixed integer linear programming is too time consuming when considering our problem instances). While the orienteering methods manages to visit a much larger number of locations in $\mathcal{V}$ they suffer two major setbacks. The first is that their number of failures are much higher. This is explained by the fact that orienteering assumes deterministic travel costs, while in our scenario these are stochastic. The second is that the subset of selected points still renders a less accurate reconstruction, as attested by the MSE column. Note that in this case our algorithm is slower because we have increased the number of iterations to 200.

Figure 4 (a) and (b) show all the sample locations in the environment and predicted model using them. Sub-figures (c) and (d) show the sample locations selected by learning based selection method M_{LRRT} and the reconstructed underlying model with budget 200. M_{LRRT} can more precisely reconstruct the underlying field even with half the locations visited compared to both orienteering methods. Sub-figures (e) and (f) show the sample locations selected by the M_{Or} method and the reconstructed underlying model. Since the robot does not reach the second peak, the reconstructed field is incomplete. Sub-figures (g) and (h) show the sample locations selected by M_{OrwP} method. In this case, the robot attempts to visit both peaks, but is unsuccessful in rebuilding the underlying field correctly which in result leads to higher MSE. The other drawback of the M_{Or} and M_{OrwP} methods is that they attempt to select sample locations near one another in densely populated areas, which results in the robot's energy depletion and its inability to cover all peaks in environments with distribution like Fig.4.

VI. Conclusions

In this paper we proposed a planning algorithm to solve the problem of information path-planning in a stochastic dynamic environment with stationary obstacles where the

Budget	methods	N_f	$N_{\mathcal{A}}$	E_T	MSE	time
100	M_{LRRT}	3	7.5	89.22	**0.00121**	3:20.78
	M_{Or}	12	23	99.54	0.00137	0:11.09
	M_{OrwP}	9	17.8	98.80	0.00128	0:10.33
200	M_{LRRT}	2	13.2	187.84	**0.00072**	5:14.08
	M_{Or}	8	38	199.03	0.00126	0:32.39
	M_{OrwP}	10	32.8	198.47	0.00111	0:29.12

TABLE II: Summary of results for 25 runs with $B = 100$ and $B = 200$. The iteration number for M_{LRRT} is set to 200.

goal is to determine a path through a set of preassigned sampling locations. The objective is to provide the best estimation of an unknown scalar field while subject to a travel budget. The algorithm determines the next sample location based on the consumed energy, budget and mutual information criteria, and then a second loop finds the best obstacle free path using the RRT algorithm to visit the designated sample location. The simulation result showed the effectiveness and application of the proposed method. Our proposed learning method predicts the underlying model more accurately than the heuristic greedy-random method and orienteering method in complex environments, with tight budget according to MSE criteria. For future work, we plan to extend our work to use multiple robots in the environment to divide the task between them and decrease the work load and increase the efficiency and run this method in the field.

REFERENCES

[1] D. V. Gealy, S. McKinley, M. Gou, L. Miller, S. Vougioukas, J. Viers, S. Carpin, and K. Goldberg. Co-robotic device for automated tuning of emitters to enable precision irrigation. In *Proceedings of the IEEE Conference on Automation Science and Engineering*, pages 922–927, 2016.

[2] M. Gh. Jadidi, J. V Miro, and G. Dissanayake. Sampling-based incremental information gathering with applications to robotic exploration and environmental monitoring. *The International Journal of Robotics Research*, 38(6):658–685, 2019.

[3] C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 265–272, 2005.

[4] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 2–10, 2006.

[5] J. Kuffner Jr. S. M. lavalle. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[6] C. E. Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[7] E. Schulz, M. Speekenbrink, and A. Krause. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

[8] M.L. Stein. *Interpolation of Spatial Data – Some Theory for Kriging*. Springer, 1999.

[9] V. Suryan and P. Tokekar. Learning a spatial field in minimum time with a team of robots. *IEEE Transactions on Robotics*, 36(5):1562–1576, 2020.

[10] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[11] T. Thayer and S. Carpin. An adaptive method for the stochastic orienteering problem. *IEEE Robotics and Automation Letters*, 6(2):4185–4192, 2021.

[12] T. Thayer and S. Carpin. A fast algorithm for stochastic orienteering with chance constraints. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021 (to appear).

[13] T. Thayer and S. Carpin. A resolution adaptive algorithm for the stochastic orienteering problem with chance constraints. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021 (to appear).

[14] T. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin. Routing algorithms for robot assisted precision irrigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2221–2228, 2018.

[15] T. Tsiligiridis. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35:797–809, 09 1984.

[16] S. Vougioukas. Agricultural robotics. *Annual review of control, robotics, and autonomous systems*, 2:339–364, 2019.

[17] Y. Wei and R. Zheng. Informative path planning for mobile sensing with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 864–873. IEEE, 2020.