Environmental Map Learning with Multiple-robots

Azin Shamshirgaran Stefano Carpin

Abstract— **This paper explores decision-making processes in robotic systems tasked with reconstructing scalar fields through sensing in uncertain environments. Each robot must handle noisy perception and operate within specific environmental and physical constraints. The complexity increases in multi-agent scenarios, where robots must not only plan their actions but also anticipate the movements and strategies of other agents. Effective coordination is crucial to prevent collisions and minimize redundant tasks. To address this challenge, we propose an online, distributed multi-robot sampling algorithm that combines Monte Carlo Tree Search (MCTS) with Gaussian regression. In this approach, each robot iteratively selects its next sampling point while exchanging limited information with other robots and predicting their future actions. Predictions about other robots future actions are computed with a MCTS that is recomputed at each iteration to incorporate all information collected up to that point. We evaluate the performance of our method across diverse environments and team sizes, comparing it to algorithmic alternatives.**

I. Introduction

The use of multi-robot teams for information gathering across multiple domains has seen a steady increase in recent years. Examples of applications include search and rescue to locate survivors [11], ocean exploration to map underwater terrain and monitor water quality parameters such as pH and chlorophyll [20], and data gathering in agricultural settings for pesticide spraying, seed sowing, as well as farm monitoring and sampling [6]. The work presented in this paper continues our ongoing research on the use of single and multi-robot systems to implement precision agriculture practices relying on accurate data collection at scale [4], [15], [16]. In particular, we focus on the implementation of budget-aware algorithms, therefore casting our problem as an instance of constrained optimization, whereby the goal is to collect *good data* (in a sense to be formally defined later), while being subject to constraints on the available energy limiting the distance a robot can travel while fulfilling its mission. The main contribution of this work relies on predicting the next movements of other robots in the team to prevent multiple robots from visiting and collecting data samples at the same locations. This is achieved by sharing only limited information, consistently with technologies currently used in precision agriculture, such as LoRA. Each robot maintains an estimate of the quantity being estimated using Gaussian Process (GP) regression – a technique that has found widespread application in this domain. For coordination, each robot relies on Monte Carlo tree search to anticipate what other robots may do based on the limited information exchanged. The remainder of this paper is organized as follows. Selected related work is presented in Section II and the necessary background information needed is given in Section III. The problem formulation for a single robot is presented in Section IV, while the multi-robot version is discussed in Section V. Extensive simulation results are presented in Section VI, and conclusions are offered in Section VII.

A. Shamshirgaran is with Omron Robotics and Safety Technologies, Inc., Pleasanton, CA, USA. S. Carpin is with the University of California, Merced, USA. A. Shamshirgaran performed this work while she was with UC Merced and was partially 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 USDA or the NSF.

II. Related Work

We review selected works addressing some of the issues we discuss in this paper. For multi-robot informative path planning, MCTS has recently gained a great deal of attention. In [2], [8], the authors proposed a method that combines Gaussian processes (GPs) and MCTS to monitor the environment, while in [14], the authors study the 2D area exploration by modifying MCTS. The goal is to minimize the time required to cover the areas of interest by first creating a tree of possible actions for the robots to take, and then choosing the one that maximizes exploration gain. To adapt to changes in the environment as it is explored, the underlying tree structure is continuously adjusted by changing the feasible actions for each node. The authors of [14] also explored the decentralized multi-robot scenario. Each robot shares its plan with another and each robot stores the shared information in a buffer. At the beginning of the next round of MCTS, the robot will choose one of each robot's shared information from the buffer and it will update the map (reward function) based on that. Among the differences between these works and ours is the fact that budget constraints do not influence the choice of actions. In [20], the goal is to plan the paths to identify hotspots in unknown 2D environments with single as well as multiple mobile robots. The spatial field of the environment is modeled using Gaussian processes whose covariance function has unknown hyperparameters. Their adaptive GP-MCTS monotonically changes the hyperparameters of the GP model to capture more complex function candidates. In the multi-robot version, they divide the environment using Voronoi partitioning whose center has been computed based on the GP model. Our method differs from these because we do not aim to only identify hotspots, but we rather aim at estimating an unknown scalar field over its entire domain. This, in turn, leads to the definition of a different reward function guiding the selection of sampling locations.

III. Preliminaries and Problem Definition

A. Informative Path Planning

By collecting samples at a finite set of locations in an area of interest $\mathcal{U} \subset \Re^2$, using a team of robots we aim to estimate a scalar function $h : \mathcal{U} \rightarrow \mathbb{R}$ which models a parameter of interest (e.g., water chlorophyll, soil moisture, etc.). As common in this domain [3], [19], the underlying physical phenomenon is modeled using a Gaussian Process (GP). We assume there are m robots in the team, each indicated as R_i with $i \in [1, 2, 3, ..., m]$. All robots in the team are equipped with identical sensors to estimate h. The location of each robot (x, y-position) is denoted by $s_s^{R_i}$. Each robot starts from a preassigned position $s_{init}^{R_i}$, and must terminate at a preassigned final location $s_f^{R_i}$ The location $s_{init}^{R_i}$ models the deployment location for robot R_i, while $s_f^{R_i}$ represents the desired end location for the robot. Each robot is assigned a predetermined travel budget B^{R_i} limiting the distance it can travel. This constraint models the limited energy provided by the robot's battery. If a robot exceeds its travel budget before reaching its assigned goal location $s_f^{R_i}$, the robot stops and this is considered a failure, as from a practical standpoint this will require someone to go and retrieve or recharge the robot in the field – a costly operation. For simplicity, we assume in the following that all robots have the same budget B^{R_i}, but the algorithms we present can generalize. A multi-robot informative path planning (IPP) method aims to select paths and sampling locations for each robot R_i that do not exceed the travel budget and maximize the quality of the reconstructed field h. In this paper, as common in literature [4], [7], we use mean square error to quantify the quality of the reconstructed field (see also Section VI.)

B. Gaussian Process Regression

To model the spatial distribution of the scalar field h, we use Gaussian Processes (GP). GPs are widely used in geostatistics [17], [19] to model environmental parameters. Considering that $x_g^{R_i}$ is the scalar reading collected by robot R_i at location s_g, and $\chi_g^{R_i}$ is the random variable modeling $x_g^{R_i}$ that follows a Gaussian distribution with mean μ_g and variance σ_g^2, GP regression algorithms can be used to estimate the posterior of h using the data collected at the sample locations (see [13] for a thorough introduction to this topic.)

C. Monte Carlo tree search (MCTS)

MCTS is widely used to solve model-based sequential stochastic decision problems [5], [21]. In the MCTS framework, one constructs a tree with a root representing the current state and edges connecting states that can be reached by executing a single action. A tree is built by adding nodes that represent states that can be reached by following a sequence of actions. Q-values are assigned to each action indicating *how good* the action will be, which is an estimation of the value that will be obtained from complete execution starting with that action. The action is selected after the tree has been constructed. As soon as the selected action is executed, the tree is discarded and rebuilt with the next state as its root.

A critical component in MCTS is the tree policy for action selection. For selection of actions, one popular criterion is the UCT rule defined in Eq. (1) (Upper Confidence bound for Trees) derived from [9] in which each candidate action a is assigned a $UCT(a)$ value defined as

$$UCT(a) = Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \tag{1}$$

In the end, the action with the highest UCT value is selected. In Eq. (1), $Q_t(a)$ is the estimate of action value based on rollouts, $N_t(a)$ is the number of times that action a has been selected before time t, and c is the exploration constant balances exploration and exploitation. Initially, $N_t(a)$ is zero for all actions[1].

IV. Single Robot Sampling

We start presenting two methods for single robot information acquisition that will then be extended for the multi-robot instance. In both cases the area of interest is partitioned into a uniform grid where each robot occupies one grid cell. We introduce two methods: the All grid MCTS algorithm (All-MCTS) and the sampling location based MCTS (SMCTS).

A. All grid MCTS algorithm (All-MCTS)

The All-MCTS method has been built upon the AdaptGP-MCTS discussed in [20]. The main difference is that the method presented in [20] operates on a continuous domain and assumes that robots can move to any point by executing a set of preassigned motion primitives. In addition, [20] does not consider a finite travel budget. The method presented in this section, instead, operates on a grid and aims at reaching the preassigned final location before the robot runs out of energy. Each robot's next sampling location is selected using the MCTS algorithm where the current location of the robot s_s is considered as the root node of the MCTS tree. Each grid cell (and therefore each robot position) has an associated children set $\Psi[s_s]$ including all locations that can be reached through the execution of a single motion action. $\Psi[s_s]$ includes all grid neighbors located one or two hops away from the current robot location (see Figure 1 where the highlighted area represents the children; one step neighbors are shown in blue and two step neighbors are in orange color.) In addition, the final location s_f is always added to the children map so that from any location robot R_i can always consider moving to the final goal location. This is useful when the travel budget is about to expire. Due to the fact that the children set includes all possible neighbors surrounding the robot's current location, we call this algorithm All-MCTS. MCTS then expands the tree by adding for each node one of the node's children.

The reward r_g associated with each neighbor potential sampling location s_g considered by robot R_i is defined as (as per the original method in [20]):

[1] $UCT(a)$ is assumed to be ∞ when $N_t(a) = 0$, thus forcing exploration.

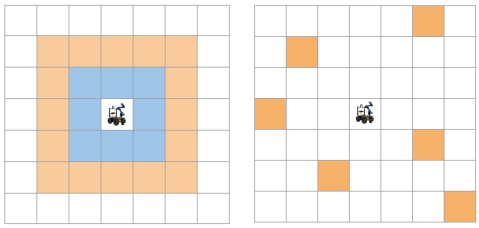

Fig. 1: Left: Robot's children set $\Psi[s_s]$ in the All-MCTS method. One step neighbors are shown in blue and two step neighbors are in orange color. Right: Candidate locations in the SMCTS method consisting of a set of locations $\mathcal{V}$ scattered in the environment.

$$r_g = \sigma + \beta^{1/2}\mu \tag{2}$$

where σ is the variance of the candidate location, μ is mean, and β is the number of measurements collected. The factor β encourages the robot to visit locations with high estimated values for h before exploring unknown areas with higher variance. The selection of the next location is performed online, i.e., the reward associated with each location is not predetermined, but re-estimated iteratively using GP regression algorithms using the data collected at the locations already visited. This means that in Eq. (2), σ and μ are the values estimated by the GP regression algorithm based on the samples collected thus far.

MCTS is iterated for a fixed number of times, and at each iteration, the path and leaf are chosen using the UCT formula defined in Eq. (1). When a leaf is reached, as per the MCTS framework, a rollout is executed to estimate the quality of the leaf, i.e., its Q value. In our implementation we use a simple random rollout, i.e., the planner continues to select additional random locations from the children map until it either reaches the final destination or runs out of energy. During the MCTS expansion and rollout, every time a candidate location is included in the tree, a generative model is used to estimate how much energy would be consumed. This estimate is given by the formula defined in Eq. (3)

$$c_s^g = \alpha d(s_s, s_g) + l \cdot \varepsilon(1, d) \tag{3}$$

where $d(s_s, s_g)$ is the Euclidean distance between the current location, s_s and candidate location s_g, ε is a random sample from a uniform distribution over the interval $[1, d]$ and l is a constant. This additional term accounts for the stochasticity in the energy consumed. After the tree $\mathcal{T}$ has been built, the next location s_g is selected based on the UCT formula and the robot moves to s_g and collects a sample. Then, the budget of the robot is updated by considering the amount of estimated energy consumed during the motion and the GP is updated based on the value read at s_g. The process continues until the robot reaches the final destination, which is a success, or it runs out of energy, which is a failure. Algorithm 1 sketches the process.

The inputs are the initial location s_{init}, the final location s_f, and the assigned budget, B. In the beginning, the set of visited locations $\mathcal{A}$ contains just the initial location. To avoid revisiting the same locations, the input of the MCTS planner is the set $\Psi[s_s] \setminus \mathcal{A}$ stored in *cand* set instead of $\Psi[s_s]$. This algorithm returns the remaining budget and the set of visited locations.

Algorithm 1 Online All-MCTS planner for robot R with limited Budget B

1: **Input**: s_{init}, s_f, B
2: $\mathcal{A} \leftarrow \{s_{init}\}$
3: $s_s \leftarrow s_{init}$
4: **while** $B > 0$ **and** $s_s \neq s_f$ **do**
5: $\quad cand \leftarrow \Psi[s_s] \setminus \mathcal{A}$
6: $\quad \mathcal{T}, s_g \leftarrow$ MCTS(s_s, $cand$)
7: $\quad$ Move to s_g, collect reading x_g, and observe consumed energy c_s^g
8: $\quad \sigma_g, \mu_g \leftarrow$ update GP with new observation x_g
9: $\quad B \leftarrow B - c_s^g$
10: $\quad \mathcal{A} \leftarrow \mathcal{A} \cup \{s_g\}$
11: $\quad s_s \leftarrow s_g$
12: **end while**
13: **return** B, $\mathcal{A}$

B. Sampling location based MCTS algorithm (SMCTS)

This method (dubbed SMCTS in the following) differs from All-MCTS in how it selects the possible next locations and in how it assigns rewards to candidate locations. The children set in All-MCTS method included all grids surrounding the current location for sampling, but the children set in this method consists of n preassigned sample locations of interest scattered in the environment that are often identified a-priori by domain experts based on past experience. In this case, we have one more input to the Algorithm 1, i.e., the set $\mathcal{V} = \{s_1, s_2, \ldots, s_n\}$ of candidate locations. The number of locations and the placements of the elements $\mathcal{V}$ is such that the robot does not have sufficient budget to visit all of them, otherwise the problem becomes trivial (see Figure 1). This setup is similar to what we considered in our former works [15], [16], and is informed by practices implemented in precision agriculture (e.g., the definition of sentinel locations to be monitored through the growing season). For SMCTS, the children set $\Psi[s_g]$ contains the locations that can be reached from s_s. Problem instances may have tens or hundreds of possible locations, so considering them all would result in search trees with extremely high branching factors, and this would be unmanageable. To minimize planning time, we limit the locations that are considered from each location in children set, which is returned by $\Psi[s_s]$. Setting M (an even number) as the maximum for the branching factor for the MCTS, $M/2$ elements in $\Psi[s_s]$ are the nearest elements in $\mathcal{V}$, while $M/2-1$ are chosen randomly from the rest. This selection balances global exploration and local exploitation. Additionally, the final location s_f is always added to children set Ψ. As in the previous method, the addition of s_f ensures that from any location the robot can always consider moving to the final location if the travel budget is about to expire. In SMCTS, each candidate location is assigned a reward r_g as

per the following formula which favors locations with high uncertainty (σ_g) and small distance:

$$r_g = \frac{\sigma_g^2}{d(s_s, s_g) + l \cdot \varepsilon(1, d)} \tag{4}$$

where $d(s_s, s_g)$ is the Euclidean distance between the current location, s_s and candidate location s_g and $l \cdot \varepsilon(1, d)$ models noise as formerly described.

V. Multi-robot extension

We now extend our algorithms for multi-robot scenarios. Since the differences between All-MCTS and SMCTS are in the children set and reward function, for brevity we will only cover Multi Robot SMCTS, with a note that Multi Robot All-MCTS is the same except for changing the two aforementioned components. Algorithm 2 sketches how the previous SMCTS algorithm can be extended for multiple robots. Our proposed method relies on predicting the next k movement of other robots to prevent two robots from visiting the same location simultaneously, which is the main contribution of this work. Simultaneous visits to the same location by multiple robots are undesirable because duplicate efforts lead to less efficient resource use. The proposed algorithm also balances exploration and exploitation based on energy consumed, remaining energy, and rewards each robot receives after visiting different locations. This will allow each robot to determine its next destination while respecting the budget constraint. Let us assume there are m robots in the team. Considering all visited locations by robot R_i and other robots $j \neq i$ that have been shared with robot R_i and all predicted locations for other robots $j \neq i$, the next location, s_g will be chosen as follows:

$$\begin{aligned} &s_g = \arg\max Q_t(s_g) \text{ for } s_g \in cand[s_s^{R_i}] \\ &\text{with} \\ &cand[s_s^{R_i}] = \Psi[s_s^{R_i}] - \mathcal{A}^{R_i} - \cup_{j=1_{j\neq i}}^{m} \mathcal{A}^{R_j} - \cup_{j=1_{j\neq i}}^{m} s_g^{R_j} \end{aligned} \tag{5}$$

where $\Psi[s_s^{R_i}]$ is the children set for the current location $s_s^{R_i}$, $\mathcal{A}^{R_i}$ is the set of visited locations by the robot itself, $\mathcal{A}^{R_j}$ is the set of visited locations by other robots, and $s_g^{R_j}$ is the predicted locations visited by the other robots in a team.

Robot R_i predicts the next decision (the next sample location that will be visited), $s_g^{R_j}$ of other robots, R_j, using the following equation

$$r_g^{R_j} = \frac{\sigma_g^2}{d(s_s^{R_j}, s_g) + l \cdot \varepsilon(1, d)} \tag{6}$$

where σ_g is variance of the candidate location $s_g \in \mathcal{V}$, and $d(s_s^{R_j}, s_g)$ is the Euclidean distance between the current location of robot R_j, $s_s^{R_j}$, and selected location, s_g, and $l \cdot \varepsilon(1, d)$ models noise. In Eq. 5, $Q_t(s_g)$ is defined as a function of the reward sequence

$$Q_t(s_g) = r_g^t + \lambda r_{g'}^{t+1} + ... + \lambda^{T-t} r_{g^T}^T;\ 0 \leq \lambda \leq 1$$

where λ is a factor discounting future rewards, T is the time of the last action, $g, g', ..., g^T$ are selected sample locations and r_g is the reward associated with the sampling location g.

Lemma 5.1: In case s_r has been predicted in step k as the next sampling location of robot j, i.e $s_g^{R_j}$, which will be eliminated from candidate locations of current location robot R_i, s_r still has a chance of being considered in candidate location of robot R_i and being chosen in step $k+m$, where $m \geq 1$. (In other words, removing the nodes from candidates in step k does not remove them permanently)

Proof: Let us consider sample location s_r is in the children set in step k, $\Psi[s_k] = \{s_l, ..., s_r, ..., s_f\}$ and in step $k+m$, $\Psi[s_{k+m}] = \{s_t, ..., s_r, ..., s_f\}$ for location s_k and s_{k+m}. Let us assume in step k, the s_r has been removed from $cand[s_k^{R_i}]$ because it has been predicted as a $s_g^{R_j}$, so $cand[s_k^{R_i}] = \{s_l, , ..., s_f\}$. In step $k+m$, if s_r has not visited by other robots and has not predicted as a next sampling location by other robots, then it will appear in the $cand[s_{k+m}^{R_i}]$ in step $k+m$ no matter that it has been removed from it previously in step k (that is due to the fact that in each step and for each location, the $\Psi[s_k]$ are independent from each other) and then it has a chance of being chosen based on Equation 6.

■

The MCTS algorithm is used to select the next location to be visited, s_g. The current location of the robot R_i, $s_s^{R_i}$ is considered as a root node of the MCTS tree.

Algorithm 2 Online Multi Robot SMCTS planner (executed by each robot R_i)

Input: $\mathcal{V}$, $s_{init}^{R_i}$, $s_f^{R_i}$, B^{R_i}
$\mathcal{A}^{R_i} \leftarrow \{s_{init}^{R_i}\}$
$s_s^{R_i} \leftarrow s_{init}^{R_i}$
while $B^{R_i} > 0$ **and** $s_s^{R_i} \neq s_f^{R_i}$ **do**
  $cand^{R_i} \leftarrow \Psi[s_s^{R_i}] \setminus \mathcal{A}^{R_i}$
  for all $j \neq i$ & $j \in m$ **do**
    estimate $s_g^{R_j}$ based on $\arg\max r_g^{R_j}$ (Eq. (4))
    $cand^{R_i} \leftarrow cand^{R_i} \setminus \mathcal{A}^{R_j}$
  end for
  $\mathcal{T}, s_g \leftarrow$ MCTS($s_s^{R_i}$, $cand^{R_i}$)
  Move to s_g, collect reading $x_g^{R_i}$, and measure consumed energy c_s^g
  $\sigma_g^2 \leftarrow$ update GP with new observation $x_g^{R_i}$
  $B^{R_i} \leftarrow B^{R_i} - c_s^g$
  $\mathcal{A}^{R_i} \leftarrow \mathcal{A}^{R_i} \cup \{s_g\}$
  $s_s^{R_i} \leftarrow s_g$
  Brodacast($s_s^{R_i}$)
end while
return B^{R_i}, $\mathcal{A}^{R_i}$

In our proposed method, each robot R_i shares its visited locations with other robots. Even though robots share their locations at each iteration, they do not share the values of the collected samples. Using the data collected at the sample locations, a posterior of h can be estimated using standard

GP regression algorithms. As this posterior was created based on local data from one robot, it remains local and is not shared. Our communication model assumes that robots can exchange limited information (such as locations) at long range. This is in line with the current technology used by robots in agricultural applications and previous work [4], [10]. In particular, LoRa [18] permits transmitting limited data at long distances. If two candidates' predicted variances are the same, adding the distance to the reward function biases the algorithm toward closer locations. In Eq. (6) and Eq. (3), noise is added in the predictive models to account for the uncertainty in travel costs.

Lemma 5.2: If the robots' current locations are different, but their distance to the candidate location is the same, they will not choose the same location.

Proof: The next location is selected based on $\operatorname{argmax} Q_t(s_g)$ for s_g where $Q_t(s_g)$ is defined as a function of reward sequence

$$Q_t(s_g) = r_g^t + \lambda r_{g'}^{t+1} + ... + \lambda^{T-t} r_{g^T}^T; \qquad 0 \leq \lambda \leq 1$$

$$Q_t(s_g) = \frac{\sigma_g^2}{d(s_s^{R_i}, s_g) + l\epsilon(1,d)} + \lambda \frac{\sigma_{g'}^2}{d(s_s^{R_i}, s_{g'}) + l\epsilon(1,d)} + ...$$

where λ is a factor discounting future rewards and T is the time of the last action. $g, g', ..., g''$ are selected sample locations and r_g is the reward associated with the sampling location g. As it can be seen, r_g^t depends on the distance metrics plus random samples from a uniform distribution, so even though the variance may be equal, the distance will not be similar since it uses random samples and the rewards will not be equal either. ■

VI. Results and Discussion

We evaluate the two proposed methods using two different datasets (see Figure 2.) The California Central Valley soil moisture dataset is a self developed dataset featuring soil moisture data manually collected and interpolated in a commercial vineyard [22]. The NASA chlorophyll concentration dataset includes measures collected on Sep 1, 2023, obtained from NASA Earth Observations from a Pacific ocean subregion [1]. The purpose of this satellite observation is to determine how much phytoplankton is growing in the ocean. The green color of phytoplankton is due to chlorophyll. Both datasets, albeit different in nature, include a scalar datafield to be estimated through measurements. To assess and compare the performance of the various algorithms, we consider three metrics. The first is the mean square error (MSE) between $\hat{h}$ (the estimate of h) and h itself:

$$MSE = \frac{1}{n} \Sigma_{i=1}^{n} (h_i - \hat{h}_i)^2 \tag{7}$$

where n is the number of cells in the grid. In our implementation, GP regression is computed using the *scikit-learn* Python library [12] and its GP regression module using Mattérn Kernel with length scale of 1 and smoothness

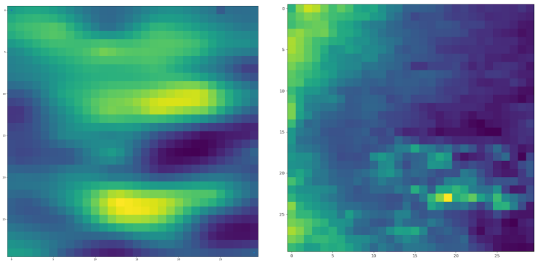

Fig. 2: Left: the scalar field modeled by the California Central Valley soil moisture dataset. Right: the scalar field modeled by the NASA chlorophyll concentration dataset. In both cases, warmer colors indicate higher values for the underlying scalar field h.

parameter of $\nu = 1.5$. The choice of the kernel and of the parameters was made after having experimentally evaluated different alternatives and having assessed that these are the best choices. Both algorithms build the posterior for h over all grid cells based on all collected measurements. As the SMCTS selects sample locations that are potentially far away while All-grid MCTS always selects nearby locations, we allow SMCTS to also collect samples along the way towards the sample location. This ensures that in both cases the number of collected samples is comparable. The second metric we consider is the remaining budget, which is the amount of energy that has not been used when the mission terminates. Ideally, this value should be low to ensure robots make the best of use of their allocated energy resources. Finally, we want to minimize the number of failures in the team. In this context, a team failure is defined as the event when at least one robot runs out of energy before reaching the final location. In the soil moisture dataset, we compare our algorithms with the algorithm proposed in [10] using their own implementation. This experimental setup is similar to what we did in [16] and allows to compare our proposed methods against established literature.

A. California Central Valley soil moisture dataset

For SMCTS, we have 100 sample locations that are distributed throughout the environment by sampling from a uniform distribution, i.e., the locations are not informed by the underlying unknown scalar field. For Ψ we set $M = 30$, while in Eq. (1) we set $c = 3$, and in Eq. (3) we use Euclidean distance. Table I summarizes the results. The table displays the budget B, the number of robots N_{R_i}, the average MSE error, and the average remaining budget for each robot B_{re}. The numerical comparison shows that SMCTS outperforms All-MCTS across the board. As the budget and number of robots increase, the performance of All-MCTS becomes similar to SMCTS. This happens because with a large budget and a large number of robots, it is possible to visit more locations without coordination. The opposite is true when the number of robots is smaller or the budget is tight, and in such cases coordination is essential. This is ensured by the different reward function used by SMCTS. Importantly, we also see that for small budgets, SMTCS outperforms MRS, while for large budgets MRS emerges as the best option. This is an important observation, as we are interested in scenarios

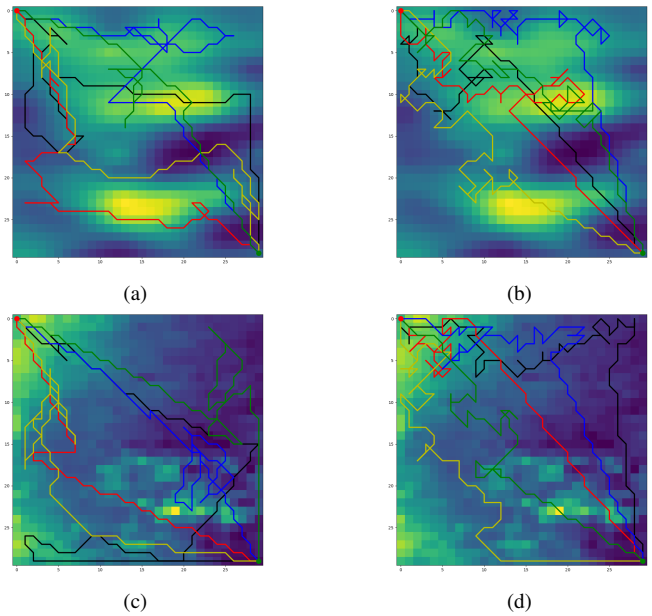

Fig. 3: Figures (a)-(b) show five-robots sampling paths with budget $B = 100$ in vineyard environment using SMCTS and All-MCTS. Figures (c)-(d) show five-robots sampling paths with budget $B = 100$ in ocean environment using SMCTS and All-MCTS.

where resources are constrained, i.e., small budgets. Note that since MRS is a deterministic method, its standard deviation is 0.

Figures 3(a) and 3(b) show the paths taken by 5 robots with $B = 100$ using SMCTS and All-MCTS, respectively. In SMCTS, the reward function encourages the robots to explore the entire environment while in All-MCTS, the reward function drives them to visit hotspot locations first, and thus they cannot explore the rest of the environment due to budget constraints. Therefore, with tighter budgets, SMCTS perform better than All-MCTS.

B. NASA chlorophyll concentration dataset

Having established the relative merit of SMCTS and All-MCTS with respect to MRS, for lack of space we here just compare SMCTS and All-MCTS. For SMCTS, again, we have 100 sample locations that are distributed throughout the environment. For the function Ψ we set $M = 30$. In Eq. (1) we set $c = 3$, and in Eq. (3) we use Euclidean distance and $k = 0.1$. Table II summarizes the metrics for All-MCTS, and SMCTS. It can be seen that SMCTS again outperforms All-MCTS with a tight budget, while All-MCTS achieves better MSE with a higher budget. Figure 3 (c) and (d) show the paths taken by 5 robots with $B = 100$ using SMCTS and All-MCTS, respectively. All-MCTS rewards the robots for only visiting hotspot locations, and since there is a hotspot at the beginning of their paths, they spend much of their time exploring it, which means they can not explore the rest of the environment due to a limited budget. Alternatively, SMCTS rewards the robot for exploring the entire environment resulting in better performance with a tight budget.

B	N_{R_i}	method	MSE (std)	$B_{re}(std)$
100	1	All-MCTS	3.81 (0.94)	8.18 (2.17)
100	1	SMCTS	3.66 (0.79)	7.91 (1.09)
100	1	MRS	3.84 (0)	8.5 (0)
100	3	All-MCTS	3.52 (0.80)	9.24 (3.41)
100	3	SMCTS	3.21 (0.71)	8.22 (2.87)
100	3	MRS	3.30 (0)	7.5 (0)
100	5	All-MCTS	2.43 (0.83)	12.96 (3.27)
100	5	SMCTS	2.30 (0.64)	10.78 (2.76)
100	5	MRS	2.37 (0)	14 (0)
200	1	All-MCTS	3.61 (0.82)	12.16 (3.04)
200	1	SMCTS	3.27 (0.70)	13.09 (3.41)
200	1	MRS	3.47 (0)	13.5(0)
200	3	All-MCTS	2.31 (0.68)	14.11 (3.81)
200	3	SMCTS	2.38 (0.73)	14.17 (4.20)
200	3	MRS	1.14 (0)	13 (0)
200	5	All-MCTS	2.35 (0.55)	13.01 (4.10)
200	5	SMCTS	2.21 (0.48)	11.90 (3.29)
200	5	MRS	0.24 (0)	12.5 (0)

TABLE I: Results for the California Central Valley soil moisture dataset.

B	N_{R_i}	method	MSE (std)	$B_{re}(std)$
100	1	All-MCTS	1.12 (0.37)	5.6 (1.74)
100	1	SMCTS	1.04 (0.29)	5.1 (1.08)
100	3	All-MCTS	0.93 (0.28)	7.18 (2.01)
100	3	SMCTS	0.84 (0.23)	8.1 (2.17)
100	5	All-MCTS	0.69 (0.17)	9.80 (3.21)
100	5	SMCTS	0.48 (0.11)	8.23 (2.84)
200	1	All-MCTS	0.91 (0.23)	9.19 (2.97)
200	1	SMCTS	0.74 (0.19)	10.45 (3.61)
200	3	ALL-MCTS	0.42 (0.16)	8.76 (2.64)
200	3	SMCTS	0.37 (0.13)	8.31 (2.26)
200	5	All-MCTS	0.28 (0.07)	11.04 (3.49)
200	5	SMCTS	0.32 (0.09)	11.27 (4.18)

TABLE II: Results for the NASA chlorophyll concentration dataset.

VII. Conclusions and Future Work

Using the MCTS algorithm, we proposed an online distributed multi-robot sampling method that scales with the number of robots in the team. The robots share their past experiences (visited sampling locations), and each robot estimates the next movement and decision of the others in order to minimize revisiting locations. Each time a sample location is measured, the GP model of the scalar field is updated. We propose a method that is more accurate, fails less frequently, and has a lower remaining budget than baseline methods. Also, our proposed methods do not require prior knowledge of the environment distribution. The next step will be to examine different fleet sizes and to test the proposed method with real robots.

REFERENCES

[1] NASA Chlorophyll concentration data-set. Available at https://neo.gsfc.nasa.gov/view.php?datasetId=MY1DMM_CHLORA (2024/04/01).

[2] G. Best, O. M Cliff, T. Patten, R. R. Mettu, and R. Fitch. Dec-mcts: Decentralized planning for multi-robot active perception. *The International Journal of Robotics Research*, 38(2-3):316–337, 2019.

[3] L. Booth and S. Carpin. Informative path planning for scalar dynamic reconstruction using coregionalized Gaussian processes and a spatiotemporal kernel. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8112–8119, 2023.

[4] L. Booth and S. Carpin. Distributed estimation of scalar fields with implicit coordination. In J. Bourgeois et al., editor, *Distributed Autonomous Robotic Systems 16.*, pages 466–478. Springer, 2024.

[5] S. Carpin and T. C. Thayer. Solving stochastic orienteering problems with chance constraints using monte carlo tree search. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 1170–1177. IEEE, 2022.

[6] A. Dechemi, D. Chatziparaschis, J. Chen, M. Campbell, A. Shamshirgaran, C. Mucchiani, A. Roy-Chowdhury, S. Carpin, and K. Karydis. Robotic assessment of a crop's need for watering. *IEEE Robotics and Automation Magazine*, 30(4):52 – 67, 2023.

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

[8] D. Jang, J. Yoo, C. Y. Son, and H. J. Kim. Fully distributed informative planning for environmental learning with multi-robot systems. *arXiv preprint arXiv:2112.14433*, 2021.

[9] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[10] S. Manjanna, M. A. Hsieh, and G. Dudek. Scalable multirobot planning for informed spatial sampling. *Autonomous Robots*, 46(7):817–829, 2022.

[11] J. Orr and A. Dutta. Multi-agent deep reinforcement learning for multi-robot applications: A survey. *Sensors*, 23(7):3625, 2023.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] C.E. Rasmussen and C.K.I. Williams. *Gaussian processes for machine learning*. MIT Press, 2005.

[14] B. Sean, L. Bartolomei, F. Kennel-Maushart, and M. Chli. Decentralised multi-robot exploration using monte carlo tree search. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robotis and Systems*, pages 7354–7361, 2023.

[15] A. Shamshirgaran and S. Carpin. Reconstructing a spatial field with an autonomous robot under a budget constraint. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8963–8970, 2022.

[16] A. Shamshirgaran, S. Manjanna, and S. Carpin. Distributed multi-robot online sampling with budget constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 12658–12664, 2024.

[17] M. L. Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 1999.

[18] J. S. P. Sundaram, W. Du, and Zh. Zhiwei. A survey on lora networking: Research problems, current solutions, and open issues. *IEEE Communications Surveys & Tutorials*, 22(1):371–388, 2019.

[19] 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.

[20] V. Suryan and P. Tokekar. Efficiently identifying hotspots in a spatially varying field with multiple robots. *arXiv preprint arXiv:2309.07981*, 2023.

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

[22] T. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin. Multi-robot routing algorithms for robots operating in vineyards. *IEEE Transactions on Automation Science and Engineering*, 17(3):1184–1194, 2020.