Decentralized Multi-Agent Exploration with Online-Learning of Gaussian Processes

Alberto Viseras[1], Thomas Wiedemann[1], Christoph Manss[1], Lukas Magel[1], Joachim Mueller[1], Dmitriy Shutin[1], Luis Merino[2]

***Abstract*— Exploration is a crucial problem in safety of life applications, such as search and rescue missions. Gaussian processes constitute an interesting underlying data model that leverages the spatial correlations of the process to be explored to reduce the required sampling of data. Furthermore, multi-agent approaches offer well known advantages for exploration. Previous decentralized multi-agent exploration algorithms that use Gaussian processes as underlying data model, have only been validated through simulations. However, the implementation of an exploration algorithm brings difficulties that were not tackle yet. In this work, we propose an exploration algorithm that deals with the following challenges: (i) which information to transmit to achieve multi-agent coordination; (ii) how to implement a light-weight collision avoidance; (iii) how to learn the data's model without prior information. We validate our algorithm with two experiments employing real robots. First, we explore the magnetic field intensity with a ground-based robot. Second, two quadcopters equipped with an ultrasound sensor explore a terrain profile. We show that our algorithm outperforms a meander and a random trajectory, as well as we are able to learn the data's model online while exploring.**

I. INTRODUCTION

A. Motivation

Exploration is a fundamental task in a wide range of applications, such as surveying, environmental analysis, or search and rescue. For these scenarios, mobile robots are often well-suited to carry sensors to relevant sampling locations. In general, it is desirable to reduce the required time and manpower for the exploration of a given environment. This naturally leads to parallelization by means of multiple robots, which form a self-coordinating multi-agent system with relatively little human supervision per robot.

Let us consider as an example a region devastated by an earthquake. In such a situation, we must explore the area as efficiently as possible to, for instance, obtain a usable map of the disaster area fast and with high resolution. We foresee the use of a swarm of intelligent agents to carry out such exploration tasks. Here, the following questions arise: i) which information should the agents communicate to coordinate themselves efficiently, and to decide where to measure; ii) how to avoid inter-agent collisions; iii) how to learn a model of the map's structure without prior information and iv) how to tackle these challenges in a decentralized manner to increase the algorithm robustness.

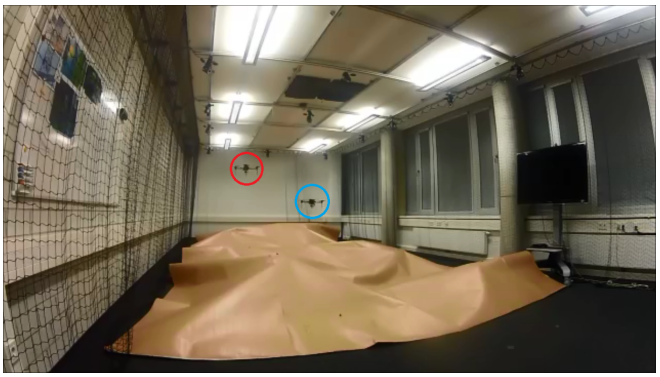


Fig. 1: Two quadcopters exploring an *a priori* unknown terrain profile with our proposed decentralized multi-agent exploration algorithm. They are equipped with an ultrasound sensor facing down to measure the distance to the floor.

In Figure 1 we show a scaled-down version of our motivation statement. Here, we explore a terrain profile with two quadcopters using our proposed algorithm.

B. Related Work

The recent advances in mobile robotics, together with the appearance of smaller and more controllable flying robots, have opened new frontiers for the development of novel exploration algorithms [1]. The trend in the robotics community points towards swarms of small robots that run light algorithms in a distributed manner. In this direction, Julian *et al.* present a coordination method for distributed robots based on the maximum informativeness criterion [2]. Here we extend their work by exploiting spatial correlations to accelerate the exploration performance.

These ideas have already been explored in several works. For example, Markov random fields have been used to model the spatial correlation of an oceanographic process with the goal of deriving a level curve tracking algorithm [3]. We are interested in probabilistic models that could allow us to learn the process' model from the acquired data. Specifically, Gaussian processes represent a powerful method to model spatial phenomena, as well as to learn the process' spatial structure. Singh *et al.* propose a procedure to define

[1]All authors are with the Institute of Communications and Navigation of the German Aerospace Center (DLR), Oberpfaffenhofen, 82234 Wessling, Germany, alberto.viserasruiz@dlr.de, thomas.wiedemann@dlr.de, christoph.manss@dlr.de, lukas.magel@dlr.de, joachim.mueller@dlr.de, dmitriy.shutin@dlr.de

[2]L. Merino is with School of Engineering, Pablo de Olavide University, Crta. Utrera km 1, Seville, Spain, lmercab@upo.es. His work is partially funded by the Junta de Andalucia through the project PAIS-MultiRobot (TIC-7390).

suitable covariance functions for Gaussian process regression in environmental surveillance applications [4]. They extend those covariance functions to perform informative path planning. Informative path planning – also called exploration, active sensing or informative sampling – is often combined with Gaussian processes due to their ability to predict the remaining uncertainty about the unobserved part of the process. Krause *et al.* study what is the optimal placement for a network of sensors in order to reduce the uncertainty of the process under study [5]. However, in their model, the sensor placements are fixed. Here, we are interested in exploration with mobile robots. This allows us to cover larger environments, requires less robots, gives us more flexibility to monitor dynamic processes, and makes the developed algorithm more robust against agent failures.

In the literature, single mobile robots have been considered for different exploration applications using Gaussian processes as models, such as SLAM [6] or environmental monitoring [7]. Online estimation of a radio signal source has been studied in the Gaussian processes context as well to model the signal propagated by the radio signal source [8]. These works only focus on exploration with a single agent. We, however, argue that the appropriate cooperation within a swarm of multiple agents could increase the performance and robustness of our exploration algorithms, as we will show in this work.

The problem of multi-robot exploration was tackled by Singh *et al.* to optimally plan the trajectories of multiple robots to explore a process [9]. This process is modeled as a pre-learned Gaussian process and the optimal exploration problem is solved in a centralized manner. Decentralization brings advantages in terms of the algorithm's robustness respect to agent failures. Chen *et al.* propose the use of Gaussian processes to monitor online traffic with multiple robots in a decentralized manner [10]. In [11], we proposed a decentralized multi-agent strategy to explore a magnetic field intensity in an indoor environment. We evaluated the algorithm's performance with respect to several covariance functions and proved its convergence and scalability with simulations. However, both of the aforementioned works assume the model's hyperparameters to be known. Ouyang *et al.* go one step further and propose an algorithm that also learns these as the process' structure [12]. They use this to actively sense an environmental phenomenon, which is modeled as a non-stationary Dirichlet process mixture of Gaussian processes. However, they validated the algorithm in simulations and did not tackle the challenges that appear in an experiment. Similar to [5], in [13] the authors propose a decentralized algorithm with mobile robots that act as elements of a sensor network to monitor a physical phenomenon in a lab environment. However, they focus on the spatio-temporal monitoring; i.e. the sensor placements are fixed and the robots must decide where to move next in order to monitor the process. The main difference with our work lies on the fact that we consider all the positions in the environment as possible sensor placements. Then, we only aim to measure in some of the locations to reconstruct the physical process of the complete environment with a high resolution. In addition, we concentrate on the online-learning of the hyperparameters and show the experimental performance of the learning process.

The aforementioned works for a decentralized multi-agent exploration algorithm are solely validated through simulations. However, the inclusion of several robots in-the-loop brings difficulties that were not yet shown in the literature. In this work, we propose an algorithm that addresses the following three questions: i) which information should the robots transmit to allow a decentralized multi-agent coordination; ii) how to implement a light-weight collision avoidance mechanism and iii) how to learn the hyperparameters online, while exploring.

The remainder of the paper is organized as follows. Section II states formally the problem. Section III summarizes the Gaussian process model used to represent the physical phenomena to be explored. We describe in Section IV the multi-agent exploration algorithm. Section V presents the experiments performed, and is then followed by the conclusions.

II. PROBLEM STATEMENT

We wish to explore an *a priori* unknown physical process as accurately as possible, in the sense of minimizing the difference between estimate and ground truth, and do so efficiently, i.e. consuming as little as possible of the limited resources such as time, energy, or communication capacity.

In this work, we assume the following:

- The borders that define the environment are *a priori* known.
- The physical process is time-invariant, such as magnetic field intensity, or terrain profile.
- The agent's position is known exactly and noise-free.
- The agents network is fully connected and the communication between agents is perfect.

We employ a swarm of N agents to explore the physical process under study. The position of the ith agent is $\mathbf{x}_i$; with $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^3$, and $i = 1, 2, ..., N$, and $\mathcal{X}$ is the environment in which the robot can operate. The physical process in position $\mathbf{x}_i \in \mathcal{X}$ is given by the variable $y_{\mathbf{x}_i} \in \mathcal{Y}$, with $\mathcal{Y} \subset \mathbb{R}$. We assume a sensor model that is described as $z_{\mathbf{x}_i} = y_{\mathbf{x}_i} + \epsilon$, where $z_{\mathbf{x}_i}$ is the measurement of the physical process taken by robot i at position $\mathbf{x}_i$ and ϵ is a noise factor that is distributed according to $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$.

III. SPATIAL GAUSSIAN PROCESS MODEL

Consider, for example, the task of measuring a terrain profile of an unknown environment, or the magnetic field intensity in an indoor environment as the process at hand. If we could model the spatial correlation of this process, we could fill spatial gaps between measurements with predictions [14]. The stronger the correlations and the better they are represented in a model, the fewer measurements are needed to achieve a certain accuracy. Gaussian processes represent a method to model physical phenomena with strong

spatial variations, like ozone concentration [7], magnetic field intensity [15], etc.

A Gaussian process [16] is a collection of random variables, any finite number of which have a multivariate Gaussian distribution. As such, it is fully specified by a mean function $\mathbf{m}(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$ for any given positions $\mathbf{x}$ and $\mathbf{x}'$.

We define the following vectors for the ith robot: 1) $\mathbf{X}_i$ is a matrix that contains all locations where the ith robot has measured $\mathbf{X}_i = \left[\mathbf{x}_i^{[1]}, \mathbf{x}_i^{[2]}, \cdots, \mathbf{x}_i^{[p]}\right]^T$; 2) $\mathbf{z}_i$ are the measurements taken at locations $\mathbf{X}_i$ according to the sensor model from section II; 3) $\mathbf{X}_{i*} = \left[\mathbf{x}_{i*}^{[1]}, \mathbf{x}_{i*}^{[2]}, \cdots, \mathbf{x}_{i*}^{[n]}\right]^T$ is a matrix composed by all positions where we aim to predict the physical process' values.

Gaussian processes are commonly used as priors in a Bayesian setting. Given $\mathbf{z}_i$ and $\mathbf{X}_i$, we can predict the target values $\mathbf{y}_{i*}$ for the corresponding $\mathbf{X}_{i*}$. The elements in $\mathbf{y}_{i*}$ are distributed according to: $p(\mathbf{y}_{i*}|\mathbf{X}_{i*}, \mathbf{X}_i, \mathbf{z}_i) = \mathcal{N}(\boldsymbol{\mu}_{i*}, \boldsymbol{\Sigma}_{i*})$. The mean vector $\boldsymbol{\mu}_{i*}$ and the covariance matrix $\boldsymbol{\Sigma}_{i*}$ of the predictive distribution are calculated as:

$$\begin{aligned} \boldsymbol{\mu}_{i*} &= \mathbf{m}(\mathbf{X}_{i*}) + \mathbf{K}_*^T \mathbf{K}^{-1}(\mathbf{z}_i - \mathbf{m}(\mathbf{X}_i)), \\ \boldsymbol{\Sigma}_{i*} &= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*, \end{aligned} \tag{1}$$

with $\mathbf{K}, \mathbf{K}_*, \mathbf{K}_{**}$ the following matrices defined from the covariance function as:

$$\begin{aligned} \mathbf{K} &= \begin{bmatrix} k(\mathbf{x}_i^{[1]}, \mathbf{x}_i^{[1]}) & \cdots & k(\mathbf{x}_i^{[1]}, \mathbf{x}_i^{[p]}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_i^{[p]}, \mathbf{x}_i^{[1]}) & \cdots & k(\mathbf{x}_i^{[p]}, \mathbf{x}_i^{[p]}) \end{bmatrix}, \\ \mathbf{K}_* &= \begin{bmatrix} k(\mathbf{x}^{[1]}, \mathbf{x}_{i*}^{[1]}) & \cdots & k(\mathbf{x}^{[1]}, \mathbf{x}_{i*}^{[n]}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}^{[p]}, \mathbf{x}_{i*}^{[1]}) & \cdots & k(\mathbf{x}^{[p]}, \mathbf{x}_{i*}^{[n]}) \end{bmatrix}, \\ \mathbf{K}_{**} &= \begin{bmatrix} k(\mathbf{x}_{i*}^{[1]}, \mathbf{x}_{i*}^{[1]}) & \cdots & k(\mathbf{x}_{i*}^{[1]}, \mathbf{x}_{i*}^{[n]}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_{i*}^{[n]}, \mathbf{x}_{i*}^{[1]}) & \cdots & k(\mathbf{x}_{i*}^{[n]}, \mathbf{x}_{i*}^{[n]}) \end{bmatrix}. \end{aligned} \tag{2}$$

The definition of the covariance function assumes the notion of similarity, which means that we expect that closer points are more likely to be similar. We focus our interest in stationary and isotropic covariance functions. In this work, we employ the squared exponential covariance function (3) because of its ability to model smooth processes, as the ones we aim to explore.

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \cdot \exp\left[\frac{-(\mathbf{x} - \mathbf{x}')^2}{2l^2}\right] + \sigma_n^2 \delta_{xx'}, \tag{3}$$

where $\delta_{xx'} = 1$ iff $x = x'$ is the Kronecker's delta.

Let us define the hyper-parameters $\boldsymbol{\theta} = [\sigma_f^2, l, \sigma_n^2]^T$ as a set of parameters that completely define the process' model. Gaussian processes represent a powerful method to learn this model from the data. Given the training data, we can compute the hyperparameters $\boldsymbol{\theta}_{i*}$ that best fit our measurements. This can be done by maximizing the log-marginal likelihood (LML) with respect to the hyper-parameters $\boldsymbol{\theta}$,

$$\boldsymbol{\theta}_{i*} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\{ \frac{1}{2} \mathbf{z}_i^T \mathbf{K}^{-1} \mathbf{z}_i + \frac{1}{2} log|\mathbf{K}| \right\}, \tag{4}$$

where $\mathbf{K}$ is a function of the hyperparameters $\boldsymbol{\theta}$ according to (2). We employ conjugate gradients for the optimization.

IV. DECENTRALIZED MULTI-AGENT EXPLORATION

We aim to explore an *a priori* unknown process. That means we should learn the process' model while exploring in order to sample at those locations that better exploit the current model. First, we present the algorithm for the single-agent exploration. Second, we extend it to allow the exploration with multiple robots in a decentralized manner.

A. Single Agent Exploration

We propose an algorithm that is based on four steps: Sense, Learn, Predict and Move (see Algorithm 1). It takes as an input the set of locations $\mathcal{X}$ over which the explored process is defined, and a stopping criterion. This stopping criterion could be defined by the user (e.g. time, battery life) or could be determined from the data (e.g. measure of remaining uncertainty). We initialize the ith robot's position as $\mathbf{x}_i^{[1]}$, and the vector of measurements $\mathbf{z}_i$ and their respective positions $\mathbf{X}_i$ as empty (line 1). Then we run the algorithm until it fulfills the stopping criterion.

Algorithm 1 `SingleAgentExploration`$(\mathcal{X}, StopCriterion)$

1: $i = 1; \mathbf{x}_i \leftarrow \mathbf{x}_i^{[1]}; \mathbf{z}_i \leftarrow NULL; \mathbf{X}_i \leftarrow NULL$
2:
3: **while** ! *StopCriterion* **do**
4: $\quad z_{\mathbf{x}_i} \leftarrow$ `Sense`$(\mathbf{x}_i)$
5: $\quad \mathbf{z}_i \leftarrow [\mathbf{z}_i; z_{\mathbf{x}_i}]$
6: $\quad \mathbf{X}_i \leftarrow [\mathbf{X}_i; \mathbf{x}_i^T]$
7: $\quad \boldsymbol{\theta}_{i*} \leftarrow$ `LearnHyp`$(\mathbf{z}_i, \mathbf{X}_i)$
8: $\quad \mathbf{X}_{i*} \leftarrow$ `CalcNeighbors`$(\mathcal{X}, \mathbf{x}_i)$
9: $\quad \boldsymbol{\mu}_{i*}, \boldsymbol{\Sigma}_{i*} \leftarrow$ `PredictGP`$(\mathbf{z}_i, \mathbf{X}_i, \mathbf{X}_{i*}, \boldsymbol{\theta}_{i*})$
10: $\quad \mathbf{x}_{next} \leftarrow \underset{\mathbf{x} \in \mathbf{X}_{i*}}{\operatorname{argmax}}(\boldsymbol{\sigma}_{i*}^2)$
11: $\quad \mathbf{x}_i \leftarrow$ `MoveTo`$(\mathbf{x}_{next})$

First, we take several measurements at position $\mathbf{x}_i$ and filter them to obtain $z_{\mathbf{x}_i}$ (line 4). This filtering step allows us to mitigate the sensor noise and to reject possible outliers.

Second, we incorporate the new measurement and its location into our measurements' vector and positions' vector (lines 5,6). We use those measurements to learn the model's parameters that best represent our data (line 7). The better the model, the better we will be able to predict the most interesting areas to explore next. In our Gaussian process model, learning the model is equivalent to learning the optimal hyperparameters that characterize the covariance function. This can be done with equation (4).

Next, we select the set of positions where the agent can move in the current iteration (line 8). Since we are interested in learning the local correlation between measurements, we consider a greedy approach to select the agent's next

position. Therefore, the matrix of possible next positions $\mathbf{X}_{i*}$ corresponds to those located within a sphere $B_r(\mathbf{x}_i)$[1], of radius r equal to the desired measurements' resolution[2] and centered at the agent's position $\mathbf{x}_i$:

$$\mathbf{X}_{i*} = \{\mathbf{x} \in \mathcal{X} \mid \mathbf{x} \in B_r(\mathbf{x}_i)\}. \quad (5)$$

We predict the process' vector of means $\boldsymbol{\mu}_{i*}$ and covariance matrix $\boldsymbol{\Sigma}_{i*}$ in the positions $\mathbf{X}_{i*}$ using the Gaussian process model learned in line 7. These vectors of means and variances are calculated according to equation (1), where the inputs are the vector of measurements $\mathbf{z}_i$ and their respective locations $\mathbf{X}_i$ (line 9). In [11], we showed that the locality of the predictions does not affect the algorithm's performance. Therefore, here we just perform the prediction in a local area centered at the agent's position in order to preserve the scalability with respect to the number of measurements.

The predicted variance is a measure of the process entropy [17]. We aim to sample where the entropy is highest; i.e. at the most informative location. Therefore, we move to the position $\mathbf{x}_{next} \in \mathbf{X}_{i*}$ with the highest predicted uncertainty/variance (line 10). The variance is given by vector $\boldsymbol{\sigma}^2_{i*}$ that is composed of the elements of the main diagonal of matrix $\boldsymbol{\Sigma}_{i*}$. Each of the elements of vector $\boldsymbol{\sigma}^2_{i*}$ represents the variance at positions $\mathbf{x}_i^{[j]} \in \mathbf{X}_{i*}$.

The robot continues running the algorithm by moving towards positions with high uncertainty. Once the algorithm stops, the collected measurements are used to predict the process value at any possible location in $\mathcal{X}$. This also suppresses the measurement noise. The prediction gives us the mean and variance of the distribution at all positions $\mathbf{x} \in \mathcal{X}$. We select the mean of each of the distributions as our exploration result of the physical process under study.

B. Multi-Agent Exploration

Now, we extend the single-agent exploration algorithm to handle multiple robots (see Algorithm 2). The proposed algorithm is able to coordinate the robots while avoiding inter-agent collisions. This algorithm runs in a decentralized manner, such that each of the agents does its own decisions based on the current information. This information consists of the single measurements taken by the agents, together with their respective locations and the positions where the agents are heading to. They correspond to an exchange of ten scalar values per agent in a 3-dimensional environment, which makes the information exchange feasible from the communication perspective. Multi-agent coordination by exchanging a small amount of data is possible due to the fact that agents share the same data model; i.e. they all employ the same mean function and covariance function for the Gaussian processes regression and learning. In the following, we explain the algorithm in detail.

Let's consider N robots for the exploration, with each robot i starting at a different position $\mathbf{x}_i^{[1]}$. The vectors of measurements and measurements' positions of each of the robots are initialized as an empty set (line 1). Then, each individual robot runs its own algorithm and communicates with the other robots until the stopping criterion is fulfilled. It is important to remark that the *for-loop* runs in parallel in each of the agents. However, the instructions contained in the *while-loop* run sequentially for each of the N agents. We assume as well that the agents are timely synchronized.

Algorithm 2 `MultiAgentExploration`$(\mathcal{X}, StopCriterion)$

1: $\{\mathbf{x}_i\}_{i=1}^N \leftarrow \mathbf{x}_i^{[1]}; \{\mathbf{z}_i\}_{i=1}^N \leftarrow NULL; \{\mathbf{X}_i\}_{i=1}^N \leftarrow NULL$
2:
3: **for** *agent* $i = 1, ..., N$ **do**
4: **while** ! *StopCriterion* **do**
5: $\mathbf{z}_{othLast}, \mathbf{X}_{othLast}, \mathbf{X}_{othNext} \leftarrow$ `ReceiveInfo`$(\forall j \in N)$
6: $z_{\mathbf{x}_i} \leftarrow$ `Sense`$(\mathbf{x}_i)$
7: $\mathbf{z}_i \leftarrow [\mathbf{z}_i; \mathbf{z}_{othLast}; z_{\mathbf{x}_i}]$
8: $\mathbf{X}_i \leftarrow [\mathbf{X}_i; \mathbf{X}_{othLast}; {\mathbf{x}_i}^T]$
9: $\boldsymbol{\theta}_{i*} \leftarrow$ `LearnHyp`$(\mathbf{z}_i, \mathbf{X}_i)$
10: $\mathbf{X}_{i*} \leftarrow$ `CalcNeighborsMA`$(\mathcal{X}, \mathbf{X}_{othLast}, \mathbf{X}_{othNext}, \mathbf{x}_i)$
11: $\boldsymbol{\mu}_{i*}, \boldsymbol{\Sigma}_{i*} \leftarrow$ `PredictGP`$(\mathbf{z}_i, \mathbf{X}_i, \mathbf{X}_{i*}, \boldsymbol{\theta}_{i*})$
12: $\mathbf{x}_{next} \leftarrow \underset{\mathbf{x} \in \mathbf{X}_{i*}}{\operatorname{argmax}}(\boldsymbol{\sigma}^2_{i*})$
13: `BroadcastInfo`$(z_{\mathbf{x}_i}, \mathbf{x}_i, \mathbf{x}_{next})$
14: $\mathbf{x}_i \leftarrow$ `MoveTo`$(\mathbf{x}_{next})$

In a first step, we receive the information broadcasted by the other agents (line 5). It consists of the last measurement taken by each of the other agents ($\mathbf{z}_{othLast}$); as well as the positions where those measurements were taken ($\mathbf{X}_{othLast}$), and the next positions where the agents are heading to ($\mathbf{X}_{othNext}$)[3]. This information is sufficient to achieve the inter-agent coordination. On the one hand, the knowledge about the other agents' current position and next positions allows us to implement a collision avoidance mechanism. On the other hand, the set of measurements provided by the other agents act as an indirect mechanism to coordinate the exploration efforts. Since all agents share the same data model, each of them can reproduce what the other agents' uncertainties look like. This leads to an implicit coordination that avoids, for example, that two agents measure at the same position if it is not strictly necessary. It is important to notice that the reception works as a process in parallel, capturing all the broadcasted information during one iteration of the algorithm.

In a second step, the robot incorporates the new measurements to its own vector of measurements (lines 6-8) and computes its next position as explained in the previous section. However, we must consider the possible inter-agent collisions to calculate the set of next positions $\mathbf{X}_{i*}$ (line 10). Here, we add a new constraint to calculate $\mathbf{X}_{i*}$ that forbids agents to be closer than a safety distance d_{safe} to each other. We must guarantee this distance respect to the other agents' positions $\mathbf{X}_{othLast}$, as well as to the positions $\mathbf{X}_{othNext}$ where the other agents are heading to. Now the

[1]In our particular experimental setup, where the agents move in a two-dimensonal space, the sphere B_r corresponds to a circle of radius r.

[2]The measurements' resolution corresponds to the resolution with which we aim to reconstruct the physical process under study.

[3]Vector $\mathbf{z}_{othLast}$ has dimensions $(N-1, 1)$. Matrices $\mathbf{X}_{othLast}$ and $\mathbf{X}_{othNext}$ have dimensions $(N-1, 3)$.

set of possible next positions is reduced to:

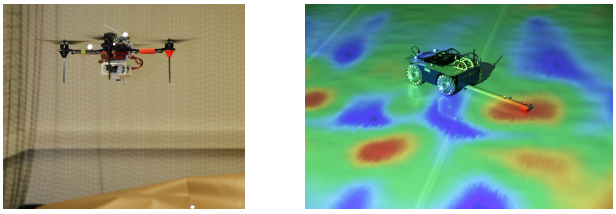
$$\mathbf{X}_{i*} = \left\{ \mathbf{x} \in \mathcal{X} \;\middle|\; \begin{array}{l} \mathbf{x} \in B_r(\mathbf{x}_i), \\ min(||\mathbf{x} - \mathbf{X}_{othLast}||_2) > d_{safe}, \\ min(||\mathbf{x} - \mathbf{X}_{othNext}||_2) > d_{safe}, \end{array} \right\} \quad (6)$$

where $min(||\mathbf{x} - \mathbf{X}||_2)$ is the minimum euclidean distance between $\mathbf{x}$ and any of the rows in vector $\mathbf{X}$.

Once the agent has calculated its next position, it broadcasts its current measurement and its next position to visit. The agents explore the physical process until they reach their stopping criteria. Then each of them is able to reconstruct the physical process as we described it for the single-agent exploration case.

V. EXPERIMENTS AND DISCUSSION OF RESULTS

We validate our algorithm for the exploration of two different physical processes with two different types of robotic platforms (ground-based and flying robots). In both cases, and without loss of generality, we assume a two-dimensional environment $\mathcal{X} \subset \mathbb{R}^2$. For the second experiment, each of the quadcopters move in a 2-dimensional space at a constant height. We employ a commercial motion capture system (Vicon) to provide ground truth information of the robot's position. Our particular setup consists of 16 infrared sensitive cameras and infrared strobes.



(a) Quadcopter. (b) Slider.

Fig. 2: Left: a quadcopter equipped with an ultrasound sensor facing down to measure the range to the floor. Right: a holonomic ground-based robot measuring the magnetic field intensity with a magnetometer.

A. Experiment 1: Magnetic Field Intensity

1) Experimental Setup: We explore the magnetic field intensity on the floor in an indoor environment using a ground-based holonomic robot equipped with a magnetometer (see Figure 2b). We aim to reconstruct this magnetic field intensity with a resolution of 10 cm in an environment that measures 7.5 $m \times 3$ m. The robot is a modified version of the commercially available Slider platform by Commonplace Robotics. The magnetic field sensor module used in the reported experiments is part of a commercial integrated sensor package (Xsens MTx). The complete algorithm runs on a laptop mounted on top of the robot.

We measured the complete magnetic field intensity in the exploration environment with a resolution of 10 cm and we took these measurements as ground truth. This is a valid assumption, considering that the magnetometer is considered as almost noise-free according to its specifications.

2) Experimental Results: In this first experiment, we are interested in testing the proposed exploration strategy for the single-robot case. Therefore, we test the algorithm performance assuming we know *a priori* the data model; i.e. we know the model's hyperparameters *a priori*. Figure 3 shows the evolution of the root-mean-square error (RMSE) as a function of exploration time for three different trajectories: a meander like trajectory, a random trajectory where the target positions are selected according to a uniform distribution from the set of positions in the environment, and finally our algorithm's trajectory. We predict the magnetic field values in the non-measured positions using Gaussian processes regression with the optimal hyperparameters learned from the ground truth data. We observe that the RMSE with our algorithm is close to zero, while we have just measured a 40% of the environment.

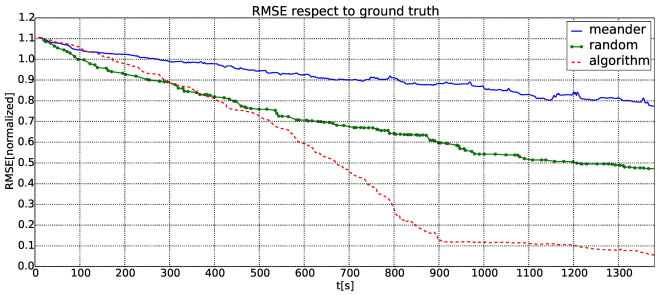


Fig. 3: Root mean squared error of the estimated magnetic field intensity with respect to the ground truth for three different algorithms during a 23 minutes exploration.

B. Experiment 2: Terrain Profile

1) Experimental Setup: We validate our algorithm in a second experiment with two quadcopters to explore a height profile. In this case, the environment measures 8 $m \times 3$ m, and the built height profile measures approx. 60 cm from top to bottom. We intend to explore it with a lateral resolution of 20 cm. Each of the quadcopters flies at a different constant height above the floor of 1 m and 1.5 m to avoid the risk of collisions, although they are not aware of it. They are each equipped with a commercial ultrasound sensor from MaxBotix facing down to measure the range to the floor (see Figure 2a). The sensors each have a nominal range of approx. 7.5 m with opening angles of 45° in the near field and a cylindrical measurement profile for ranges greater than 1.5 m. The profile's height is calculated as the difference between the robot's actual height, which is known and noise-free, and the range between the quadcopter and the height profile. This range is measured with the ultrasound sensor. It is important to remark that the measured terrain's height is independent of the quadcopter's z component.

In order to consider valid measurements for the exploration task, we carry out a pre-filtering step. This is needed due to the characteristics of the ultrasound sensor. Its working principle is based on sending an impulse and waiting for the echo to calculate the distance to the closest object within the sensor's footprint. It could happen that reflections

in the environment and oscillations of the quadcopter's pose could lead to missing measurements (the echo does not return to the ultrasound sensor) or not-plausible measurements (the impulse gets affected by multiple reflections and the resulting measurement is inconsistent considering the environment structure). Therefore, multiple measurements need to be taken to mitigate such effects. We solve this problem by averaging over 10 valid measurements taken at the position of interest; i.e. we discard those measurements that are not plausible, such as negative heights and heights that are above the quadcopter actual z position. In case we obtain no valid measurements during an interval of 30 seconds, we average over the last 10 valid measurements, which are stored in a buffer. This approximation is possible due to the notion of similarity that assumes that closer points in space are more likely to be similar.

The algorithm uses the robot operating system (ROS) [18], and utilizes WiFi for the communication between agents. It is important to notice that, although the algorithm itself is decentralized, only parts of it run on the Raspberry Pi mounted on the quadcopters due to the computational limitations of these devices. Instead, the Gaussian processes regression and learning of hyperparameters run in one separate central computer, but in a decentralized manner. This decentralization is possible because of the nature of ROS, where nodes run in different threads. We employ the pyGPs [19] library to perform these calculations.

For the terrain profile, the ground truth corresponds to 13 markers placed randomly distributed along the height profile. Their positions are recorded using the Vicon tracking system.

2) Experimental Results: We have shown in the previous experiment that our algorithm outperforms the meander and random trajectory. However, we assumed the data model as known. In a realistic scenario, such as a search and rescue mission, we have no prior information about the physical process under study. Instead, we must learn the data model online while performing the exploration. We show in this section results corresponding to the online learning of the model's hyperparameters; as well as the performance of the proposed multi-agent exploration algorithm.

First, we analyze the root-mean-square error (RMSE) between estimate and ground truth (the 13 markers mentioned above) after running the exploration algorithm during the lifetime of the quadcopter's battery, which is approx. 8 minutes in our case. Exchanging the batteries of the quadcopter is a highly time-consuming process. Therefore, our goal is developing exploration algorithms that are able to reconstruct the original process during this battery's lifetime. This is done by both developing intelligent exploration strategies and increasing the number of robots in the swarm.

We compare in Figure 4 the performance of five exploration strategies over time: (i) meander-like trajectory with pre-set hyperparameters that measures all the positions in the environment; (ii) random trajectory with online learning of hyperparameters; (iii) our single-agent exploration algorithm with online learning of hyperparameters; (iv) our multi-agent exploration algorithm with online learning of hyperparameters ; (v) our multi-agent exploration algorithm with pre-set hyperparameters. Both trajectories (i) and (ii) employ Gaussian processes regression to predict the physical process values at positions that were not yet measured. Trajectories (i) and (v) carry out the regression using the optimal hyperparameters learned from the data collected with the meander trajectory in a previous exploration run.

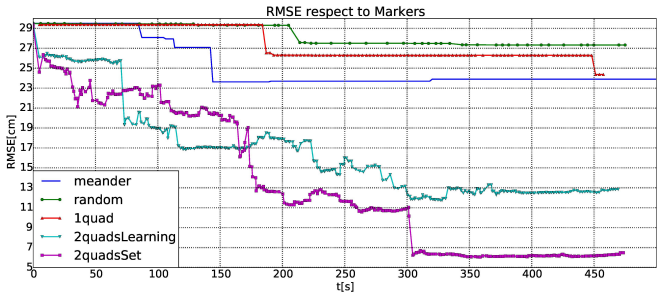


Fig. 4: Terrain profile. Root mean squared error with respect to the ground truth for five different algorithms during a 8 minutes exploration.

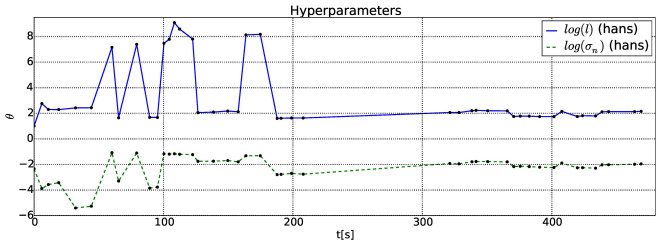
First thing we observe from results is that the minimum error we achieve is approx. 7 cm. This error may seem relatively large. However, it is equal to the best performance we can obtain with the ultrasound sensor in the explored environment. We measured all positions in the environment (we needed 3 batteries and 53 minutes) with the meander trajectory and calculated the RMSE as benchmark. The resulting RMSE was 7.64 cm[4], although we had measured the complete physical process. This error is considerably large and is due to the sensor's footprint and sensor's characteristics, and to the oscillations of the quadcopter while flying. The sensor takes the minimum range – maximum height – within its footprint. Therefore it is not able to distinguish between different heights that are close. This fact induces the error. Since that is the best performance we can get with the sensor without any postprocessing of the measurements, we assume this error as the best possible solution we can obtain.

Next fact we notice is that, in contrast to the results obtained for the magnetic field intensity, the error with the random and single-agent exploration is larger than the one obtained with the meander trajectory. The difference in this case is that we are learning the model's hyperparameters online, while in the other situation they were pre-set to the optimal values. Now, the amount of measurements collected with a single quad in the initial phase of the exploration run is not enough to learn the process model fast, which incurs in an initial loss of performance that is dragged during the rest of the exploration.

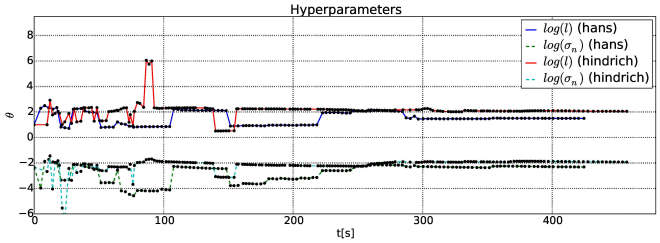
However, attending at results from Figure 4, performance of the random trajectory is comparable to the performance

[4]The fact that the meander's error after measuring the complete process is larger than the one obtained with our algorithm is due to the sensor's noise.

of the single-agent exploration algorithm. Figure 5a demonstrates that this performance of the random trajectory is a mere coincidence, since this trajectory does not guarantee the convergence of the hyperparameters learning (see interval between 210 and 320 seconds). In contrast, as we show in Figure 5b, our algorithm converges fast to the optimal hyperparameters given the available measurements because of the greedy nature of the algorithm.



(a) Hyperparameters learned while following a random trajectory.

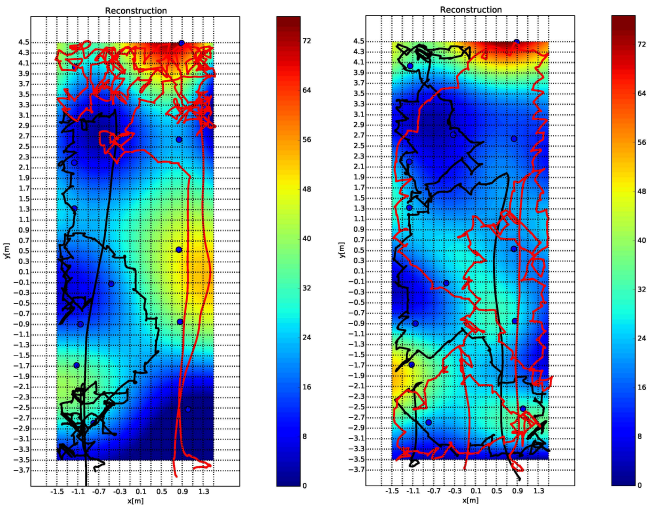


(b) Hyperparameters learned by the two quads, hans and hindrich, while exploring with our algorithm.

Fig. 5: Hyperparameters learned during the exploration run. The black points represent the actual values. The color lines are the result of a linear interpolation of those points.

Since one quad is not sufficient to explore this physical process, we use the multi-agent exploration algorithm with online learning of hyperparameters with two quads (trajectory (iv)). Here, we observe in Figure 4 that the error has been reduced by one half respect to our benchmark, which proves the correct coordination between robots. However, this error of $12.84\ cm$ is still larger than our benchmark error of $7.64\ cm$. On the other hand, the exploration time was 8 minutes, which is approx. seven times smaller than the time needed to achieve our benchmark error.

In order to understand where this remaining error lies, we run the multi-agent exploration algorithm with two quads but with the optimal hyperparameters that where set and prelearned from the data collected with the meander trajectory. This corresponds to trajectory (v). Here, we notice that the error is approx. equal as our best possible solution while the exploration time is approx. seven times smaller. Figure 6 shows the trajectories of the two robots for the trajectories (iv) and (v). We can see that for the online learning case, the robots fly around in a local area in the starting phase. This response is due to the fact that the robots have not learned a proper model and are not able to decide correctly where to measure next. We observe this behavior as well in Figure 5b. In the first 220 seconds, the hyperparameters learning does not converge and this causes an inferior performance compared to the set hyperparameters trajectory. However, we remark that the convergence time is fast considering the non-smooth nature of the measured physical process.



(a) Hyperparameters learning. (b) Hyperparameters set.

Fig. 6: Reconstruction of the terrain profile (see Figure 1) after running our algorithm for two different setups: online learning (left) and defined hyperparameters (right). On top we show the trajectories of the two quadcopters. The big blue dots correspond to the markers' positions that serve us as ground truth.

We can conclude that our multi-agent exploration algorithm is able to achieve the correct coordination between the robots. However, as part of the future work we must improve the learning phase to get closer to the performance obtained while using the optimal hyperparameters.

VI. CONCLUSIONS AND FUTURE WORK

The paper has presented a method for multi-agent exploration of spatially distributed physical phenomena. Gaussian processes are employed as the underlying representation. Then, the system is able to learn, online and in a decentralized fashion, the hyperparameters of the Gaussian process, and, at the same time, determine the best next positions of the agents in the team to obtain new measurements from the point of view of information gain. Agent coordination and collision avoidance is achieved by sharing information. The experiments show how the method allows for a more efficient exploration of the environment compared to other strategies.

Future extensions of the algorithm include considering a more complex environment populated with obstacles. In this sense, we aim to extend our previous work in [20] to propose a multi-agent exploration algorithm in complex, unknown environments. We would like as well to perform exploration missions with robots with complex dynamics. In addition, in order to consider the algorithm for actual search and rescue operations, we should extend it to handle uncertainty in the robot's motion and positioning. As part of the future work, we believe we could reduce the ultrasound sensor error

adding a post-processing step and considering an array of sensors. Additionally, we are interested in exploring alternative covariance functions to represent better the process' model.

One of the assumptions of our proposed work is that the network of agents is fully connected. This assumption could be relaxed by considering a connected network and using appropriate communication protocols. However, to achieve scalability respect to the number of agents, we aim to consider distributed algorithms. That includes solving the Gaussian processes regression and learning of hyperparameters in a distributed fashion; as well as determining the exploration targets in a distributed manner by employing consensus algorithms.

REFERENCES

[1] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1279–1291, 2012.

[2] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "Distributed robotic sensor networks: An information-theoretic approach," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1134–1154, 2012.

[3] R. K. Williams and G. Sukhatme, "Probabilistic spatial mapping and curve tracking in distributed multi-agent systems," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1125–1130.

[4] A. Singh, F. Ramos, H. D. Whyte, and W. J. Kaiser, "Modeling and decision making in spatio-temporal processes for environmental surveillance," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5490–5497.

[5] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *The Journal of Machine Learning Research*, vol. 9, pp. 235–284, 2008.

[6] M. Ghaffari Jadidi, J. Valls Miro, R. Valencia, and J. Andrade-Cetto, "Exploration on continuous gaussian process frontier maps," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6077–6082.

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

[8] J. Fink and V. Kumar, "Online methods for radio signal mapping with mobile robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1940–1945.

[9] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, pp. 707–755, 2009.

[10] J. Chen, K. H. Low, Y. Yao, and P. Jaillet, "Gaussian process decentralized data fusion and active sensing for spatiotemporal traffic modeling and prediction in mobility-on-demand systems."

[11] A. V. Ruiz, M. Angermann, I. Wieser, M. Frassl, and J. Mueller, "Efficient multi-agent exploration with gaussian processes," in *Australasian Conference on Robotics and Automation (ACRA)*, 2014.

[12] R. Ouyang, K. H. Low, J. Chen, and P. Jaillet, "Multi-robot active sensing of non-stationary gaussian process-based environmental phenomena," in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 573–580.

[13] R. Stranders, A. Rogers, and N. Jennings, "A decentralized, online coordination mechanism for monitoring spatial phenomena with mobile sensors," 2008.

[14] N. Cressie, "Statistics for spatial data," *Terra Nova*, vol. 4, no. 5, pp. 613–617, 1992.

[15] A. Kemppainen, J. Haverinen, I. Vallivaara, and J. Roning, "Near-optimal slam exploration in gaussian processes," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*. IEEE, 2010, pp. 7–13.

[16] C. E. Rasmussen and C. K. Williams, "Gaussian processes for machine learning (adaptive computation and machine learning)," 2005.

[17] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[18] *ROS (Robot Operating System)*, 2015 (accessed September 14, 2015). [Online]. Available: http://wiki.ros.org/

[19] *pyGPs - A Package for Gaussian Processes*, 2015 (accessed September 14, 2015). [Online]. Available: http://www-ai.cs.uni-dortmund.de/weblab/static/api_docs/pyGPs/

[20] A. Viseras Ruiz and C. Olariu, "A general algorithm for exploration with gaussian processes in complex, unknown environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.