2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Asynchronous Adaptive Sampling and Reduced-Order Modeling of Dynamic Processes by Robot Teams via Intermittently Connected Networks

Hannes Rovina[1,2], Tahiya Salam[2], Yiannis Kantaros[2], and M. Ani Hsieh[2]

Abstract— **This work presents an asynchronous multi-robot adaptive sampling strategy through the synthesis of an intermittently connected mobile robot communication network. The objective is to enable a team of robots to adaptively sample and model a nonlinear dynamic spatiotemporal process. By employing an intermittently connected communication network, the team is not required to maintain an all-time connected network enabling them to cover larger areas, especially when the team size is small. The approach first determines the next meeting locations for data exchange and as the robots move towards these predetermined locations, they take measurements along the way. The data is then shared with other team members at the designated meeting locations and a reduced-order-model (ROM) of the process is obtained in a distributed fashion. The ROM is used to estimate field values in areas without sensor measurements, which informs the path planning algorithm when determining a new meeting location for the team. The main contribution of this work is an intermittent communication framework for asynchronous adaptive sampling of dynamic spatiotemporal processes. We demonstrate the framework in simulation and compare different reduced-order models under full, all-time and intermittent connectivity.**

I. Introduction

Distributed mobile robot networks have gained more and more interest in recent years due to their ability to cover larger areas and work together on a task. Their mobility and ability to carry sensor equipment make them very well-suited for the tracking and modeling of dynamic processes. Examples of processes of interest include the monitoring of ocean currents for efficient navigation of cargo ships and weather forecasting, wildlife and nature monitoring as in tracking of endangered animal populations or the distribution of plankton in an area of the ocean and modeling of events like oil spills in the ocean or forest fire boundaries. In general, it is difficult to adequately capture the spatiotemporal dependencies of complex, nonlinear dynamic processes using only static sensors since optimal sampling locations can change over time, are unknown *a priori*, and may require large quantities of sensors to ensure proper coverage of the space. Mobile robots can mitigate this by adaptively changing their sampling locations which significantly reduces the number of required units for any given workspace.

This work was supported by ARL DCIST CRA W911NF-17-2-0181 and National Science Foundation (NSF) grant IIS-1812319.

[1]H. Rovina is with the DISAL Laboratory, Swiss Federal Institute of Technology, Lausanne, VD 1015 CH. [2]H. Rovina, T. Salam, Y. Kantaros, and M. A. Hsieh are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104 USA hannes.rovina@epfl.ch; {tsalam,kantaros,m.hsieh}@seas.upenn.edu

Digital Object Identifier (DOI): see top of this page.

Network connectivity is critical for any multi-robot and/or robot swarm task objective, but is often unreliable and difficult to maintain in practice. Works on preserving connectivity at all-times in a centralized [1]–[3] and distributed [4]–[6] fashion mostly focus on maintaining the Fiedler value of the underlying communication network graph above some acceptable value. Maintaining all-time network connectivity often overly constrains robot motions. To overcome this issue, intermittent communication frameworks have been developed. Intermittent communication can be achieved by having a periodic reconnection of the complete network within a given time horizon as in [7]. The need to reconnect the complete network is overcome in [8]–[12], where the network is guaranteed to connect over time, infinitely often. The main idea is to connect teams of robots at a common location in space and allow them to pursue their task in between the communication events, thus removing the constraint of maintaining the network connectivity at all times. By carefully assigning robots to multiple subgroups and/or selecting the periodicity and location of communication events, the information is guaranteed to propagate across the entire network in a finite time.

A perfect example to demonstrate the need of intermittently connected robot networks consists of the sampling and tracking of spatiotemporal processes. Many of these spatiotemporal processes often occur at large spatial scales, in uncertain dynamic environments that are difficult to model, and driven by a complex interplay of physical, chemical, and/or biological processes. This means that robots tasked to monitor these processes must not only be robust to failures, but also be able to operate with potentially significant communication constraints. Furthermore, in situations where only sparse measurement data is available, robots must have the ability to operate in a distributed fashion, often with intermittent network connectivity, while simultaneously extracting as much information from the obtained data in order to successfully estimate and track the process of interest.

Modeling complex, nonlinear dynamic processes is generally a challenging task, but there are existing techniques to accomplish this in an online fashion. One such method are Gaussian Processes (GPs) [13] which use a model free approach to optimize hyperparameters to fit a kernel function to the underlying process. While generally applied to stationary processes [14]–[16] for spatial field estimation, GPs can be extended to include temporal components of the processes as in [17]. Another method is the proper-orthogonal decomposition (POD) [18]. POD has been extensively used to determine the placement of static sensors for modeling and estimation of various spatiotemporal processes [19]–[22]. In [23], a distributed POD was presented to determine the best sensing locations for a team of robots to adaptively track and sample a dynamic spatiotemporal process assuming a fully connected mobile robot communication network. What all these methods have in common, is that they disregard the communication constraints on the mobile robot network. Assuming full connectivity or maintaining an all-time connected network is unfeasible in large spatial scales with a reduced number of mobile robots but can be overcome with the introduction of an intermittently connected network.

In this work, we take inspiration from [23], [24] and present an asynchronous adaptive sampling and reduced-order modeling framework for multi-robot teams via intermittently connected mobile communication networks. Taking inspiration from [24], we employ the use of communication events to create an intermittently connected network. Different from [24], the communication events in this work are not constrained to a common location in space but to a connected subnetwork for every team somewhere in continuous space. Towards this end, we consider the development of a reduced-order model (ROM) of the process of interest when robots must rely on measurements obtained online and shared across an intermittently connected communication network. Similar to [23], we assume our team size is small and each robot's sensing radius is finite and thus the team is unable to achieve full coverage of the workspace. Thus, a ROM of the spatiotemporal process is computed using sparse measurements obtained by the team and the ROM is employed to estimate field values in regions not covered by the team. Different from [23], we explicitly address the planning of trajectories that will enable the robots to move to the most informative sampling locations.

The main contribution of this work consists of an intermittent communication framework for distributed multi-robot teams adaptively sampling and modeling nonlinear dynamic spatiotemporal processes. To the best of our knowledge, this is the first framework that explicitly addresses the estimation of dynamic spatiotemporal processes via asynchronous updates within an intermittently connected communication network. Our solution allows the efficient modeling of a dynamic process while guaranteeing communication between robots with constrained communication abilities. The ROM enables the robots to infer measurements of the dynamic process at unseen locations and to plan their paths to reduce the model uncertainty of the dynamic process of interest. The algorithm can be scaled to larger networks of robot teams due to its inherent distributed nature and intermittent communication. Finally, we show in simulation studies for different scenarios how the algorithm compares to fully and all-time connected robot frameworks and outperforms them.

The rest of this work is organized as follows, in II the problem is formalized, in III the intermittent connectivity framework is presented, in IV the simulation setup is described, the results of the simulations are shown in V and conclusive remarks in VI summarize the work.

II. Problem Formulation

Consider modeling a dynamic process P in a continuous spatial region $\Omega \subset \mathbb{R}^2$. We can discretize Ω into n spatial points at which measurements of the dynamic process P can be obtained. Let there be R mobile robots, where $R \ll n$, which live in Ω, that can collect measurements of the dynamic process according to

$$\mathbf{y}(t, \mathbf{x}) = P(t, \mathbf{x}) + \mathbf{z}, \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^m$ are the measurements of a single robot at discrete time steps $t = 1, ..., m$ at locations $\mathbf{x} \in \Omega$ and $\mathbf{z} \sim \mathcal{N}(0, w)$ is the measurement noise with variance w.

Furthermore, assume robots have a limited communication range $c_r \ll \Omega$ so that they can only communicate with other robots in close physical proximity. This introduces a strong constraint on the mobility of robots if they are to maintain a communication network at all time. Therefore, we propose an intermittent communication which requires only a subset of the robots to form a connected subnetwork at a spatial location. Robots are assigned to $M \geq 1$ teams T_i for $i = 1, ..., M$ which define the subset of the subnetwork and each robot is part of exactly two teams. The teams are designed so that there exists a sequence of teams, where consecutive teams have non-empty intersections, connecting every two teams of robots. This ensures that information can propagate in the network.

Consider a team of robots T_i, we define a graph $\mathcal{G}_{T_i}(\mathcal{V}_{T_i}, \mathcal{E}_{T_i})$ with nodes $\mathcal{V}_{T_i}$ as the locations of the robots and edges $\mathcal{E}_{T_i}$ as the communication links between robots in T_i, as the communication graph of team T_i. Whenever $\mathcal{G}_{T_i}$ forms a connected network, we have a communication event in team T_i. To ensure communication events over time for all teams, we require that $\mathcal{G}_{T_i}$ is connected infinitely often over time for all teams. This requires that paths of the robots in each team are planned in order to form a connected graph while sampling at informative locations in Ω along the paths to construct a model of P.

Problem Statement: Given a region Ω discretized into n spatial points, R homogeneous robots with limited communication range c_r divided into M teams T_i such that $R \ll n$, develop a path planning strategy to adaptively sample and model the process P such that the communication graphs $\mathcal{G}_{T_i}$ of all teams are intermittently connected infinitely often.

For the sake of simplicity, we assume robots are point masses with perfect state estimation and reliable communication. Nevertheless, these assumptions can be relaxed and will be a direction for future inquiry.

III. Methodology

To enable robots to better model and track the dynamic spatiotemporal process of interest, robots meet asynchronously in a given order to exchange information which creates an intermittently connected robot communication network. In this work, we take inspiration from [8], [12], [24], [25] to determine how best to generate a schedule to ensure infinite repetitions of meeting events for the distributed robot teams. We pair this with an RRT* based path planning strategy to identify the meeting locations for robots that are scheduled to meet.

The integrated control and coordination framework for sampling and modeling of a dynamic process under intermittent communication is described in Alg. 1. It requires the initial paths and a periodic communication schedule that determines when each team should communicate. The robots then follow their respective paths and take measurements along the way. At the final waypoint, all the robots from the same team meet and exchange their measurements. Now, the model can be updated, which only needs to be done by one member of the team since they all have the same data available. Once the model is updated and shared with the team members, paths for the next meeting are calculated. This process then repeats infinitely often.

Algorithm 1 Coordination Framework

```
1: **Input** Initial paths $\mathbf{p}_{ij}^{init}$ for robots and schedule($k$)
2: **for** $k = 1 : \infty$ **do**
3:     Follow paths $\mathbf{p}_{ij}^{k}$ and collect measurements every $\Delta t$
4:     Exchange information at meeting location with team
       members $r_{ij} \in T_i$
5:     Update model parameters and estimate field
6:     Create paths $\mathbf{p}_{ij}^{k+s}$ for next meeting event for team
       $T_i$ (Alg. 2)
7: **end for**
```

In order to avoid a memory overflow, only data from a newer timestep is kept in memory if two robots took measurements at the same location. This has a drawback on the modeling side, but ensures a small memory footprint. Furthermore, even though the path planner schedules meeting locations with simultaneous arrival times, it is not a hard requirement. The team members will wait until all members are at the meeting location before continuing with the algorithm. This ensures that there is no deadlock in case robots arrive late due to disturbances.

A schematic overview of the coordination framework is shown in Fig. 1 and the following sections will explain the different blocks of the algorithm.

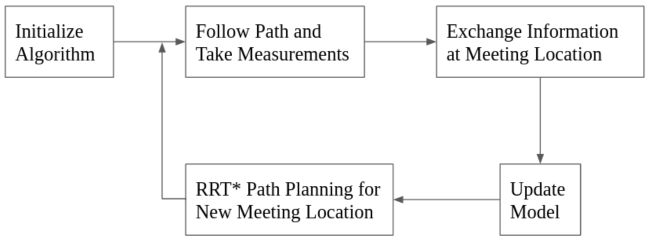

Initialize Algorithm
Follow Path and Take Measurements
Exchange Information at Meeting Location
Update Model
RRT* Path Planning for New Meeting Location

Fig. 1: Schematic representation of overall framework.

A. Scheduling Communication Events

To ensure communication events between multiple robot teams, a schedule needs to be created. The schedule does not determine the physical time of meetings, but the order in which the meetings should happen, thus creating asynchronous communication events for the teams. Hereafter, the discrete time instants when a meeting happens are called epochs and are denoted by k. The constructed schedules are periodic in the sense that communication within team T_i happen every s epochs. Assume 4 robots distributed in 4 teams as depicted in Fig. 2, where each robot is a member of two teams as illustrated by the different colors on the robot body. In order to have a conflict free schedule, robots cannot have a meeting with two different teams during the same epoch. In the illustrated case, the schedule results in 2 epochs with a maximum period $s = 2$ between meetings with the same team. A more in-depth explanation of the construction of such schedules can be found in [8]. The resulting schedule for the illustrated example is

$$\begin{bmatrix} sched_{yg} \\ sched_{rg} \\ sched_{rb} \\ sched_{yb} \end{bmatrix} = \begin{bmatrix} 1 & 4 \\ 1 & 3 \\ 2 & 3 \\ 2 & 4 \end{bmatrix}^{\infty}, \tag{2}$$

where the subscripts for the schedules are the colors of the corresponding boat and the ∞-symbol shows the infinite repetition of the schedule. The right matrix shows the team correspondence and which team has a meeting at a given epoch. In our illustrative example, at epoch $k = 1$, teams T_1, T_2 have a meeting and at epoch $k = 2$, teams T_3, T_4 have a meeting. In a scenario with a different configuration of robots and teams, it is possible that in order to create a conflict free schedule, some robots will have no meeting at a given epoch. This does not mean that they will wait during that epoch but that they are free to roam around and take measurements during that time.

Given the schedule, the next task is to plan the paths to enable the robots to arrive at their scheduled meeting.

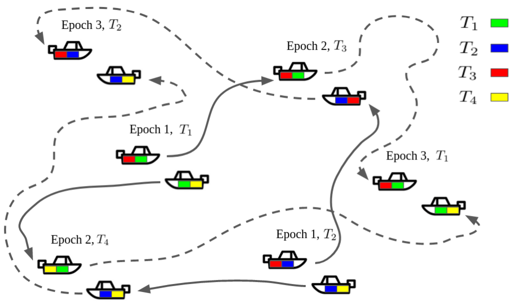


Fig. 2: Schematic of the intermittent communication between 4 different teams (T_1 to T_4). Connected line corresponds to the first executed paths and the interrupted line to the second. After just 2 epochs the initial robot teams meet again and have gathered all the information from the other teams.

B. Sampling-Based RRT* Path Planning

Sampling-based path planning can be done with RRT* [26]. The algorithm constructs a graph $\mathcal{G}_{T_i}^{k+s}(\mathcal{V}, \mathcal{E})$ for team T_i at epoch $k+s$ in the convex workspace Ω, which we can use to create a goal set $\mathcal{X}_g^i$ for every team i with robots r_{ij} with properties:

$$\mathcal{X}_g^i = \{v \in \Omega | \text{ (i) } \lambda_2(\mathcal{L}(\mathcal{G}_{T_i}^{k+s}(v))) > 0 \\ \text{(ii) } \| \min_{r_{ij} \in T_i} t_{ij}(v) - \max_{r_{ij} \in T_i} t_{ij}(v) \| < \Delta t \\ \text{(iii) } \mathrm{unc}(v) < \delta \}, \quad (3)$$

where (i) defines a connected graph if the second smallest eigenvalue of the Laplacian is positive [27], (ii) ensures that the robots arrive at the meeting location in a predefined time interval Δt and (iii) ensures a reduction of the uncertainty in the model so that the robots explore before meeting again.

We propose an adapted version of the RRT* algorithm in Alg. 2 similar to [24]. It is similar to the original version but in line 7 instead of just using a random sampling location v_{rand}, we look for a random sampling location which gives the highest information gain along the trajectory towards it, for a given number of tries n_{tries}. The output of the algorithm is the path for each robot which is given by its tree that is rooted in the last meeting locations of the teams in epoch $k+s-1$ and is connected to the trees of the team members in the new meeting location at epoch $k+s$.

In the following, the algorithms for steer, extend and rewire will be explained and their respective differences highlighted.

Algorithm 2 Path Planning

```
1:  **Input** robots $r_{ij}$ in team $T_i$
2:  Initialize graph $\mathcal{V} = \{v_0\}, \mathcal{E} = \emptyset, \mathcal{X}_g = \emptyset$
3:  for $s = 1 : n_{samples}$ do
4:      for $c = 1 : n_{tries}$ do
5:          Sample $v_{rand,c}$ from $\Omega$
6:          Find nearest node $v_{nearest} \in \mathcal{V}$ to $v_{rand,c}$
7:          Calculate information gain along path
            $v_{nearest} \to v_{rand,c}$
8:          Keep $v_{rand} \leftarrow v_{rand,c}$ with highest gain
9:      end for
10:     Steer towards $v_{rand}$ to find $v_{new}$ (Alg. 3)
11:     Update vertices $\mathcal{V} = \mathcal{V} \cup v_{new}$
12:     Build set $\mathcal{V}_{near} = \{v \in \mathcal{V} | \|v - v_{new}\| \leq \epsilon\}$
13:     Extend tree towards $v_{new}$ (Alg. 4)
14:     Rewire tree (Alg. 5)
15: end for
16: Find $v_{end} \in \mathcal{X}_g^i$ with smallest travel time
17: return $p_{ij}^{k+s} = (v_0, ..., v_{end})$
```

The algorithm for the steer function is shown in Alg. 3. It takes the nearest neighbor node to the sampled node and tries to extend the tree towards the sampled node while satisfying the robot dynamics. In this case, the robots have single integrator dynamics with bounded maximum velocities u_{max}. Furthermore, the robots are limited to a maximum travel distance ϵ for the extension of the tree. In lines 5-9 the expected delay at the nearest node for all robots of the same team is calculated. The expected delay reduces the allowed distance to v_{new} to synchronize arrival times at the meeting location. The output of the algorithm is a new position v_{new} which is then added to the graph.

Algorithm 3 Steer

```
1:  Input robots $r_{ij}$ in team $T_i$, max travel distance $\epsilon$
2:  Compute $d_{ij} = v_{rand} - v_{nearest}$
3:  Compute $d_{min} = \min_{r_{ij}}(\epsilon, \|d_{ij}\|)$
4:  Compute travel time $\Delta t_{d_{min}} = d_{min}/u_{max}$
5:  Compute expected delay $\Delta t_{c,ij}$ for each $r_{ij}$
    $\Delta t_{c,ij} = \Delta t_{d_{min}} - (t_{ij}(v_{nearest}) - \min_{r_{ij}} t_{ij}(v_{nearest}))$
6:  if $\Delta t_{c,ij} > 0$ then
7:      $v_{new,ij} = v_{nearest,ij} + u_{max}\Delta t_{c,ij} d_{ij}/\|d_{ij}\|$
8:  else
9:      $v_{new,ij} = v_{nearest,ij}$                ▷ $r_{ij}$ does not move
10: end if
```

In Alg. 4 the parent node to the new node is selected, out of a set of close nodes, based on some cost function. In this case, the cost function measures the utility of the node and is a trade-off between travel time and information gain. The output is the extension of the graph by adding an edge between the best parent node to the new node. Furthermore, the new node is examined on the goal set conditions and if applicable, added to the goal set.

Algorithm 4 Extend

```
1:  Input robots $r_{ij}$ in team $T_i$
2:  Set $v_{min} = v_{nearest}$ and $utility_{min} = \mathrm{Utility}(v_{new})$
3:  for $v_{near} \in \mathcal{V}_{near}$ do
4:      Compute $\mathrm{Utility}(v_{new})$ with $v_{near}$ as parent
5:      if $\mathrm{Utility}(v_{new}) > utility_{min}$ then
6:          Set $v_{min} = v_{near}$, $utility_{min} = \mathrm{Utility}(v_{new})$
7:      end if
8:  end for
9:  Update graph with edges $\mathcal{E} = \mathcal{E} \cup \{(v_{min}, v_{new})\}$
10: $\mathrm{Utility}(v_{new}) \leftarrow utility_{min}$
11: if $v_{new} \in \mathcal{X}_g^i$ then
12:     Update $\mathcal{X}_g^i = \mathcal{X}_g^i \cup \{v_{new}\}$
13: end if
```

During the rewiring process, as shown in Alg. 5, we look at the case when the new node is the parent to close nodes. If the utility increases by selecting the new node as a parent, we update the edges by removing the old parent node and adding the new node as the parent. This can have implications on the goal set, which thus needs to be updated again.

The proposed sampling-based algorithm is probabilistically complete since RRT* is probabilistically complete and the steer function satisfies the condition that v_{new} is closer to v_{rand} than $v_{nearest}$. Unlike in [24], we drop the equal time constraint in the rewiring process and allow the robots to arrive in a small time interval at the meeting location, which makes the algorithm asymptotically optimal again.

C. Reduced-Order Modeling

Online modeling of dynamic spatiotemporal processes is no straightforward task and while several methods exist, it

Algorithm 5 Rewire

```
1: Input robots $r_{ij}$ in team $T_i$
2: for $v_{near} \in \mathcal{V}_{near}$ do
3:     Compute $utility_{new} \leftarrow$ Utility($v_{near}$) with $v_{new}$ as parent
4:     if $utility_{new} >$ Utility($v_{near}$) then
5:         Update edges $\mathcal{E} = \mathcal{E} \setminus \{(v_{parent}, v_{near})\} \cup \{(v_{new}, v_{near})\}$
6:     end if
7:     Update goal set $\mathcal{X}_g^i$
8: end for
```

is difficult to choose the appropriate one for a given task. For this reason, this work focuses on two different methods: Proper-Orthogonal Decomposition [18] and Gaussian Process Regression [13]. In the following sections we briefly summarize the two methods and describe how each method can be used to model spatiotemporal processes, how new measurements can be incorporated into the models, and how the models can be used to estimate missing field values.

1) Proper-Orthogonal Decomposition

Modeling fluid dynamics consists of capturing the complex temporal and spatial behaviors of the process. Proper-Orthogonal Decomposition [18], also known as Principal Component Analysis, Empirical Orthogonal Decomposition, and/or Karhunen–Loeve Decomposition, extracts the dominant dynamics of the process by transforming the set of spatiotemporal observations into a set of linearly uncorrelated principal modes that enables the construction of a low-dimensional approximation of the process itself. We briefly summarize the online POD methodology for completeness and refer the interested reader to [23] for details.

In general, POD is calculated using all available snapshots of data. Since we are interested in the online estimation of the process, we assume data is only available from the measurements obtained up to the current time. Thus, for the time instances $t = 1, ..., m$, we denote the set of measurements obtained as $\mathbf{y}(\mathbf{t}) = [x_1(t_1), x_2(t_2), ..., x_n(t_m)]^T$, where t_m is the current time step and x_n is the n-th location of a measurement in Ω. Let

$$\mathbf{K} = \frac{1}{n}\sum_{t=1}^{t=t_c} \mathbf{y}(\mathbf{t})\mathbf{y}(\mathbf{t})^T = \frac{1}{n}\mathbf{Y}\mathbf{Y}^T, \tag{4}$$

denote the covariance matrix where $\mathbf{Y} \in \mathbb{R}^{n\times m}$. By solving the symmetric eigenvalue problem $\mathbf{K}\phi_i = \lambda_i\phi_i$ and sorting the n eigenvalues and orthonormal eigenvectors of $\mathbf{K}$ such that $\lambda_1 > \lambda_2 > ... > \lambda_n$, we obtain a new basis $\boldsymbol{\phi}$.

The basis ϕ can now be truncated into a low-dimensional basis $\mathbf{\Phi}$ by choosing k eigenvectors so that their corresponding eigenvalues satisfy

$$\frac{\sum_{i=1}^{i=k}\lambda_i}{\sum_{i=1}^{i=n}\lambda_i} \geq E, \tag{5}$$

where E is a user-defined fraction of the total variance of the system. The truncated basis provides a reduced-order approximation of the dynamic spatiotemporal process which can be used to infer the measurements of the dynamic process in regions of the workspace that have not been visited and/or not covered by a sensor as follows

$$\hat{\mathbf{y}}(\mathbf{t}) = \mathbf{\Phi}\mathbf{c}(\mathbf{t}), \tag{6}$$

where $\mathbf{c}(\mathbf{t})$ are time dependent coefficients.

Since the data points correspond to sparse measurements and do not cover the complete field, we employ the gappy POD [18], [19], [21] which is a variant of the standard POD developed to be robust to missing or sparse measurements. Gappy POD allows us to calculate an estimate of the time dependent coefficients $\mathbf{c}(\mathbf{t})$ from the reduced basis $\mathbf{\Phi}$ and the sparse measurements $\mathbf{y}(\mathbf{t})$ with

$$\begin{aligned}&\hat{\mathbf{c}}(\mathbf{t}) = \mathbf{A}^{-1}\mathbf{B}\\ &\text{for } \mathbf{A} = \mathbf{\Phi}^T\mathbf{\Phi} \text{ and } \mathbf{B} = \mathbf{\Phi}^T\mathbf{y}(\mathbf{t}),\end{aligned} \tag{7}$$

where this estimated coefficient can then be used in (6) to infer the missing measurements in the field.

This process is repeated whenever we want to incorporate new measurements into our model.

2) Gaussian Process Regression

Another widely used method for modeling of spatial processes are Gaussian Processes [13]. GPs are a non-parametric Bayesian technique and are characterized by a mean and covariance function (kernel) and a set of hyperparameters. In order to incorporate the spatial and temporal characteristic of the dynamic process, an appropriate kernel has to be chosen.

The most used kernel for spatial processes is the squared exponential (SE) kernel, also known as radial basis functions. Several spatiotemporal kernels have been investigated in [17], but higher complexity in the kernels introduce more computational costs. By multiplying a spatial SE kernel with a temporal SE kernel,

$$\begin{aligned}SE(\mathbf{x}, \mathbf{x}'|\theta) &= \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{x}')^T M(\mathbf{x}-\mathbf{x}')\right)\\ &\quad + \delta(x, x')\sigma_n\end{aligned} \tag{8}$$

$$K(\mathbf{x}, \mathbf{x}', \mathbf{t}, \mathbf{t}'|\theta) = SE(\mathbf{x}, \mathbf{x}'|\theta) \times SE(\mathbf{t}, \mathbf{t}'|\theta), \tag{9}$$

where $M = diag(\mathbf{l})^{-2}$ and $\theta = [\sigma_f^2, \mathbf{l}, \sigma_n]$, a time dimension can be added to the normally 2-dimensional SE kernel. The hyperparameters θ are the model variance σ_f, lengthscale l for each dimension, and the noise variance σ_n.

GPs allow us to estimate the joint distribution of measurements at observed locations $(\mathbf{x}, \mathbf{y}(\mathbf{t}))$ and infer estimated measurements at unobserved locations $(\mathbf{x}_*, \mathbf{y}_*(\mathbf{t}))$ as

$$\begin{bmatrix}\mathbf{y}(\mathbf{t})\\ \mathbf{y}_*(\mathbf{t})\end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix}K+\sigma_n^2 I & K_*^T\\ K_* & K_{**}\end{bmatrix}\right), \tag{10}$$

where $K = K(\mathbf{x}, \mathbf{x})$, $K_* = K(\mathbf{x}_*, \mathbf{x})$ and $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$. Thus we can estimate the Gaussian posterior mean and covariance with $K_y = K + \sigma_n^2 I$ as

$$\begin{aligned}\hat{\mathbf{y}}_*(\mathbf{t}) &= K_* K_y^{-1}\mathbf{y}(\mathbf{t})\\ \hat{\mathbf{\Sigma}}_*(\mathbf{t}) &= K_{**} - K_* K_y^{-1} K_*^T.\end{aligned} \tag{11}$$

We can train the model by optimizing the kernel's hyperparameters with the available data. This can be done by maximizing the log-marginal likelihood

$$\log p(\mathbf{y}|\mathbf{x}) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi \quad (12)$$

with respect to the hyperparameters θ.

As in the case of the POD, this process is repeated whenever new measurements are included into the model.

IV. Simulations

In this section, we describe our simulation setup, scenarios and the evaluation metrics used to evaluate the proposed coordination framework. The implementation of the GP modeling was done using the Python library *GPy* [28]. An accompanying video of a simulation is available online.

A. Setup

In order to analyze the performance of the coordination framework, a simulation of a 600×600 2-dimensional pixel grid space was used. The dynamic process overlaid on the grid space was obtained from video data of an experimental flow tank at low Reynolds number. The conducted fluid experiments consisted of glycerol in a $10\times10cm$ tank with a depth of 1 - $2cm$. Submerged in the tank was a 4×4 array of equally spaced disks, where two sets of 8 disks could be separately controlled. The resulting flow is spatially non-uniform and time-varying. The dye was strategically placed to ensure it stretched along boundaries of dynamically distinct regions over time. A grayscale video of the experiment was used to estimate the concentration values of the dye at each time step. Fig. 3 shows the dye concentration over time and is akin to plankton assemblages in the ocean.

The selected configuration for the robot teams was 4 robots in 4 different teams resulting in a schedule with 2 epochs as already described in the example of III-A. The starting locations for the robots were in the 4 corners of the simulated space for the distributed, intermittently connected scenario and in the center for the all-time connected scenario. Gaussian noise $z \sim \mathcal{N}(0, 0.2)$ was added to the measurements of the robots and the simulation duration was $100s$. The sensing and communication radius s_r and c_r depended on the applied scenario and were $s_r = 1, c_r = 3$ for the GP modeling and $s_r = 20, c_r = 3$ for the POD in the case of intermittent connectivity, while for all-time connectivity the communication radius was $c_r = 20$.

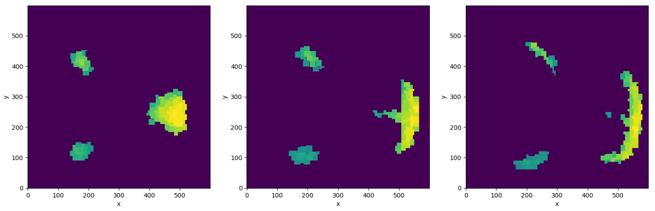


Fig. 3: Different time instances of the dye solution depicting the dynamic process to be modeled. Time evolves from left to right in images. Color indicates dye concentration.

1) Scenarios

Simulations were carried out for 12 different scenarios with 10 random runs for each scenario. Abbreviations for the cases are: stationary process SP, dynamic process DP, all-time connected AC, fully connected FC, intermittently connected IC, Gaussian Process GP and proper-orthogonal decomposition POD. The scenarios are the following:

1) SP, AC, GP
2) SP, AC, POD
3) SP, FC, GP
4) SP, FC, POD
5) SP, IC, GP
6) SP, IC, POD
7) DP, AC, GP
8) DP, AC, POD
9) DP, FC, GP
10) DP, FC, POD
11) DP, IC, GP
12) DP, IC, POD

For the all-time connected framework, the robots were geometrically constraint to stay in a fixed configuration from each other in order to guarantee an all-time connected network. The path planning algorithm was then applied to their respective geometric mean position.

B. Error Metric

Capturing the error of the model is not a straightforward task since we are not only interested in a pixel to pixel comparison, but also if the model is able to capture all the important dynamics of the complex spatiotemporal process. For this reason, two error metrics have been employed.

1) Root-Mean-Square Error

A classic metric is the Root-Mean-Square Error (RMSE). It looks at a pixel to pixel comparison and is calculated as

$$RMSE = \sqrt{\mathbf{E}((\hat{X} - X)^2)}, \quad (13)$$

where $\hat{X}$ is the estimated field and X is the ground truth at the corresponding time instance.

2) Procrustes Analysis

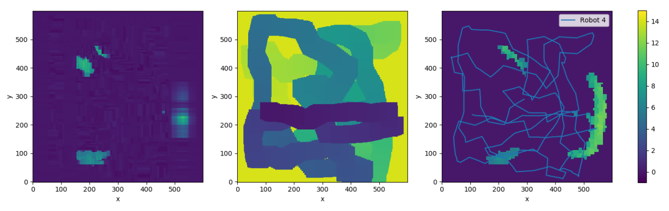
The goal in Procrustes analysis [29], [30] is to find an optimal scale factor f, translation and rotation matrix Θ, which then allow to superimpose the model estimate onto the ground truth and compare the dissimilarity index (DISSIM), defined as

$$R = \underset{\Theta}{\text{argmin}}\|\Theta\hat{X} - X\|_F \text{ subject to } \Theta^T\Theta = I$$
$$DISSIM = \sum(X - \hat{X}\Theta^T f)^2, \quad (14)$$

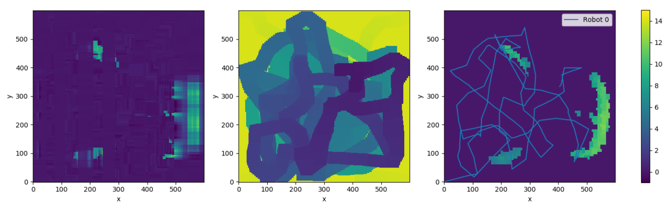
where f is the sum of the singular values of $X^T\hat{X}$. The dissimilarity index goes from 0 to 1, where 0 means identical. Procrustes analysis thus gives a measure of the captured features of the dynamic process.

V. Results

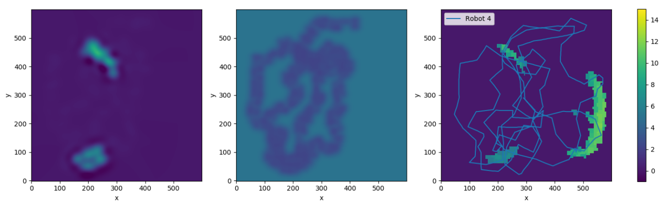
The simulation results of 4 robots in 4 teams under the scenarios described in IV-A.1 can be seen in Fig. 4 for the spatiotemporal cases. From a visual perspective one can clearly see that the distributed, intermittently connected approach results in a more accurate modeling of the process



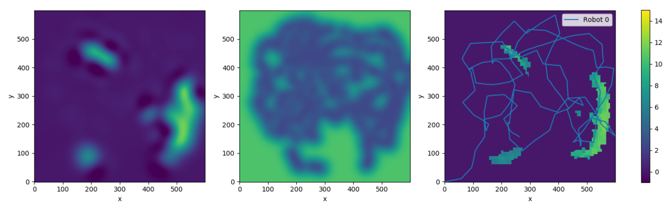
(a) All-time connected with POD.



(b) Intermittently connected with POD.

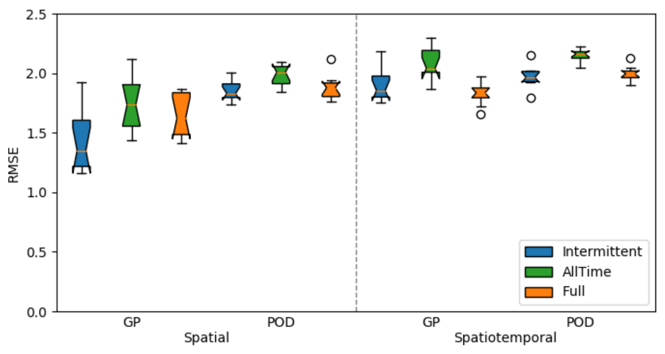


(c) All-time connected with GP.

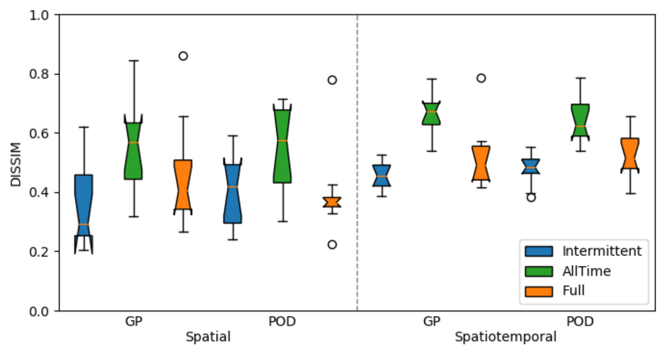


(d) Intermittently connected with GP.

Fig. 4: Modeling accuracy of the different reduced-order models for the dynamic process with all-time 4a, 4c and intermittently 4b, 4d connected robots. Depicted from left to right: estimated field based on model, trajectory/estimate uncertainty (coverage), ground truth with robot trajectory overlaid.



(a) RMSE for all the different scenarios.



(b) Dissimilarity index for all the different scenarios.

Fig. 5: Comparative results of the RMSE 5a and the dissimilarity index 5b for all the different scenarios with 10 trials each. Algorithm is compared between POD and GP modeling as well as intermittent, full and all-time connected scenarios. The dissimilarity index goes from 0 to 1 where 0 means identical.

compared to the scenario where the robots are constraint to be all-time connected. It is also clear why we need two error metrics, since there are only a few features to be captured by the model, and in case of failure to do so, the RMSE error might not be able to elucidate this. Nevertheless, Fig. 5 shows a clear increase in performance for the intermittent communication framework for both the RMSE and dissimilarity index in all cases.

TABLE I: Mean error and standard deviation for all scenarios. Orange columns correspond to GP and yellow to POD.

Error Results	**Simulation Scenarios**											
	1	**2**	**3**	**4**	***5***	***6***	***7***	***8***	**9**	**10**	***11***	***12***
μ_r^{a}	1.95	1.98	1.94	1.88	1.44	1.85	2.08	2.15	2.13	2.00	1.89	1.97
σ_r	0.48	0.14	0.35	0.15	0.28	0.21	0.19	0.12	0.39	0.19	0.34	0.21
μ_d^{b}	0.55	0.55	0.45	0.39	0.36	0.40	0.67	0.64	0.51	0.53	0.45	0.48
σ_d	0.22	0.24	0.21	0.20	0.17	0.25	0.17	0.15	0.14	0.17	0.12	0.15

[a]Subscript $_r$ stands for RMSE, [b]Subscript $_d$ stands for DISSIM.

It is interesting to point out that the GP models only get a single point measurement at each sensing period, while the POD requires a larger sensing range to achieve similar performance. This influences the performance in the case of all-time connectivity, meaning that the POD is using a lot more points to estimate the process and therefore has a slightly better performance in the spatiotemporal case as seen in the dissimilarity index. In the case of a stationary process however, both modeling approaches have almost identical modeling performance for the all-time connected framework. This result, however, does not transfer when intermittent connectivity is considered. There is a small increase in performance for the GP compared to the POD modeling approach in both the stationary and dynamic process. GPs and PODs both allow to make forecasts for future time steps of the dynamic process, but in case of the GP it comes at a higher computational cost, which should be taken into account for future implementations on real robots.

Table I summarizes the results. The most interesting scenarios are 6-12 which look at the dynamic process.

VI. Conclusion

In this work, we investigated the problem of adaptively sampling and modeling a dynamic process with distributed, intermittently connected robots. The communication constraints were overcome with the introduction of meeting locations and path planning. Adaptive sampling was achieved by developing a ROM of the process and using the ROM to determine the next best sampling locations for the robots in a team. To the best of our knowledge, this is the first framework for distributed adaptive sampling and reduced-order modeling using an intermittently connected mobile robot communication network which could be extended to large-scale robot networks. Simulations demonstrated the team's ability to distributively model the process under intermittent communication for different scenarios and clearly showed increased performance when compared to robot teams maintaining full or all-time communication network connectivity.

Future work will investigate different kernels for the Gaussian Processes, look into the exploitation versus exploration dilemma during path planning as well as introduce heterogeneity in sensing and motion capabilities in the robot teams. Real world experiments should be performed to validate the framework and better investigate the computational complexity of the proposed approach.

Acknowledgment

The authors would like to thank Professor Philip A. Yecko from The Cooper Union for providing the video of the flow tank for low Reynolds numbers.

References

[1] J. Derenick, J. Spletzer, and A. Hsieh, "An optimal approach to collaborative target tracking with performance guarantees," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 1-2, pp. 47–67, 2009.

[2] Y. Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph laplacian," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 99–103.

[3] M. M. Zavlanos and G. J. Pappas, "Potential fields for maintaining connectivity of mobile networks," *IEEE Transactions on robotics*, vol. 23, no. 4, pp. 812–816, 2007.

[4] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized estimation of laplacian eigenvalues in multi-agent systems," *Automatica*, vol. 49, no. 4, pp. 1031–1036, 2013.

[5] L. Sabattini, N. Chopra, and C. Secchi, "Decentralized connectivity maintenance for cooperative control of mobile robotic systems," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1411–1423, 2013.

[6] E. Montijano, J. I. Montijano, and C. Sagues, "Adaptive consensus and algebraic connectivity estimation in sensor networks with chebyshev polynomials," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*. IEEE, 2011, pp. 4296–4301.

[7] G. Hollinger and S. Singh, "Multi-robot coordination with periodic connectivity," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 4457–4462.

[8] Y. Kantaros, M. Guo, and M. M. Zavlanos, "Temporal logic task planning and intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4105–4120, 2019.

[9] M. M. Zavlanos, "Synchronous rendezvous of very-low-range wireless agents," in *49th IEEE Conference on Decision and Control (CDC)*. IEEE, 2010, pp. 4740–4745.

[10] Y. Kantaros and M. M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3109–3121, 2016.

[11] ——, "Simultaneous intermittent communication control and path optimization in networks of mobile robots," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 1794–1799.

[12] ——, "Distributed intermittent communication control of mobile robot networks under time-critical dynamic tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[13] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.

[14] A. Viseras, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, and L. Merino, "Decentralized multi-agent exploration with online-learning of gaussian processes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4222–4229.

[15] A. Viseras, D. Shutin, and L. Merino, "Robotic active information gathering for spatial field reconstruction with rapidly-exploring random trees and online learning of gaussian processes," *Sensors*, vol. 19, no. 5, p. 1016, 2019.

[16] Y. T. Tan, A. Kunapareddy, and M. Kobilarov, "Gaussian process adaptive sampling using the cross-entropy method for environmental sensing and monitoring," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6220–6227.

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

[18] M. Kirby, *Geometric data analysis: an empirical approach to dimensionality reduction and the study of patterns*. John Wiley & Sons, Inc., 2000.

[19] R. Everson and L. Sirovich, "Karhunen–loeve procedure for gappy data," *JOSA A*, vol. 12, no. 8, pp. 1657–1664, 1995.

[20] A. A. Alonso, I. G. Kevrekidis, J. R. Banga, and C. E. Frouzakis, "Optimal sensor location and reduced order observer design for distributed process systems," *Computers & chemical engineering*, vol. 28, no. 1-2, pp. 27–35, 2004.

[21] K. Willcox, "Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition," *Computers & fluids*, vol. 35, no. 2, pp. 208–226, 2006.

[22] B. Peherstorfer and K. Willcox, "Dynamic data-driven model reduction: adapting reduced models from incomplete data," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 3, no. 1, pp. 1–22, 2016.

[23] T. Salam and M. A. Hsieh, "Adaptive sampling and reduced-order modeling of dynamic processes by robot teams," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 477–484, 2019.

[24] R. Khodayi-mehr, Y. Kantaros, and M. M. Zavlanos, "Distributed state estimation using intermittently connected robot networks," *IEEE Transactions on Robotics*, 2019.

[25] Y. Kantaros and M. M. Zavlanos, "Intermittent connectivity control in mobile robot networks," in *2015 49th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2015, pp. 1125–1129.

[26] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[27] C. Godsil and G. F. Royle, *Algebraic graph theory*. Springer Science & Business Media, 2013, vol. 207.

[28] GPy, "GPy: A Gaussian process framework in Python," http://github.com/SheffieldML/GPy, since 2012.

[29] W. Krzanowski, *Principles of multivariate analysis*. OUP Oxford, 2000, vol. 23.

[30] J. C. Gower, "Generalized Procrustes analysis," *Psychometrika*, vol. 40, no. 1, pp. 33–51, 1975.