Distributed Sampling and Tracking of Dynamic Processes with Robot Teams

Tahiya Salam[1] and M. Ani Hsieh[1]

Abstract— **This paper presents a strategy to enable a team of mobile robots to adaptively sample and track a dynamic process. We propose a distributed strategy, where robots collect sparse sensor measurements, create a reduced-order model (ROM) of a spatio-temporal process, and use this model to estimate missing measurements of the dynamic process. The robots then use the inferences to adapt the model and reconfigure their sensing locations. The key contributions of this process are two-fold: 1) leveraging the dynamics of the process of interest to determine where to sample and how to estimate the process, and 2) maintaining fully distributed models, sensor measurements, and estimates of the time-varying process. We illustrate the application of the proposed solution in simulation and compare it to centralized and global approaches. We also test our approach with physical marine robots tracking a process in a water tank.**

I. INTRODUCTION

Being able to track and predict information about dynamic processes deepens our understanding of biological, chemical, and physical phenomena in the environment. Often, these dynamic processes exhibit complex, spatio-temporal behaviors. Mobile robots are particularly well-suited to track these process because of their abilities to carry sensors and adapt their sensing locations. Robots can be used to support a wide range of activities dependent on tracking and predicting processes that vary across both space and time, such as tracking oil spills in water or pollutant concentrations in air for environmental monitoring, gas leaks for pipeline repair, or forest fire boundaries for search and rescue. For these processes, autonomous mobile robots modeling the environment and determining where to gather sensor measurements are cheaper than global tracking systems and more adaptive than fixed sensors. The process dynamics provide rich information about its spatial and temporal dependencies. Thus, robots should leverage their mobility and sensing capabilities to adequately model and estimate the environment.

However, given that they are inherently complicated, spatio-temporal processes are often difficult to model in a meaningful way, and even in scenarios where representations are available, they are often high-dimensional, which is computationally burdensome. Additionally, these processes often occur in dynamic, uncertain environments, so robots should not rely on centralized techniques to mitigate the effects of communication constraints and robot failures. The question then still remains as to how robots can leverage the spatio-temporal dynamics of the process to model and estimate the environment in a distributed way.

Previous works have studied multi-robot coordination for environmental monitoring, mapping, and modeling. The works most related to this paper fall under two categories: providing coverage and maximizing information (or alternatively minimizing entropy) using Gaussian processes (GPs). In [1], a technique was developed for providing optimal sensor placement in an environment, where a weighting function accounting for sensing quality and coverage of the environment has to be known a priori. This work has been extended in several ways. In [2], the stochastic uncertainty of modeling the weighting function is incorporated online to optimize the deployment of the sensors. In [3], authors propose a method that does not rely on weighting functions being known a priori and instead learns them online. Despite their advantages, coverage control techniques do not take into account the equations governing the dynamic process and thus the determined placement of sensors may not capture the relevant features needed to estimate the field.

GPs are widely used in modeling spatio-temporal processes. The framework in [4] models the environment as GPs, learns confidence measures on the uncertainty of the model, and utilizes this uncertainty in path planning to minimize risk. In [5], authors also use GPs to model the desired quantity of interest for monitoring as part of a stochastic optimization strategy to minimize regret when collecting samples. In [6], robots use GPs to create a map of the environment, partition the space to determine nearby locations, and selects future sampling locations based on reducing the entropy in the map. The work presented in [7] adapts the model in real-time based on observations and optimizes sensing locations based on the changing model. However, as with coverage control techniques, GPs neglect the principle dynamics of the fluid flow. GPs may not capture important nonlinearities of the process of interest and are inappropriate for functions with varying smoothness or scale of variation.

Other approaches, such as [8] and [9] study sensor placements. However, sensor placement method do not leverage robots' mobility and do not account for their ability to move locations. In [10] and [11], authors consider the fusion and control of active sensor networks. In this work, we are more interested in using the dynamics of the process to inform the robots' modeling and estimation.

The contributions of this work are two-fold. First, we propose a framework that uses the dynamics of the process to allow robots to compactly model the environment, infer

We gratefully acknowledge the support of ARL DCIST CRA W911NF-17-2-0181 and National Science Foundation (NSF) grant IIS-1812319.

[1]T. Salam and M.A. Hsieh are with the GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA {tsalam, mya}@seas.upenn.edu

properties of the environment using sparse sensing data, and assimilate these inferences to update the model and determine if they should navigate to new sensing locations. The framework allows for a non-balanced assignment of regions to robots, where robots are able to estimate properties of the environment in regions for which there is no available sensing data. Second, we exploit the structures of the model and inference techniques to allow for the process model and estimated field values to be computed in a fully distributed fashion. Unlike other works in this domain, we explicitly use the dominant spatial and temporal characteristics of the dynamic process in order to allow robots to determine sensing locations and adapt the model and estimations.

II. PROBLEM STATEMENT

Consider tracking a dynamic process in a continuous spatial region $R \in \mathbb{R}^2$ or $R \in \mathbb{R}^3$. R can be discretized into n spatial points such that at each of the points, a measurement, such as concentration or temperature, can be obtained and provides a representation of the spatio-temporal dynamic process, P. The n spatial points can be grouped into s non-overlapping regions.

Consider a team of q robots, where each robot is equipped with a sensor that is capable of sensing across each of the s sensing regions. The quality and range of the sensors on the robots are homogeneous. Furthermore, the robots are capable of localization and can communicate small packets of information, such as matrices, with their neighbors. To begin, robots can estimate the dynamic process using either historical data or some forecast model. Each robot only maintains a model of the environment for its assigned regions. Each robot is able to share a compact amount of information about its model with its neighbors and use aggregated information about the models to determine the optimal sensing regions to achieve this estimation. The communication network only needs to maintain connectivity. Robots can move and create or break their connections with other robots, so long as the graph topology remains connected.

Problem Statement: Given a region R, a team of q homogeneous robots such that $q << s$, develop an adaptive sampling and tracking strategy to track a dynamic process P, where each robot is capable of sensing all the points within the region it is assigned to.

In our solution to this problem, each robot is assigned its own sensing region. The $q - s$ regions that are not being sensed are assigned arbitrarily to robots. Thus, robots do not need to keep a full model of the environment. Though robots are taking sparse measurements, they are able to produce the least-squares error estimation of the dynamic process. Robots can update their models in a distributed fashion, even though the process exhibits complex relationships over the regions. Each robot is able to adapt its existing model based on new sensing information in its own region and a reduced representation of the new sensing information from other robots. All of the robots can reconfigure their locations based on their updated models from the new sensing data.

III. METHODOLOGY

The following section will describe the procedures for a) obtaining a reduced order model, b) selecting sensing locations for optimal field reconstruction, and c) using new measurements to obtain estimates of the field, update the reduced order model and select new sensing locations. We will begin by describing the method as a fully centralized procedure and later describe how to implement the procedure in a distributed fashion.

A. Reduced Order Model using Proper Orthogonal Decomposition

Fluid flows are infinite dimensional fields that can widely vary temporally and spatially and exhibit complex behavior. In order to extract the dominant dynamics of these fields, techniques for modal analysis are often used to construct a low-dimensional approximation of flows. We use the proper orthogonal decomposition (POD) [12], [13] to obtain a representative reduced order model of the flow field.

For POD analysis, m snapshots of the field are collected, either through experimentation or numerical simulations, such that at each time $t = 1, ..., m$, $\boldsymbol{x(t)} = [x_1(t), ..., x_n(t)]^\top$, where n is the spatial dimension of some discretization of the flow field. A covariance matrix is constructed as

$$\boldsymbol{K} = \frac{1}{l} \sum_{t=1}^{m} \boldsymbol{x(t)} \boldsymbol{x(t)}^\top = \frac{1}{m} \boldsymbol{X} \boldsymbol{X}^\top, \tag{1}$$

where $\boldsymbol{X} \in \mathbb{R}^{n \times m}$ with its columns as $\boldsymbol{x(t)}$.

The low-dimensional basis is created by solving the symmetric eigenvalue problem

$$\boldsymbol{K} \boldsymbol{\phi_i} = \lambda_i \boldsymbol{\phi_i}, \tag{2}$$

where $\boldsymbol{K}$ has n eigenvalues such that $\lambda_1 \geq \lambda_2 ... \geq \lambda_n \geq 0$ and the eigenvectors $\boldsymbol{\phi}$ are pairwise orthonormal.

The original basis is then truncated into a new basis $\boldsymbol{\Phi}$ by choosing k eigenvectors that capture a user-defined fraction, E, of the total variance of the system, such that their eigenvalues satisfy

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq E. \tag{3}$$

Thus, each term $\boldsymbol{x(t)}$ can be written as

$$\boldsymbol{x(t)} = \boldsymbol{\Phi} \boldsymbol{c(t)}, \tag{4}$$

where $\boldsymbol{c(t)} = [c_1(t), ..., c_k(t)]^\top$ holds time-dependent coefficients and $\boldsymbol{\Phi} \in \mathbb{R}^{n \times k}$ with its columns as $\boldsymbol{\phi_1}, ..., \boldsymbol{\phi_k}$. The low-dimensional, orthogonal subspace associated with $\boldsymbol{\Phi}$ is an optimal approximation of the data with respect to minimizing least squares error.

B. Optimizing Robot Locations for Field Reconstruction

Given a low-dimensional representation of the subspace on which the data is located, the properties of the orthogonal bases can be used to compute the optimal set of locations to place robots in order to reconstruct the field from sparse data in real-time.

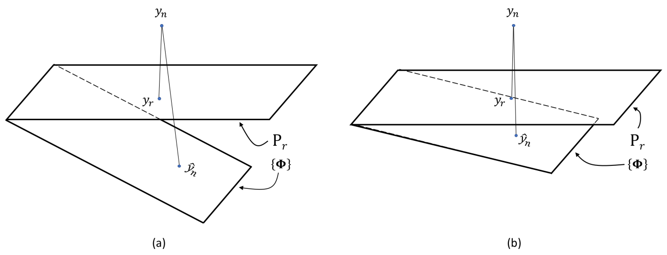


Fig. 1. Geometric interpretation of maximizing minimum eigenvalue. The data point y_n containing all the measurements of the field is projected as y_r onto the subspace P_r, where y_r equivalently represents a vector of just the sensor measurements. As the angle between P_r and the subspace associated with $\mathbf{\Phi}$ decreases, so does the projection error of $\hat{y}_r$.

Consider the problem of reconstructing a field from measurements in q arbitrary sensing regions. Given s total sensing regions, let $S \subset \{1, ..., s\}$ where S contains the locations of the q sensing regions. Measurements of the field are collected over the q sensing regions as $\boldsymbol{y_r}(\boldsymbol{t})$ for sensing region $r \in S$, where each $\boldsymbol{y_r}(\boldsymbol{t}) \in \mathbb{R}^{n_r \times 1}$ for n_r points of measurements in region r. Let matrices $\boldsymbol{\Phi_r} \in \mathbb{R}^{n_r \times k}$ such that the rows of $\boldsymbol{\Phi_r}$ are the rows of $\boldsymbol{\Phi}$ corresponding to the locations in sensing region r. Using the gappy POD [13]–[15], the time-dependent coefficients that minimize the distance between $\boldsymbol{y}(\boldsymbol{t})$, sensor values, and $\hat{\boldsymbol{y}}(\boldsymbol{t})$, the projection of sensor values onto the subspace associated with the vectors $\{\boldsymbol{\Phi_r}\}_{r \in S}$ can be found using

$$\hat{\boldsymbol{c}}(\boldsymbol{t}) = \boldsymbol{AB}$$
$$\text{for } \boldsymbol{A} = \sum_{r \in S} \boldsymbol{\Phi_r}^\top \boldsymbol{\Phi_r} \text{ and } \boldsymbol{B} = \sum_{r \in S} \boldsymbol{\Phi_r}^\top \boldsymbol{y_r}(\boldsymbol{t}), \tag{5}$$

where this time-dependent coefficient $\hat{\boldsymbol{c}}(\boldsymbol{t})$ is then applied to $\boldsymbol{\Phi}$ as in (4) to recover the missing values of the field.

Next, we discuss how to select q sensing regions from the set of S possible regions to optimize the reconstruction of the full field using only measurements from the q regions. The matrix $\boldsymbol{A} \in \mathbb{R}^{k \times k}$ depends only on the set of S sensing regions and is not time varying. If measurements from all sensing regions were used, the matrix $\boldsymbol{A}$ would be the identity matrix since $\boldsymbol{A} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{I}$ for $\boldsymbol{\Phi}$ containing orthonormal columns and the coefficients $\hat{\boldsymbol{c}}(\boldsymbol{t})$ could be calculated exactly using (4). However, since only some and not all sensing regions are being used, the sensing regions should be chosen such that the rows of the eigenvectors corresponding to these sensing regions create a basis that is close to orthogonal. Additionally, [16] provides a criteria for selecting the optimal set of sensing regions S as

$$\max_S \min_i \lambda_i(\boldsymbol{A}), \tag{6}$$

where maximizing the minimum eigenvalue of $\boldsymbol{A}$ in turn minimizes the maximum angle between the subspace associated with $\boldsymbol{\Phi}$ and P_r, the subspace associated with using only the sensor measurements, as shown in Fig. 1.

Let a_{ij} represent entries in $\boldsymbol{A}$ and $r_i = \sum_{j \neq i} a_{ij}$, the Gershgorin circle theorem [17] states that all eigenvalues of $\boldsymbol{A}$ lie in a circle centered at a_{ii} with radius r_i. Using this property of the eigenvalues of $\boldsymbol{A}$, an estimation of (6) is given by

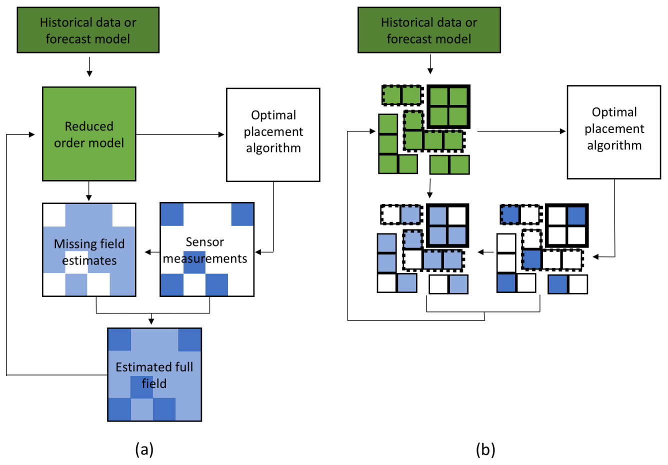

Historical data or forecast model
Reduced order model
Optimal placement algorithm
Missing field estimates
Sensor measurements
Estimated full field
Historical data or forecast model
Optimal placement algorithm
(a)
(b)

Fig. 2. Comparison of centralized framework for model-inference-assimilation scheme and corresponding distributed scheme. In (a), the centralized frameworks keeps a global model which is combined with sensor measurements to estimate the field and update the model. In (b), the distributed framework allows robots take sensing measurements at specific regions and estimate the values of the field using the current model and their neighbors' data. These estimates are used to update the model at robots' assigned locations.

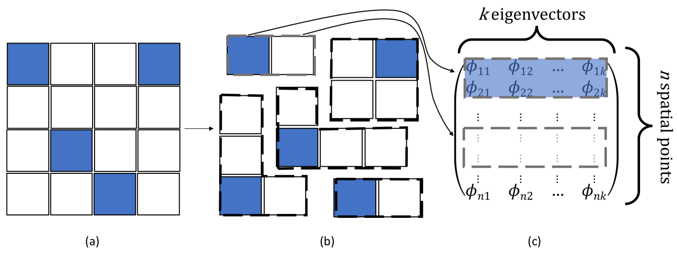
$$\max_S \min_i a_{ii}, \tag{7}$$

where maximizing the minimum diagonal element of $\boldsymbol{A}$ seeks the set S that results in $\boldsymbol{A}$ being both close to orthonormal and minimizing the distance between the the subspaces associated with $\boldsymbol{\Phi}$ and $\{\boldsymbol{\Phi_r}\}_{r \in S}$. The algorithm developed in [16] and extended in [8] is then used to find the set S that satisfies criteria (7).

C. Adaptive Computation of Reduced Order Model and Robot Locations

The techniques described in [8], [14]–[16] rely on computing full POD basis vectors using snapshots of data over the process. Instead, we propose a method that dynamically adapts the POD basis vectors using incoming data and reconfigures the position of the robots based on the adapted POD.

To begin, POD basis vectors $\boldsymbol{\Phi}(\boldsymbol{t_1})$ are computed using T arbitrary snapshots $\{\boldsymbol{x}(\boldsymbol{t_{1,1}}), ..., \boldsymbol{x}(\boldsymbol{t_{1,T}})\}$ where $\boldsymbol{x}(\boldsymbol{t}) \in \mathbb{R}^{n \times 1}$. The snapshots are gathered from either experiments or numerical simulation based on the equations governing the process of interest. A set of sensing regions $S(t_1)$ is selected according to the algorithm described above, where robots are then deployed to collect measurements. Estimates of the field are computed using $\boldsymbol{y_r}(\boldsymbol{t})$ for $r \in S_{t_1}$ the collected sensing data, $\{\boldsymbol{\Phi_r}(\boldsymbol{t_1})\}_{r \in S_{t_1}}$ the POD basis over the sensing regions, and the relationship (5). The new inferences are assimilated into the covariance matrix $\boldsymbol{R}$ as in (1) as new snapshots, at which point the POD basis vectors are recomputed as $\boldsymbol{\Phi}(\boldsymbol{t_2})$ and a new set of sensing regions $S(t_2)$ are found. This procedure is repeated for the duration of the mission.



k eigenvectors
ϕ_{11} ϕ_{12} ... ϕ_{1k}
ϕ_{21} ϕ_{22} ... ϕ_{2k}
ϕ_{n1} ϕ_{n2} ... ϕ_{nk}
n spatial points
(a)
(b)
(c)

Fig. 3. Visualization of spatial points corresponding to rows of eigenvectors in POD basis. In (a), the full field is shown, where each region is a set of points in the field. Blue regions are monitored by the robots and thus are the regions with sensor measurements, while the field in the white regions are inferred using the ROM. In (b), the dashed lines contain the regions for which each robot either takes measurements or estimates values. The matrix in (c) shows rows in the POD basis that correspond to a single robot's assigned regions indicated with the gray dashed lines.

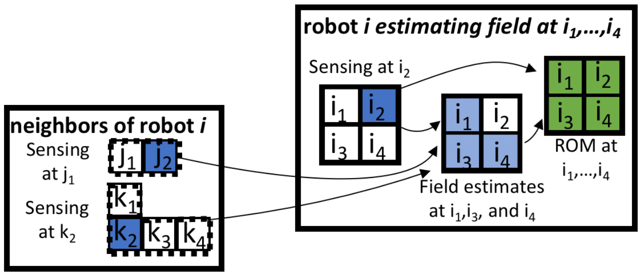


Fig. 4. Regions assigned to robot for sensing, estimating and modeling. The robot senses at a location and is assigned arbitrary regions for which no sensor measurements exists. The robot uses its own ROM over its assigned regions, its collected sensor measurements, and sensor measurements from its neighbors to estimate the missing values of the field, all of which will then be used to update its reduced order model.

D. Distributed Algorithm

The procedure described above can be implemented in a distributed fashion. A comparison of the centralized and distributed approach is shown in Fig. 2. The model of the environment is represented as the matrix of eigenvectors, where each row of the matrix corresponds to a spatial point. These can be distributed to different robots, and robots can keep on-board the rows corresponding to their assigned regions as shown in Fig. 3. The push-sum algorithm [18] is leveraged to allow for robots to maintain field measurements only over their respective regions, while occasionally exchanging small packets of information with their neighbors to understand the full field and recompute optimal sensing locations, shown in Fig. 4. Instead of having all robots compute the estimates of field measurements for regions without sensor measurements, these regions are assigned arbitrarily to robots, such that the values at each region are only estimated by one robot.

The push-sum algorithm is described here. Suppose some matrix $\boldsymbol{P} = \sum_i \boldsymbol{P_i}$. Further, there exists agents where each agent i has access to matrix $\boldsymbol{P_i}$ and can communicate with its neighbors N_i. Let $\boldsymbol{M}$ be an arbitrary stochastic matrix such that $m_{ij} = 0$ if agent i is not a neighbor to agent j. A stochastic matrix is used to exploit the equivalence between averaging and Markov chains; we refer the interested reader to [18] for more details. Each agent i can compute $\hat{\boldsymbol{P}}^{\boldsymbol{i}}$, its own estimate of $\boldsymbol{P}$, as shown in Algorithm 1:

Algorithm 1: Push-sum algorithm

```
Input  : $P_i$ from each robot
Output: estimates $\hat{P}^i$ for each robot of sum of all $P_i$
select $\hat{i}$, let $w_{\hat{i}} = 1$, and $w_i = 0\ \forall \hat{i} \neq i$;
for each robot i in parallel do
    $\hat{P}_i = P_i$;
    for loop do
        $\hat{P}_i = \sum_{j \in N_i} m_{ij} \hat{P}_j$;
        $w_i = \sum_{j \in N_i} m_{ij} w_j$;
    end
    return $\hat{P}^i = \frac{\hat{P}_i}{w_i}$
end
```

The push-sum algorithm is used for the distributed computations of a) the covariance matrix from data at sensing regions, b) the eigenvectors and eigenvalues for the POD basis vectors, and c) the time-dependent coefficients for estimating the full field. First, we show how to compute the eigenvalues and eigenvectors of a pre-computed covariance matrix in a distributed fashion using existing techniques. Then, we will bypass the need to directly compute the covariance matrix and instead compute the eigenvalues and eigenvectors from on-board data in a distributed setting.

The method of orthogonal iteration allows for the computation of the top k eigenvectors and eigenvalues of a symmetric matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ using Algorithm 2:

Algorithm 2: Orthogonal iteration

```
set $Q \in \mathbb{R}^{n\times k}$ with random elements;
for loop do
    $V = KQ$;
    $QR \overset{QR}{=} V$;
end
return columns of $Q$ as eigenvectors;
return diagonals of $R$ as eigenvalues
```

The distributed computation of Algorithm 2 rests on the following matrix properties, shown in detail in [18]. However, while [18] assumes a bijection between the rows of the covariance matrix and the robots performing the computation, we show here that this is not strictly necessary, allowing for a non-balanced assignment of rows to robots. Every row of the covariance matrix $\boldsymbol{K}$ corresponds to a location in the field. Each robot i is assigned the set of rows, L_i, of the matrices $\boldsymbol{K}$ and $\boldsymbol{Q}$ corresponding to the spatial points in its sensing region and some arbitrary subset of the spatial points of the regions not covered by any robot. Let $L = L_i \cup (\bigcup_{j\in N_i} L_j)$, where N_i is the set of neighbors of robot i so that L is the set that contains all the spatial points assigned to robot i and its neighbors. To start, the rows $\boldsymbol{V_l}$ for $l \in L_i$ can be estimated as a linear combination of the random row vectors $\boldsymbol{Q_m}$ over all $m \in L$ with coefficients a_{lm}. Then each robot can use an estimate of the matrix $\boldsymbol{R}$ to apply to its set of rows $\boldsymbol{V_l}$ for $l \in L_i$ to find the corresponding rows $\boldsymbol{Q_l}$ for the next iteration of orthogonal iteration. An estimate of $\boldsymbol{R}$ is found by leveraging the relation:

$$\boldsymbol{W} = \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{R}^\top \boldsymbol{Q}^\top \boldsymbol{Q} \boldsymbol{R}, \tag{8}$$

where $\boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}$ since $\boldsymbol{Q}$ orthonormal and $\boldsymbol{R}$ is a unique upper triangular matrix. Since $\boldsymbol{W} = \sum_{c=1}^{n} \boldsymbol{V_c}^\top \boldsymbol{V_c}$, each agent can compute $\boldsymbol{W_i}$ over its sensing region as $\boldsymbol{W_i} = \sum_{l \in L_i} \boldsymbol{V_l}^\top \boldsymbol{V_l}$. Using the push-sum algorithm, each agent can compute estimates $\hat{\boldsymbol{W}}^{\boldsymbol{i}}$, perform a Cholesky factorization to compute $\hat{\boldsymbol{W}}^{\boldsymbol{i}} = \hat{\boldsymbol{R}}_{\boldsymbol{i}}^\top \hat{\boldsymbol{R}}_{\boldsymbol{i}}$, and apply $\hat{\boldsymbol{R}}_{\boldsymbol{i}}^{-1}$ to its rows $\boldsymbol{V_l}$ to compute $\boldsymbol{Q_l} = \boldsymbol{V_l} \hat{\boldsymbol{R}}_{\boldsymbol{i}}^{-1}$. $\boldsymbol{Q_l}$ is then used in the next iteration of the orthonormal iteration algorithm. This requires the entries the covariance matrix $\boldsymbol{K}$ to be known and communication between neighbors to estimate the values $\boldsymbol{V_l}$.

We leverage the following relation presented in [19] to eliminate the centralized computation of $\boldsymbol{K} = \frac{1}{m} \boldsymbol{X} \boldsymbol{X}^\top$ and instead allow for the distributed computation of the eigenvectors and eigenvalues of $\boldsymbol{K}$ directly from $\boldsymbol{X}$ without explicitly constructing $\boldsymbol{K}$. Let the $Diag$ operator create a diagonal matrix out of a given vector and the $diag$ operator extract the diagonal elements of a given matrix. Each column $\boldsymbol{v_j}$ can be computed as

$$\begin{aligned} \boldsymbol{v_j} &= \frac{1}{m} \boldsymbol{X} \boldsymbol{X}^\top \boldsymbol{q_j} \\ &= \frac{1}{m} diag(\boldsymbol{X} \boldsymbol{X}^\top \boldsymbol{q_j} \boldsymbol{1}^\top) \\ &= \frac{1}{m} diag(\boldsymbol{X} \boldsymbol{X}^\top Diag(\boldsymbol{q_j}) \boldsymbol{1} \boldsymbol{1}^\top) \\ &= \frac{1}{m} diag[\boldsymbol{X} (\boldsymbol{1} \boldsymbol{1}^\top Diag(\boldsymbol{q_j}) \boldsymbol{X})^\top]. \end{aligned} \tag{9}$$

For $\boldsymbol{q_j} = [q_j(1), ..., q_j(n)]$, the l^{th} row of $Diag(\boldsymbol{q_j}) \boldsymbol{X}$ is equal to $q_j(l) \boldsymbol{X_l}$. Furthermore, the quantity $\boldsymbol{1} \boldsymbol{1}^\top Diag(\boldsymbol{q_j}) \boldsymbol{X}$ is a matrix where each row is equal to the sum of all the rows of $Diag(\boldsymbol{q_j}) \boldsymbol{X}$. Thus, only the quantity $\boldsymbol{F} = \sum_{c=1}^{n} \boldsymbol{D_c}$, where $\boldsymbol{D} = Diag(\boldsymbol{q_j}) \boldsymbol{X}$ and $\boldsymbol{D_c}$ denotes the rows of $\boldsymbol{D}$, needs to be computed. Each robot can individually compute the quantity $\boldsymbol{F_i} = \sum_{l \in L_i} q_j(l) \boldsymbol{X_l}$ and then can compute estimates $\hat{\boldsymbol{F}}^{\boldsymbol{i}}$ using the push-sum algorithm. Then, the l^{th} row of $\boldsymbol{v_j}$ is equal to $\frac{1}{m} \hat{\boldsymbol{F}}^{\boldsymbol{i}\top} \boldsymbol{X_l}$. This is carried for all k columns of $\boldsymbol{Q}$ and $\boldsymbol{V}$. The full procedure for distributed computation of eigenvectors and eigenvalues is shown in Algorithm 3:

To estimate the time-dependent coefficients, each robot can compute its own $\boldsymbol{A_i}$ and $\boldsymbol{B_i}$ as in (5) and use the push-sum algorithm to compute estimates $\hat{\boldsymbol{A}}^{\boldsymbol{i}}$ and $\hat{\boldsymbol{B}}^{\boldsymbol{i}}$. Then, robots can compute the estimate $\hat{\boldsymbol{c}}_{\boldsymbol{i}} = \hat{\boldsymbol{A}}^{\boldsymbol{i}} \hat{\boldsymbol{B}}^{\boldsymbol{i}}$ and apply coefficients $\hat{\boldsymbol{c}}_{\boldsymbol{i}}$ to the rows $\boldsymbol{Q_l}$ to estimate the values $\hat{\boldsymbol{y}}_{\boldsymbol{l}} = \boldsymbol{Q_l} \hat{\boldsymbol{c}}_{\boldsymbol{i}}$ that are missing in the regions $l \in L_i$.

Algorithm 3: Distributed eigenvectors from data

```
set $\boldsymbol{Q} \in \mathbb{R}^{n \times k}$ with random elements;
select $\hat{i}$, let $w_{\hat{i}} = 1$, and $w_i = 0 \ \forall \hat{i} \neq i$;
for loop do
    for each row r of Q in parallel do
        for each robot i do
            $\boldsymbol{Z_i} = \sum_{l \in L_i} q_{lr} \boldsymbol{X_l}$;
            Compute $\hat{\boldsymbol{Z}}_{\boldsymbol{i}}$ with Algorithm 1 (push-sum);
            for $l \in L_i$ do
                $v_{lr} = \frac{1}{n} \hat{\boldsymbol{Z}}_{\boldsymbol{i}}^\top \boldsymbol{X_l}$;
            end
            $\boldsymbol{W_i} = \sum_{l \in L_i} \boldsymbol{V_l}^\top \boldsymbol{V_l}$ ;
            Compute $\hat{\boldsymbol{W}}_{\boldsymbol{i}}$ with Algorithm 1 (push-sum);
            Use Cholesky factorization $\hat{\boldsymbol{W}}^{\boldsymbol{i}} = \hat{\boldsymbol{R}}_{\boldsymbol{i}}^\top \hat{\boldsymbol{R}}_{\boldsymbol{i}}$;
            $\boldsymbol{Q_l} = \boldsymbol{V_l} \hat{\boldsymbol{R}}_{\boldsymbol{i}}^{-1}$ ;
        end
    end
end
return rows $\boldsymbol{Q_i}$ as eigenvectors for robot $i$;
return diagonals of $\hat{\boldsymbol{R}}_{\boldsymbol{i}}$ as eigenvalues for robot $i$;
```

E. Task Allocation

Using the distributed algorithm, individual robots can adaptively calculate their respective eigenvectors and eigenvalues. They can then share the necessary properties of their eigenvectors to each compute the optimal sensing locations. After finding the set of optimal sensing locations, robots are assigned to locations as to minimize the total cumulative path traveled by all robots.

IV. SIMULATIONS AND EXPERIMENTS

Analyses were carried out both in simulation and on physical robots. In simulation, a 1x1 m 2-dimensional grid space was modeled using video data from an experimental flow tank for low Reynolds numbers. The grid space was discretized into 9 non-overlapping regions. 4 robots were simulated in the field. A concentration field was created by placing dye in an experimental flow tank and recorded on video. Concentration values of the dye in the tank were estimated for each time step from the grayscale values of the pixels of the images from a grayscale video of the LoRe tank. Concentration values of the field were gathered for 100 equally spaced times across the time series, and noise was then added to these concentration values. These were then used to the construct the initial POD basis for the distributed optimal placement algorithm. Data was collected for another 100 sequential times before adapting the POD basis and recomputing the optimal placement algorithm. The distributed optimal placement algorithm was compared to the centralized optimal placement algorithm, where all computations occur on a centralized system and are broadcasted to robots. Additionally, the distributed optimal placement algorithm were compared with radial basis function (RBF) interpolation schemes. Radial basis function interpolations

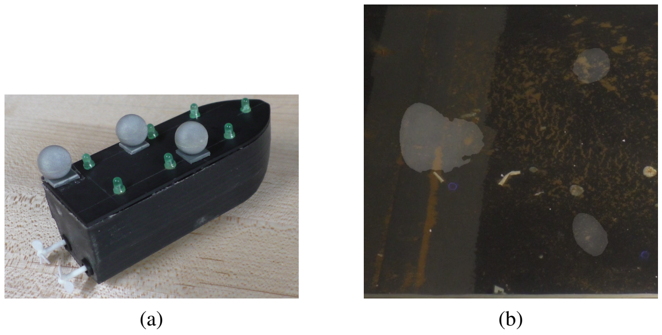


Fig. 5. Experimental setup with marine robots. Robot boat (a) equipped with pose information from motion capture system and ability to communicate. Water tank with projection of dynamic process depicted in white in (b).

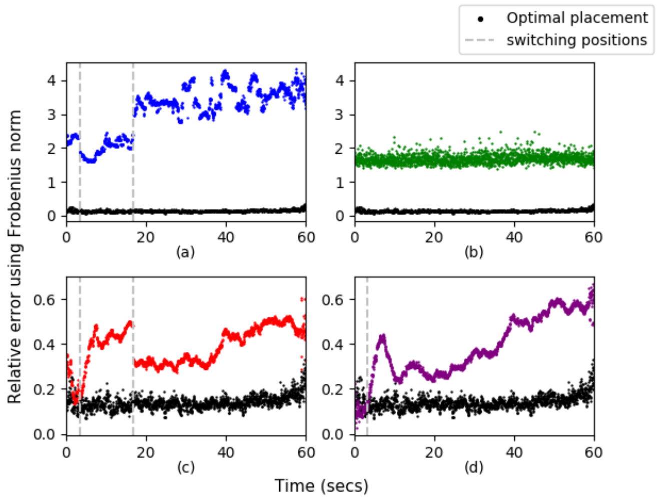


Fig. 6. Norm-wise relative error between field simulation and various field estimation algorithms at each time step. Estimations are calculated using (a) RBF using sensing data, (b) RBF using random points, (c) proposed distributed algorithm, (d) centralized version of proposed algorithm. Black circles represent estimations calculated using the optimal placement and POD basis. Gray vertical lines indicate robots switching their placement.

were computed using sensing data from the optimal sensing locations from the distributed method and using randomly selected points across the entire field. All of these methods were compared against the optimal placement algorithm, which was calculated using noiseless data across the entirety of the time series.

Experiments were carried out in a 5x3 m water tank using 4 marine robots, shown in Fig. 5a. The concentration field was mapped and projected onto the tank using the video from the LoRe tank, shown in Fig. 5b. The robots then tracked the projected concentration field using the distributed algorithm.

V. RESULTS

The simulation results of the comparison of the various field estimation schemes over the entire time series are shown in Fig. 6. The Frobenius norm-wise error between the actual concentration value and the estimated field computed using various algorithms was computed for each time step.

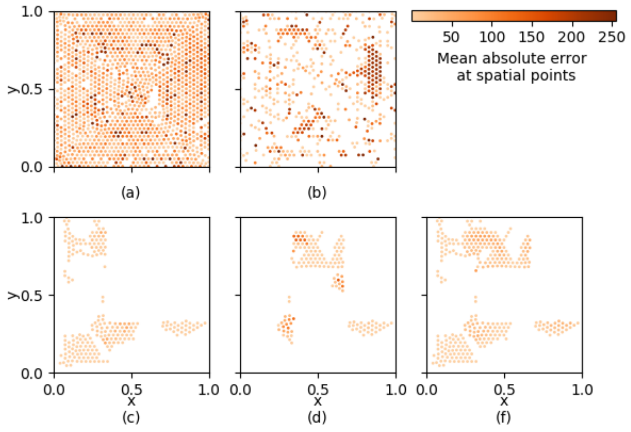


Fig. 7. Mean absolute error at spatial points calculated over time series for various field estimation algorithms. Concentrations at points are calculated using (a) RBF using sensing data, (b) RBF using random points, (c) optimal placement and POD basis, (d) centralized version of proposed algorithm, and (f) proposed distributed algorithm.

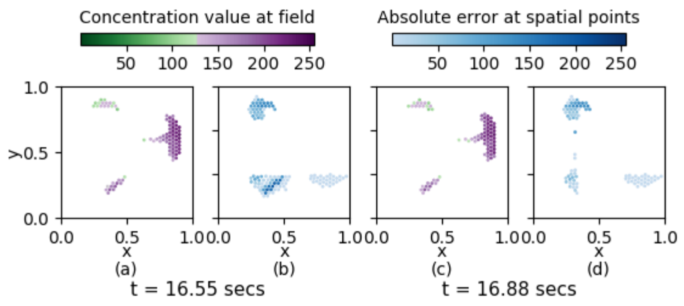


Fig. 8. Concentration field and absolute error at spatial points before and after robots switch locations using distributed placement algorithm. Concentration field (a) and absolute errors (b) are before the switch; concentration field (c) and absolute errors (d) are after assimilating data and switching positions.

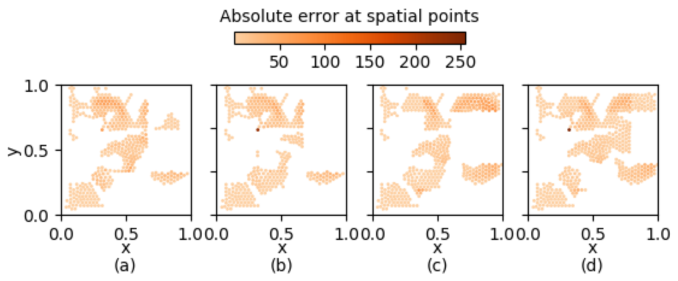


Fig. 9. Mean absolute error at spatial points calculated over time series between field and distributed algorithm for various discretizations of field, numbers of robots, and snapshots used to compute original POD basis. Algorithm tested for (a) 4 robots, 9 regions, and 100 snapshots, for (b) 4 robots, 9 regions, and 500 snapshots, for (c)8 robots, 25 regions, and 100 snapshots, and for (d) 8 robots, 25 regions, and 500 snapshots.

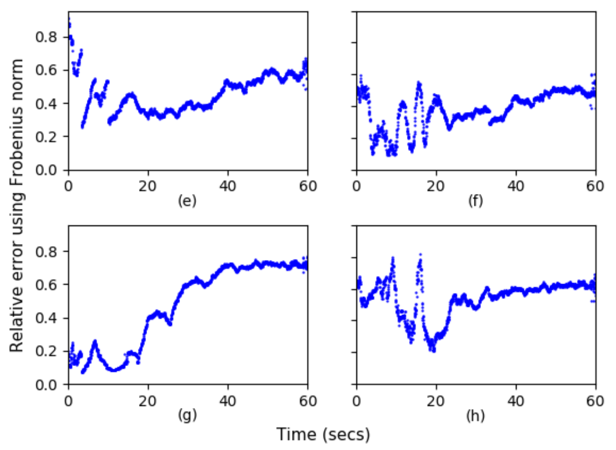


Fig. 10. Norm-wise relative error between field and distributed algorithm for various discretizations of field, numbers of robots, and snapshots used to compute original POD basis. Algorithm tested for (a) 4 robots, 9 regions, and 100 snapshots, for (b) 4 robots, 9 regions, and 500 snapshots, for (c)8 robots, 25 regions, and 100 snapshots, and for (d) 8 robots, 25 regions, and 500 snapshots.

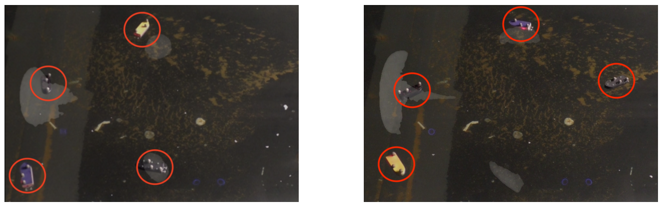


Fig. 11. Robotic boats tracking dynamic process in water tank. The dynamic process is shown in white, and robots are circled in red. The robots assume positions based on the initial POD basis in (a). The robots switch positions after collected sensor measurements and updating their model in (b).

Both RBF interpolation schemes perform significantly worse than the optimal placement algorithms. Even in the case of the RBF with randomly selected sensor points, the field estimation is approximately an order of magnitude worse than the optimal field estimation. However, the distributed algorithm and centralized algorithm perform just slightly worse than the optimal field estimation.

The mean absolute error of the various field estimation schemes is shown in Fig. 7. The RBF interpolation scheme using the data from the sensor measurements results in high error across the field, as it is unable to adequately estimate values in regions far from the sensing locations. The RBF interpolation scheme using random points fails to capture the interesting features of the process. The distributed algorithm fails in similar areas as compared to the optimal placement algorithm. This can be attributed to little to no data being collected over these regions, which makes it difficult to estimate the concentration values over these areas. Additionally, the distributed algorithm performs slightly worse than the centralized algorithm. This is expected given the fact that distributed algorithm uses only local information in its computation of the field estimate.

The adaptive nature of the algorithm allows robots to rectify tracking errors by recomputing the POD basis and possibly reassigning the sensing locations. This is shown in Fig. 8 where robots are able to improve field measurements for areas of high error after reassimilating their collected data to determine new sensing locations and a new POD basis.

The distributed algorithm demonstrates consistent results across various discretizations of the spatial region, various numbers of robots, and various initial models of the dynamic process, as shown in Fig. 9 and 10. Mean absolute errors between the estimated field and the actual field for 4 robots with 9 total regions in Fig. 9a and for 8 robots with 25 total regions in Fig. 9c perform comparably despite a nearly 15% reduction in the area being sensed by robots. This can be attributed to the robustness of the constructed model. Despite, the use of various initial POD bases, the distributed optimal placement eventually results in similar errors estimations as shown by Fig. 10a-d. This is again due to the adaptive nature of the algorithm.

In the water tank, the robots were able to track the projected dye. The robots collect measurements from their sensing locations and adapt their assigned models. They are able to switch locations to track the process as shown in Fig. 11.

VI. CONCLUSIONS

In this work, we have proposed a solution for the sampling and tracking of a dynamic process with a team of mobile robots. This approach uses distributed to techniques to allow for modeling and estimation of a field. Unlike other works, this work leverages the rich information from the process dynamics to inform where robots should sense, how they should best model their environment, and how they should adapt their belief about the environment.

For future work, we would like include an analysis on the error bounds of the algorithm. Namely, we hope to establish upper bounds on the errors introduced through reduced order modeling and the distributed computations. Additionally, we would like to investigate the use heterogeneous robots, such as a team of aerial robots and marine robots to produce multi-fidelity models of the environments. Incorporating these multi-fidelity models may allow for the use of complementary information. For example, aerial robots may be able to collect and model less granular information but over wider areas of the field, while marine robots may be able to collect and model higher granularity information but only at specific locations of the field.

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 and Dhanushka Kularatne at the GRASP Laboratory for providing help during experiments.

References

[1] J. Cortés, S. Martínez, T. Karatas, F. Bullo, and S. Member, "Coverage Control for Mobile Sensing Networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[2] J. Le Ny and G. J. Pappas, "Sensor-Based Robot Deployment Algorithms," *Proceedings of the IEEE Conference on Decision and Control*, pp. 5486–5492, 2010.

[3] M. Schwager, M. P. Vitus, S. Powers, D. Rus, and C. J. Tomlin, "Robust Adaptive Coverage Control for Robotic Sensor Networks," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 3, pp. 462–476, 2017.

[4] G. A. Hollinger, A. A. Pereira, J. Binney, T. Somers, and G. S. Sukhatme, "Learning Uncertainty in Ocean Current Predictions for Safe and Reliable Navigation of Underwater Vehicles," *Journal of Field Robotics*, vol. 33, no. 1, pp. 47–66, 2016.

[5] J. Das, F. Py, J. B. Harvey, J. P. Ryan, A. Gellene, R. Graham, D. A. Caron, K. Rajan, and G. S. Sukhatme, "Data-driven robotic sampling for marine ecosystem monitoring," *International Journal of Robotics Research*, vol. 34, no. 12, pp. 1435–1452, 2015.

[6] S. Kemna, J. G. Rogers, C. Nieto-Granda, S. Young, and G. S. Sukhatme, "Multi-robot coordination through dynamic Voronoi partitioning for informative adaptive sampling in communication-constrained environments," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2124–2130, 2017.

[7] S. Garg and N. Ayanian, "Persistent Monitoring of Stochastic Spatio-temporal Phenomena with a Small Team of Robots," *Proceedings of Robotics: Science and Systems*, 2014.

[8] M. R. García, C. Vilas, J. R. Banga, and A. A. Alonso, "Optimal field reconstruction of distributed process systems from partial measurements," *Industrial and Engineering Chemistry Research*, vol. 46, no. 2, pp. 530–539, 2007.

[9] F. Q. Elizabeth, "Stable Arrangements of Mobile Sensors for Sampling Physical Fields," 2012.

[10] K. M. Lynch, I. B. Schwartz, P. Yang, and R. A. Freeman., "Decentralized environmental modeling by mobile sensor networks," *IEEE Transactions on Robotics and Automation*, vol. 24, no. 3, pp. 710–724, 2008.

[11] H. F. Durrant-Whyte, "Sensor Models and Multisensor Integration," *International Journal of Robotics Research*, vol. 7, no. 6, 1988.

[12] L. Sirovich, "Turbulence and the dynamics of coherent structures. I. Coherent Structures," *Quarterly of Applied Mathematics*, vol. 45, no. 3, pp. 561–571, 1987.

[13] M. Kirby, *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns*. New York, NY, USA: John Wiley & Sons, Inc., 2000.

[14] R. Everson and L. Sirovich, "Karhunen–Loève procedure for gappy data," *Journal of the Optical Society of America*, vol. 12, no. 8, p. 1657, 1995.

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

[16] 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 and Chemical Engineering*, vol. 28, no. 1-2, pp. 27–35, 2004.

[17] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins University Press, 1996.

[18] D. Kempe and F. McSherry, "A Decentralized Algorithm for Spectral Analysis," *Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing*, pp. 561–568, 2004.

[19] F. Penna and S. Stanczak, "Decentralized Eigenvalue Algorithms for Distributed Signal Detection in Cognitive Networks," *IEEE Transactions on Signal Processing*, vol. 63, no. 2, pp. 427–440, 2015.