Adaptive Sampling and Energy Efficient Navigation in Time-Varying Flows

Tahiya Salam[1], Dhanushka Kularatne[1], Eric Forgoston[2], and M. Ani Hsieh[1]

[1] University of Pennsylvania, Philadelphia PA 19104, USA,
{tsalam, dkul, m.hsieh}@seas.upenn.edu,
WWW home page: https://scalar.seas.upenn.edu/
[2] Montclair State University, Montclair NJ 07043, USA
eric.forgoston@montclair.edu

Abstract. This work presents a strategy to enable a team of mobile robots to adaptively sample and track a dynamic spatio-temporal process. We propose a distributed strategy, where robots collect sparse sensor measurements, create a reduced-order model of the spatio-temporal process, and use this model to estimate field values for areas without sensor measurements of the dynamic process. The robots then use these estimates of the field, or inferences about the process, to adapt the model and reconfigure their sensing locations. We use this method to obtain an estimate for the underlying flow field and use that to plan optimal energy paths for robots to travel between sensing locations. We show that the errors due to the reduced order modeling scheme are bounded, and we illustrate the application of the proposed solution in simulation and compare it to centralized and global approaches. We then test our approach with physical marine robots sampling a spatially non-uniform time-varying process in a water tank.

Keywords: adaptive sampling, reduced order modeling, path planning, multi-robot systems, distributed robot systems, sensor networks, swarms, marine robotics

1 Introduction

The ocean poses an immensely complex fluid dynamical challenge, involving the interplay of rotation, stratification, complex topography and variable thermal and atmospheric forcing, not to mention thousands of biological, chemical, and physical inputs. It is the "engine" that drives weather-climate systems worldwide [10], inextricably linking human life to the ocean. Recent years have seen an increase in the pursuit of robotics technology to support a wide range of activities in marine and littoral environments. Examples include commerce (*e.g.*, aquaculture), natural resource gathering (*e.g.*, mining and mineral exploration), construction and repair (*e.g.*, pipeline servicing and inspection), scientific exploration (*e.g.*, health monitoring of coral reefs), search and rescue efforts (*e.g.*, aviation and maritime accidents), and military operations (*e.g.*, surveillance and mine sweeping).

In all these applications, robots must have the ability to estimate, predict, and track complex and dynamic spatio-temporal processes (*e.g.*, plankton assemblages [2], harmful algae blooms [67, 4, 7]), and temperature and salinity profiles that vary across both

space and time [41, 66, 61]. In general, teams of mobile robots are particularly well-suited to monitor these processes since they have the ability to span large physical scales and obtain simultaneous, real-time measurements at distinct locations using their on-board sensors. The gathered data can be used to develop a model of the environment/processes being monitored and the model can be continuously used and updated to adapt the robots' sensing locations and improve the team's ability to track and predict the process they are monitoring. Nevertheless, complex spatio-temporal processes are often difficult to model and represent. Even when representations are available, the representations are often high-dimensional and computationally burdensome, making it difficult for resource constrained robots to execute by themselves. Since the processes being tracked often occur in dynamic and uncertain environments, it is critical that robots do not rely on *centralized* strategies to reduce the impact of limited communication and robot failures. As such, a significant challenge is the development of *distributed* modeling and estimation strategies for teams of robots tracking time-varying spatio-temporal processes.

Existing work on multi-robot coordination for environmental monitoring, mapping, and modeling applications generally fall under two categories: coverage control and Gaussian process (GP) based information maximizing exploration/search. The seminal work by Cortes *et al.* on optimal sensor placement relies on a weighting function to account for sensing quality and to ensure coverage of the given spatial domain [6]. While the original work assumes the weighting function must be specified *a priori*, the work has been extended in several ways. The stochastic uncertainty associated with modeling the weighting function was incorporated online to optimize the deployment of the sensors in [39]. A strategy for learning the weighting functions online, rather than specifying them *a priori*, was proposed in [53]. A significant advantage of existing coverage control strategies is their ease for distributed implementation. However, these techniques only account for the sensor limitations and ignore the fundamental physics that governs the dynamics of processes of interest. As such, sensing locations provided by these strategies often fail to capture the relevant features needed to obtain a suitable estimate of the field.

In recent years, GPs have been widely used to model various spatio-temporal processes. In [23], the environment is modeled using GPs and the objective is to learn confidence measures on the uncertainty of the model. The uncertainty is then employed to plan minimum risk trajectories for underwater robots. GPs are used to model the quantities of interest being monitored by the robot where the models are then used in a stochastic optimization strategy to minimize regret when collecting measurements/samples [9]. GPs are used to create a map of the environment in [27]. The map is partitioned so robots can determine nearby locations and select future sampling locations that reduces the entropy in the map. The work presented in [18] adapts the model in real-time based on observations and optimizes next best sensing locations based on the updated model for a small team of mobile robots. However, similar to coverage control strategies, techniques that rely on using GPs to model the environment/process neglect the physical principles that give rise to the dynamics of the fluid flow. Furthermore, existing work only accounts for the spatial variations of the field and neglects the temporal variations of the field. These limitations are partly because GPs are incapable of capturing the

important nonlinearities of the process of interest since they are not suitable for modeling multi-scale processes or processes that are described by functions with varying smoothness.

Similar to [41] and [12], our work focuses on the fusion and control of active sensor networks. We consider the modeling and estimation of spatio-temporal processes and how robots can leverage the process model to adapt their sensing locations. We employ proper orthogonal diagonalization (POD) to obtain a model of the process being tracked and use this model to adapt the team's sensing locations. The problem of identifying optimal sensor placements has been addressed in existing work [17, 38], even in the context of optimal sensor placement for reduced-order modeling using POD techniques [14, 1, 64, 48, 16, 13]. Nevertheless, existing work only addresses the placement of static sensors and does not leverage a robot's mobility to adapt their measurements in response to the changing environmental conditions. These works also assume complete knowledge of the time-series data used to derive the reduced-order model using POD and do not address the assimilation of new data into the existing model.

As robots navigate within the workspace to obtain new measurements and the new data is assimilated into the model, the updated model is then used to improve the selection of next best locations for new measurements. A challenge of operating in a time-varying and stochastic flow environment is that the effects of the surrounding fluid dynamics is tightly coupled with the vehicle dynamics. With enough control authority, these environmental effects can be modeled as external disturbances [8, 65] or as a known external flow field [11] and corrected by the vehicle's navigation controller. However, it is often difficult for a power constrained vehicle to reach the desired goal location using these approaches. Recent work has shown that AUV/ASV motion planning and adaptive sampling strategies are improved by incorporating either historical ocean flow data [58, 60, 59] or multi-layer partial differential equation (PDE) models of the ocean [63, 40, 33, 32]. Nevertheless, accessibility to and the overall quality of the flow data and/or numerical models is highly dependent on how well a given region of interest is instrumented. This is because numerical PDE models are often derived through a combination of theoretical and field observations. Ocean current hindcasts, nowcasts, and forecasts provided by Naval Coastal Ocean Model (NCOM) databases [3] and regional ocean model systems (ROMS) [60] are assimilated from satellite and field observations in conjunction with predictions from numerical PDE models [55, 56]. Despite significant advances in ocean modeling and data assimilation, high fidelity in-situ sensing, and increasing computational power, the predictability of existing models has been limited, particularly at the submesoscale. The thousands of biological, chemical, and physical inputs coupled with limited data and computational power for assimilation, makes it very challenging to reconcile spare, uneven observational data with values predicted by the field models. As such, any adaptive sampling must take into consideration vehicle motion and prediction uncertainties.

In this chapter, we describe our efforts in developing an adaptive sampling framework for teams of energy constrained mobile sensors operating in dynamic and uncertain flows. We present a distributed strategy for mobile robot teams to build a reduced-order model of a spatio-temporal process of interest using sparse sensor measurements.

[3] URL: http://cordc.ucsd.edu/projects/mapping/maps

The model is then used to predict and infer field values in regions of the workspace that are not covered by the team. Using their predictions, the robots can then adapt and reconfigure their sensing locations to maximize the effectiveness of the new observations for improving the process model. Furthermore, we show how the model can be leveraged to compute minimum energy trajectories for power constrained vehicles to enable them to leverage the surrounding flow for navigation to their next sensing locations. 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 properties of the environment using sparse sensing data, assimilate these inferences to update the model, and compute energy optimal paths to new sensing locations. The proposed 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 synthesize a distributed algorithm for the computation of the process model and the estimation of the field values. Different from related works in this domain, we explicitly use the dominant spatial and temporal characteristics of the dynamic process to determine the team's next best sensing locations, compute energy efficient trajectories to reach those destinations, and update the model and predictions. We conclude with a discussion on how recent geophysical fluid dynamics insights are leading to new strategies for distributed modeling, sensing, estimation, and control strategies of autonomous unmanned systems in dynamic and uncertain environments. By exploiting these new geophysical fluid perspectives, it may be possible to develop a large-scale distributed modeling, sensing, and estimation framework where collectives of autonomous unmanned systems have the ability to maintain and update a global description of the flow dynamics in real time.

2 Preliminaries

In this section, we provide some preliminaries and present our assumptions on the vehicle models, the process/flow field, and cost functions employed to compute our energy optimal trajectories for the mobile robots.

2.1 Environment and Flow Model

We consider tracking a dynamic process in a continuous spatial region in a fluidic environment, which we will define as the workspace $\mathscr{W} \in \mathbb{R}^d$ where $d \in \{2,3\}$. $\mathscr{W}$ can be discretized into n spatial points where at each of the points, a measurement, such as concentration, temperature, or current velocity 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. The process P can be a scalar field, *e.g.*, temperature or salinity field, or a vector field, *e.g.*, flow velocity field.

When P is a time-varying flow field, it can be described as $\mathbf{V_f} : \mathscr{W}_T \mapsto \mathbb{R}^d$, where $\mathscr{W}_T = \mathscr{W} \times [\underline{t},\ \bar{t}]$ and $[\underline{t},\ \bar{t}] \subset \mathbb{R}_{\geq 0}$ denotes the time interval under consideration. As such, for $\mathbf{x} \in \mathscr{W}$ and $t \in [\underline{t},\ \bar{t}]$, $\mathbf{V_f}(\mathbf{x},t)$ denotes the flow velocity vector. The coordinates of $\mathbf{x}$ in the inertial frame are denoted x_i, $i = 1,\cdots,d$ and similarly components of $\mathbf{V_f}(\mathbf{x},t)$ are denoted by V_{fi}, $i = 1,\cdots,d$. The speed of the flow is given by

$V_f(\mathbf{x},t) = \|\mathbf{V_f}(\mathbf{x},t)\|$ and the maximum flow speed encountered in the domain is given by $V_{fm} = \max_{\mathbf{x}\in\mathbb{W},\, t\in[\underline{t},\, \bar{t}]} V_f(\mathbf{x},t)$. In this work, we assume that the robots are estimating a reduced-order description of $\mathbf{V_f}$ given by $\hat{\mathbf{V}}_\mathbf{f}$. As such, the path planning problem assumes that the reduced-order description, $\hat{\mathbf{V}}_\mathbf{f}$, is completely known since the robots would simply employ their most up-to-date estimate, $\hat{\mathbf{V}}_\mathbf{f}$, for planning purposes. When forecast uncertainties for $\hat{\mathbf{V}}_\mathbf{f}$ are available, we discuss how to plan trajectories subject to these uncertainties in Section 4. Finally, we assume that the positions of static and dynamic obstacles in the environment are available. This information could be encoded by a function $\mathbb{O} : \mathscr{W}_T \mapsto \{true, false\}$, where $\mathbb{O}(\mathbf{x},t) = true$ indicates an obstacle at $(\mathbf{x},t)$ and vice versa. Such information could be obtained online by equipping the vehicles with suitable sensors, e.g., sonar range finders, lidars, RGBD cameras, etc, and by running a parallel routine that computes the positions and possibly velocities of the obstacles.

2.2 Vehicle Model

Given a team of q robots, we assume each robot is equipped with a sensor that is capable of sensing across each of the s sensing regions such that $q < s$. We assume that the quality and range of the onboard sensors are homogeneous and that the robots can localize and communicate small packets of information, *e.g.*, matrices, with their neighbors. Furthermore, robots can obtain an initial estimate of the dynamic process using either historical data or some forecast model. Since $q < s$, each robot is responsible for monitoring multiple regions. However, since each robot's sensor range is finite, the robot can only obtain measurements for a subset of the regions assigned to it. Thus, we assume each robot will have to rely on a pre-existing model of the process and measurements from other areas to infer the value of the process in the assigned regions not covered by its sensor. In this work, robots are assigned disjoint subsets of the workspace so that each robot only has to maintain a model of the environment for its assigned regions. Each robot is then able to share a compact amount of information about its model with its neighbors and use model information aggregated from its neighbors to determine the next best optimal sensing positions to obtain new data to improve their models. As such, we assume robots communicate via a broadcast network architecture and that the communication network remains connected at all times. Robots can move and create or break their connections with other robots as long as the network remains connected.

For motion planning purposes, we employ a holonomic kinematic model for the autonomous mobile sensor. This assumption is valid if the dimensions of the autonomous marine vehicle (AMV) are small compared with the dimensions of the flow structures, and if the timescales of the dynamics governing the vehicle are much faster than those of the flow field. A typical AMV operating in the ocean satisfies both these criteria. Using this model, the net velocity of the vehicle with respect to the inertial frame is given by

$$\mathbf{V_{net}}(\mathbf{x},t) = \mathbf{V_f}(\mathbf{x},t) + \mathbf{V_{still}}(\mathbf{x},t), \tag{1}$$

where $\mathbf{V_{still}}$ is the velocity of the vehicle with respect to the flow, *i.e.,* $\mathbf{V_{still}}$ is the "thrust" vector of the vehicle. To achieve a given velocity $\mathbf{V_{net}}$, the AMV speed with respect to

the flow needs to be

$$V_{still} = \sqrt{(V_{net} - V_f \cos\theta)^2 + (V_f \sin\theta)^2} \tag{2}$$

where $V_{net} = \|\mathbf{V_{net}}\|$, $V_f = \|\mathbf{V_f}\|$, $V_{still} = \|\mathbf{V_{still}}\|$ and θ is the angle between $\mathbf{V_f}$ and $\mathbf{V_{net}}$. We further assume that that the actuation capability of the vehicle is limited and that its maximum speed is lower than the speed of the surrounding flow *i.e.,* $V_{still}(\mathbf{x},t) \leq V_{max} < V_{fm}$.

2.3 Proper Orthogonal Decomposition

In this work, we are interested in developing reduced-order models for complex spatio-temporal processes. While there are many techniques for extracting the dominant dynamics of these infinite-dimensional fields [3, 52, 50], we focus on modal analysis techniques in the construction of a low-dimensional approximation of the process dynamics. In particular, we use proper orthogonal decomposition (POD) [57, 29] to obtain a representative reduced order model of the flow field.

In POD analysis, m snapshots of the field are collected, either through experimentation or numerical simulations, such that at each time $t = 1,...,m$, $\boldsymbol{z(t)} = [z_1(t),...,z_n(t)]^\top$, where n is the spatial dimension of some discretization of the flow field. A covariance matrix is constructed as

$$\boldsymbol{K} = \frac{1}{m}\sum_{t=1}^{m} \boldsymbol{z(t)z(t)}^\top = \frac{1}{m}\boldsymbol{XX}^\top, \tag{3}$$

where $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ with its columns as $\boldsymbol{z(t)}$. The low-dimensional basis is created by solving the symmetric eigenvalue problem

$$\boldsymbol{K\phi}_i = \lambda_i \boldsymbol{\phi}_i, \tag{4}$$

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

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

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

$$\boldsymbol{z(t)} = \boldsymbol{\Phi c(t)}, \tag{6}$$

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.

2.4 Cost Function

To plan energy efficient navigation strategies in marine environments, we focus on techniques that reduces the overall energy utilized by the vehicle as it navigates through the environment. As such we consider a cost function that represents the total energy consumption of the AMV, which is given by $E_{total} = E_{hotel} + E_{drag}$, where E_{hotel} is the energy required to operate the vehicle's computing and sensor systems independent of propulsion [25], and E_{drag} is the energy expended to overcome drag forces exerted by the fluid. Assuming a constant power usage K_h by the computing and sensor systems gives $E_{hotel} = \int_{t_s}^{t_g} K_h dt$. The drag force F_d encountered by the AMV is given by $F_d(t) = K_d V_{still}^{\alpha-1}(\mathbf{x},t)$ where K_d is the drag coefficient and $\alpha \in \{2,3,...\}$. If $\alpha = 2$ the drag is linear, if $\alpha = 3$ the drag is quadratic, and so on. This leads to $E_{drag} = \int_{t_s}^{t_g} K_d V_{still}^{\alpha}(\mathbf{x},t)dt$. Thus, the total cost of a path is given by

$$C = \int_{t_s}^{t_g} K_h + K_d V_{still}^{\alpha}(\mathbf{x},t)dt. \tag{7}$$

Note that K_h and K_d can also be thought of as weighting parameters between minimum time paths and minimum energy paths. If a minimum time path is required, we could set $K_d = 0$ and proceed, and vice versa. If exact energy minimization is required, actual values for K_h and K_d should be used.

3 Adaptive Sampling and Reduced-Order Modeling

In this section, we describe a distributed strategy to enable a mobile robot team to adaptively sample and track a dynamic spatio-temporal process [51]. The approach enables the team to create a reduced-order model (ROM) of the process using sparse sensor measurements and using the model to determine the next best locations for obtaining new data to improve the model. The strategy is distributed in that each robot develops its own ROM using its sensor data and sensor data provided from their teammates. The ROM is then used to infer measurements in regions where no measurements are available and new measurement locations are then determined based on these inferences. The robots navigate to these new locations, obtain new measurements which are then assimilated into the model, and the process is repeated. In this section, we briefly summarize our strategy by presenting the centralized approach and describing a distributed implementation of the centralized strategy. We briefly discuss simulation and experimental results. The interested reader is refer to [51] for the details. Finally, in order to quantify the uncertainty between the dynamic process of interest and the reduced order model constructed by the robots, an error analysis is presented.

3.1 Optimizing Robot Locations for Field Reconstruction

Given a low-dimensional representation of the subspace on which the data is located, we leverage the properties of the orthogonal bases obtained via POD to compute the optimal set of robot locations in the workspace for the reconstruction of the field using

only sparse data in real-time. 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 $\boldsymbol{y_r(t)}$ for sensing region $r \in S$, where each $\boldsymbol{y_r(t)} \in \mathbb{R}^{n_r \times 1}$ 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 [14, 29, 64], the time-dependent coefficients that minimize the distance between $\boldsymbol{y(t)}$, sensor values, and $\boldsymbol{\hat{y}(t)}$, the projection of sensor values onto the subspace associated with the vectors $\{\boldsymbol{\Phi_r}\}_{r \in S}$ can be found using

$$\begin{aligned} \boldsymbol{\hat{c}(t)} &= \boldsymbol{A}^{-1}\boldsymbol{B} \\ \text{for } \boldsymbol{A} = \sum_{r \in S} \boldsymbol{\Phi_r}^\top \boldsymbol{\Phi_r} &\text{ and } \boldsymbol{B} = \sum_{r \in S} \boldsymbol{\Phi_r}^\top \boldsymbol{y_r(t)}, \end{aligned} \tag{8}$$

where this time-dependent coefficient $\boldsymbol{\hat{c}(t)}$ is then applied to $\boldsymbol{\Phi}$ as in (6) to recover the values of the field for which there are no sensor measurements.

To optimize the reconstruction of the full field using only measurements from the q regions, one must determine the q sensing regions from the set of S possible regions. Note that 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 $\boldsymbol{\hat{c}(t)}$ could be calculated exactly using (6). 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, [1] provides a criteria for selecting the optimal

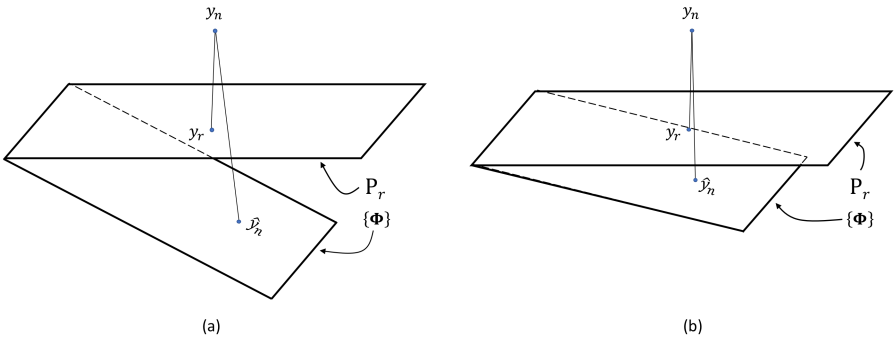


Fig. 1: Geometric interpretation of maximizing the minimum eigenvalue. The data point y_n containing all the measurements of the field is projected as y_r onto the subspace P_r, where y_r equivalently represents a vector of just the sensor measurements, and is projected as $\hat{y}_n$ onto the low-dimensional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

set of sensing regions S as

$$\max_{S} \min_{i} \lambda_i(\boldsymbol{A}), \tag{9}$$

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 [19] 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 (9) is given by

$$\max_{S} \min_{i} a_{ii}, \tag{10}$$

where maximizing the minimum diagonal element of $\boldsymbol{A}$ seeks the set S that results in the basis used to compute $\boldsymbol{A}$ being both close to orthonormal and minimizing the distance between the the subspaces associated with $\boldsymbol{\Phi}$ and $\{\boldsymbol{\Phi_r}\}_{r \in S}$. We employ the algorithm developed in [1] and extended in [17] to find the set S that satisfies criteria (10).

3.2 Adaptive Computation of Reduced Order Model and Robot Locations

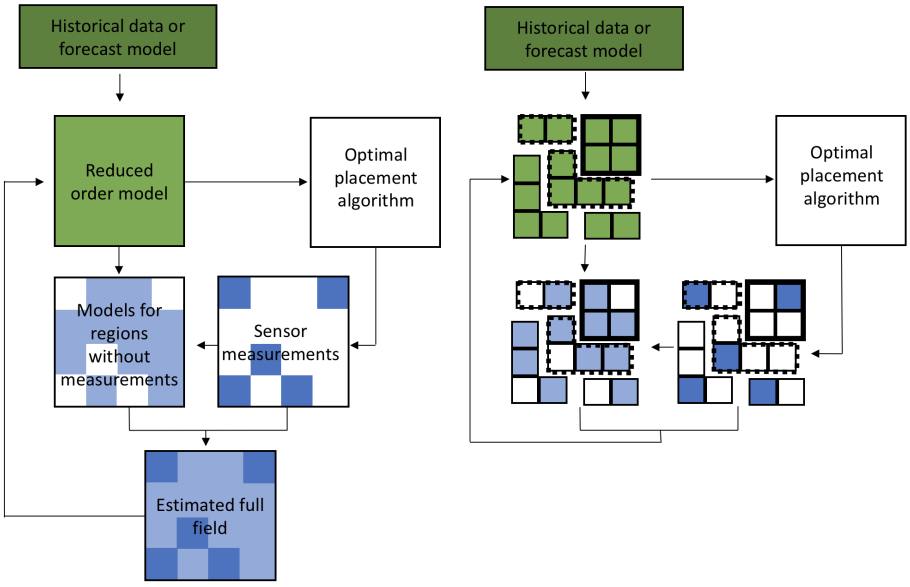
The techniques described in [14, 1, 17, 64] rely on computing POD basis vectors using all the available snapshots of data over the process. Instead, we propose a method that dynamically adapts the POD basis vectors using incoming data and reconfigures the robot positions based on the adapted POD.

At the start, *i.e.*, instance 0, POD basis vectors $\boldsymbol{\Phi(0)}$ are computed using T arbitrarily selected snapshots $\{\boldsymbol{z}(\boldsymbol{t}_{0,1}), ..., \boldsymbol{z}(\boldsymbol{t}_{0,T})\}$ where $\boldsymbol{z}(\boldsymbol{t}_{0,i}) \in \mathbb{R}^{n \times 1}$. The T snapshots are gathered from either experiments or numerical simulation based on the equations governing the process of interest. A set of sensing regions $S(0)$ 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 $r \in S_0$ the collected sensing data, $\{\boldsymbol{\Phi_r(0)}\}_{r \in S_0}$ the POD basis over the sensing regions, and the relationship (8). At instance 1, the new inferences are assimilated into the covariance matrix $\boldsymbol{K}$ as in (3) as new snapshots, at which point the POD basis vectors are recomputed as $\boldsymbol{\Phi(1)}$ and a new set of sensing regions $S(1)$ are found. This procedure is repeated for the duration of the mission. This is contrast to other techniques that compute the POD basis requiring all the initial data and do not update the POD basis after new observations.

3.3 Distributed Implementation

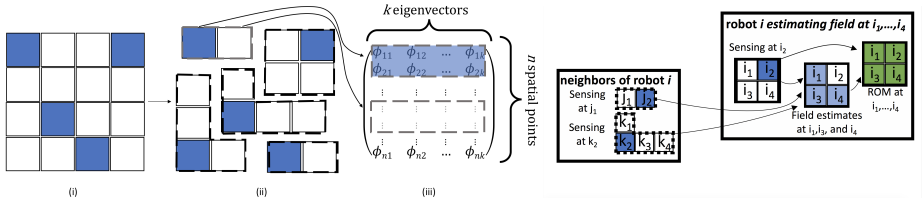
The procedure described above can be implemented in a distributed fashion. In this section, we first show how the procedure can be distributed and conclude with an analysis of the bounds on the error resulting from the ROM.

Methodology A comparison of the centralized and distributed approach is shown in Fig. 2. The model of the environment is represented as a matrix with columns of eigenvectors, where each row of the matrix corresponds to a spatial point. These can be



(a) (b)

Fig. 2: Comparison of a centralized framework for model-inference-assimilation scheme and the corresponding distributed framework. (a) The centralized framework keeps a global model which is combined with sensor measurements to estimate the field and update the model. (b) The distributed framework allows robots to 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 the robots' assigned locations.



(a) (b)

Fig. 3: (a) Visualization of spatial points corresponding to rows of eigenvectors in the POD basis. In (i), 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 (ii), the dashed lines contain the regions for which each robot either takes measurements or estimates values. The matrix in (iii) shows rows in the POD basis that correspond to a single robot's assigned regions indicated with the gray dashed lines. (b) Regions assigned to a robot for sensing, estimating and modeling. The robot senses at a location and is assigned to keep track of models of the 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 values of the field for regions without sensor measurements, all of which will then be used to update its reduced order model.

distributed to different robots, and robots can keep on-board the rows corresponding to their assigned regions as shown in Fig. 3a. The push-sum algorithm [28] 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 areas of the field without sensor measurements and recompute optimal sensing locations, shown in Fig. 3b. Estimating the models in the areas without sensor measurements requires aggregating information from previous models over these areas and the new sensing measurements. 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 estimates of the values and models for each region are only maintained by one robot.

Details of the push-sum algorithm are now described. 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 [28] for more details. Each agent i can compute $\hat{\boldsymbol{P}}^{\boldsymbol{i}}$, its own estimate of $\boldsymbol{P}$, as shown in Algorithm 1.

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

Algorithm 1: Push-sum algorithm

Input : $\boldsymbol{P_i}$ from each robot
Output: robot i's estimate $\hat{\boldsymbol{P}}^{\boldsymbol{i}}$ of matrix $\boldsymbol{P}$
select $\hat{i}$, let $w_{\hat{i}} = 1$, and $w_i = 0\ \forall \hat{i} \neq i$;
for *each robot i in parallel* **do**
  $\hat{\boldsymbol{P}}_{\boldsymbol{i}} = \boldsymbol{P_i}$;
  for *loop* **do**
    $\hat{\boldsymbol{P}}_{\boldsymbol{i}} = \sum_{j \in N_i} m_{ij} \hat{\boldsymbol{P}}_{\boldsymbol{j}}$;
    $w_i = \sum_{j \in N_i} m_{ij} w_j$;
  end
  return $\hat{\boldsymbol{P}}^{\boldsymbol{i}} = \frac{\hat{\boldsymbol{P}}_{\boldsymbol{i}}}{w_i}$
end

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

The distributed computation of Algorithm 2 rests on the following matrix properties, shown in detail in [28]. However, while [28] 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, where robots can be assigned an arbitrary number of rows. 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:

$$\boldsymbol{W} = \boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{R}^\top \boldsymbol{Q}^\top \boldsymbol{Q}\boldsymbol{R}, \tag{11}$$

where $\boldsymbol{Q}^\top \boldsymbol{Q} = \boldsymbol{I}$ since $\boldsymbol{Q}$ is 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 $\boldsymbol{Q_l} = \boldsymbol{V_l}\hat{\boldsymbol{R}}_{\boldsymbol{i}}^{-1}$. Then $\boldsymbol{Q_l}$ is used in the next iteration of the orthonormal iteration algorithm. This requires the entries of 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 [49] 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

$$\begin{aligned} \boldsymbol{v_j} &= \frac{1}{m}\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{q_j} \\ &= \frac{1}{m} diag(\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{q_j}\mathbf{1}^\top) \\ &= \frac{1}{m} diag(\boldsymbol{X}\boldsymbol{X}^\top Diag(\boldsymbol{q_j})\mathbf{1}\mathbf{1}^\top) \\ &= \frac{1}{m} diag[\boldsymbol{X}(\mathbf{1}\mathbf{1}^\top Diag(\boldsymbol{q_j})\boldsymbol{X})^\top]. \end{aligned} \tag{12}$$

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 $\mathbf{1}\mathbf{1}^\top Diag(\boldsymbol{q_j})\boldsymbol{X}$ 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 $\frac{1}{m}\hat{\boldsymbol{F}}^{i\top}\boldsymbol{X_l}$. 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 (8) 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 $\hat{\boldsymbol{c}}_{\boldsymbol{i}} = (\hat{\boldsymbol{A}}^{\boldsymbol{i}})^{-1}\hat{\boldsymbol{B}}^{\boldsymbol{i}}$ and apply coefficients $\hat{\boldsymbol{c}}_{\boldsymbol{i}}$ to the rows $\boldsymbol{Q_l}$ to estimate the values $\hat{\boldsymbol{y}}_{\boldsymbol{l}} = \boldsymbol{Q_l}\hat{\boldsymbol{c}}_{\boldsymbol{i}}$ in the regions $l \in L_i$ for which there are no sensor measurements. We note that in this framework, data broadcasted by each robot consists of matrices of fixed size corresponding to low-dimensional representations of the evolving process of interest.

Error Analysis We begin with a statement of the main result.

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**
      $Z_{\boldsymbol{i}} = \sum_{l \in L_i} q_{lr} X_{\boldsymbol{l}}$;
      Compute $\hat{\boldsymbol{Z}}_{\boldsymbol{i}}$ with Algorithm 1 (push-sum);
      for $l \in L_i$ **do**
        $v_{lr} = \frac{1}{n} \hat{\boldsymbol{Z}}_{\boldsymbol{i}}^{\top} X_{\boldsymbol{l}}$;
      end
      $\boldsymbol{W}_{\boldsymbol{i}} = \sum_{l \in L_i} \boldsymbol{V}_{\boldsymbol{l}}^{\top} \boldsymbol{V}_{\boldsymbol{l}}$;
      Compute $\hat{\boldsymbol{W}}_{\boldsymbol{i}}$ with Algorithm 1 (push-sum);
      Use Cholesky factorization $\hat{\boldsymbol{W}}^{\boldsymbol{i}} = \hat{\boldsymbol{R}}_{\boldsymbol{i}}^{\top} \hat{\boldsymbol{R}}_{\boldsymbol{i}}$;
      $\boldsymbol{Q}_{\boldsymbol{l}} = \boldsymbol{V}_{\boldsymbol{l}} \hat{\boldsymbol{R}}_{\boldsymbol{i}}^{-1}$;
    end
  end
end
return rows $\boldsymbol{Q}_{\boldsymbol{i}}$ as eigenvectors for robot i;
return diagonals of $\hat{\boldsymbol{R}}_{\boldsymbol{i}}$ as eigenvalues for robot i;

Theorem 1. *For sufficient mixing time τ, as determined by the criteria for mixing speeds of Markov Chains [28], $\delta = t_{i+1} - t_i$ the time between collected snapshots, $\left\| \frac{d\boldsymbol{f}(t)}{dt} \right\| \leq \xi$ for $\boldsymbol{f}$ describing the dynamic process, the bound between $(\boldsymbol{P} + \boldsymbol{P}^{\perp})$, the projection corresponding to the dynamic process and $\boldsymbol{P}_{\hat{\boldsymbol{Q}}_{\boldsymbol{i}}}$, the projection onto the POD basis vectors estimated by robots with high probability is defined as:*

$$\left\| (\boldsymbol{P} + \boldsymbol{P}^{\perp}) - \boldsymbol{P}_{\hat{\boldsymbol{Q}}_{\boldsymbol{i}}} \right\|_2 \leq e^{\gamma\tau}(2\lambda_{k+1} + \frac{\delta^2}{8}\xi) + \mathscr{O}\left| \frac{\lambda_{k+1}}{\lambda_k} \right|^{\tau} * n + (C+2)\varepsilon^{4\tau} \tag{13}$$

The proof consists of two main components. First, the error between a dynamic process and its centralized POD basis vector estimation is computed. This explicitly takes into account the time between snapshots. Second, the error between the centralized POD basis vector estimation and the decentralized POD basis vector estimation is computed, taking into account the errors incurred by individual robots through the push-sum algorithm. The results are aggregated to derive a final estimation on the error between the dynamic process and the estimated POD basis vectors computed by the robots. The derivation of the bound can be found in the Appendix.

3.4 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 their neighbors so each robot can compute the optimal sensing locations.

After finding the set of optimal sensing locations, robots are assigned to locations that minimize the total cumulative energy cost incurred by all robots. In the next section, we present how the path costs and the corresponding optimal paths are computed for a given robot and its list of destination points. Once the energy cost matrix between all the robots and the next set of sensing locations is computed, we perform a task assignment using the Hungarian algorithm to reduce the overall energy cost. The flow velocity description $\mathbf{V_f}$, required for path planning could be obtained using the same reduced order modelling framework described above. In this case, since the process being observed, *i.e.,* flow velocities, are d-dimensional vectors with $d \in \{2,3\}$, the measurement vector $\mathbf{z}(\mathbf{t})$ would have $d \times n$ entries instead of the n entries for the scalar case.

4 Planning in Flows

In this section a graph based solution method to compute optimal paths in time-varying flows is presented. The method allows for searching across different actuation speeds at each node in order to compute the globally optimum trajectory. It accounts for the time-varying nature of the environment and also considers an adaptive discretization resolution scheme for graph construction. In the context of the previous section, this general method could be used to compute the optimal path cost from the current agent position to the next sensing location. The graph based nature of the approach makes it possible to compute the optimal energy cost from the current position to each of the next candidate sensing locations in one go.

All the graph based path planning strategies mentioned in section 1 are based on a fixed discretization scheme, where the environment is discretized uniformly. The implicit assumption in any graph based method is that the properties of the system (thrust, flow velocity, etc.) remain constant along edges of the graph. However, in reality, these system properties will vary along edges. Thus, this assumption will give rise to errors in the state dependent costs (e.g., energy cost) computed for the edges. These errors would be more pronounced if the discretization is coarse, and would lead to incorrect results. It is possible to overcome this by allowing system properties to change along edges. However, such a solution is computationally infeasible since the edge cost computations would now require the solution to a two point boundary value problem at each edge. Alternatively, the discretization resolution could be made very fine, which would again lead to a computationally intractable problem.

In the approach presented in this section, these errors due to discretization resolution are overcome with a novel adaptive discretization scheme. Similar to existing graph based methods, the environment is represented as a discrete graph and a search algorithm is used to find an optimal path in the graph. The fundamental difference of the presented method lies in the adaptive discretization scheme used in the construction of the graph. In this method, the discretization resolution is selected locally according to the flow characteristics at each point. Specifically, the discretization resolution is selected such that the flow velocity error remains bounded along an edge. Thus, if the flow changes rapidly at a point, the resolution is made finer, and if the flow remains relatively constant, the discretization resolution is made coarser. The result is the first

graph based method that uses adaptive discretization to compute optimal paths in time-varying flows.

We use a discrete graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ to represent the workspace $\mathscr{W}_T$, where $\mathscr{V}$ is the vertex set and $\mathscr{E}$ is the edge set. Each $v_i \in \mathscr{V}$ represents a point in $\mathscr{W}_T$ and is identified by the pair $(\mathbf{x_i}, t_i)$. Each $e_{ij} \in \mathscr{E}$ represents a directed edge from v_i to v_j and has an associated traversal cost computed using (7). In computing the the edge cost, we assume that the flow velocity along the edge remains constant at $\mathbf{V_f}(v_i)$, the flow at the base of the edge. Note that with a little abuse of notation, we sometimes represent the flow velocity at a vertex v_i with $\mathbf{V_f}(v_i)$.

4.1 The need for adaptive discretization

To illustrate the need for adaptive discretization in graph based methods for planning optimal paths in time-varying flows, consider three cases of the wind-driven double gyre flow model which is often used to describe large scale recirculation regions in the ocean [62]. The model is given by

$$V_{f1}(\mathbf{x},t) = -\pi A \sin\big(\pi f(x_1,t)\big)\cos(\pi x_2) - \mu x_1 \tag{14a}$$

$$V_{f2}(\mathbf{x},t) = \pi A \cos\big(\pi f(x_1,t)\big)\sin(\pi x_2)\frac{\partial f(x_1,t)}{\partial x_1} - \mu x_2 \tag{14b}$$

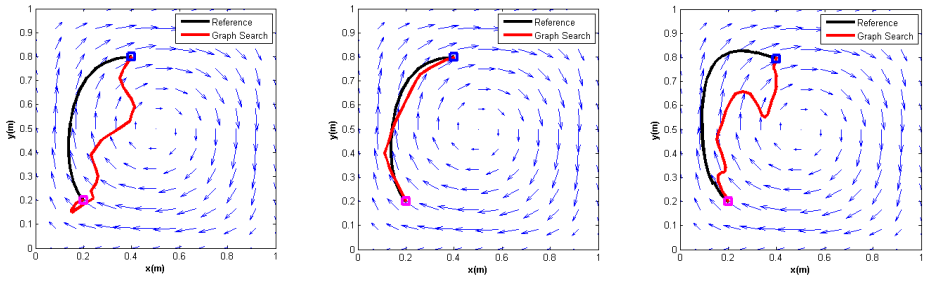
$$f(x,t) = \varepsilon \sin(\omega t)x^2 + \big(1 - 2\varepsilon \sin(\omega t)\big)x \tag{14c}$$

where A determines the maximum speed of the flow field, μ is a dissipation constant, and, ε and ω respectively determine the amplitude and the frequency of the time-varying oscillations of the flow field.

In the three cases considered, the fixed resolution multi time-step search method described in [34] is used to compute optimal energy paths. In [34], the spatio-temporal workspace is uniformly discretized using fixed intervals Δx and Δt. In the first two cases, the flow field parameters are selected to be $A = 1$, $\omega = 4\pi$ and $\varepsilon = 0.1$. For the third case, a value of $\varepsilon = 0.6$ is selected while leaving the rest of the parameters unchanged. In all three cases $\mu = 0$. This results in a maximum flow speed of $V_{fm} = 3.74$m/s and $V_{fm} = 6.73$m/s for the first two cases and the third case respectively. However, in both cases the average flow speed is ≈ 2m/s. Lastly, in all three cases, the maximum vehicle speed is set as $V_m = 2$m/s.

In case 1, a discretization resolution of $\Delta x = 0.01m$, and $\Delta t = 0.1s$ was used for the graph search and the results are shown in red in Fig. 4a. It can be clearly seen that the resulting path does not match the expected path shown in black. One reason for this discrepancy is due to Δt being set too high with respect to Δx, resulting in an overly slow edge traversal speed (≈ 0.1m/s). For case 2, Δt was set to $0.01s$ so as to increase the edge traversal speed. The results are shown in Fig.4b which is much closer to the optimal path. Thus, it is clear that the size of Δx and Δt has to be selected according to the local flow speeds to yield better results. Case 3 further supports this assertion. While the larger ε value in Case 3 (0.1 vs. 0.6) does not affect the average flow speed, it results larger spatio-temporal variations within each region of the flow field. Since the encountered flow speeds are of similar magnitudes for both cases 2 and 3, one would expect similar results if the same discretization resolutions were used. However, Fig.

4c shows that the resulting path computed for case 3 does not match the expected path. This mismatch is a result of the large spatio-temporal variations of the flow within the volume of space-time centered at each vertex, which violates the assumption that the flow velocity remains constant along an edge of the graph. As such, appropriate selection of Δx and Δt that ensures that the local flow remains relatively constant, is crucial for obtaining good results. This is especially true in highly turbulent regions of the flow field where the discretization resolution has to be made finer to account for the larger changes over the same space-time volume.



(a) Case 1 (b) Case 2 (c) Case 3

Fig. 4: Expected optimal path (in black) vs computed optimal path (in red). Case1: $\Delta x = 0.01m$ and $\Delta t = 0.1s$ results in erroneous results as the selected discretization does not match the flow speeds encountered. Case 2: $\Delta x = 0.01m$ and $\Delta t = 0.01s$ results in a better result. Case 3: The flow is changed by setting $\varepsilon = 0.6$, with the same discretization as case 2. However, the result is incorrect because the discretization does not consider the spatio-temporal variation of the flow.

4.2 Adaptive discretization method

The implicit assumption in any graph based method for computing optimal paths in time-varying flows is that the flow velocity remains constant along the edges of a graph. Thus, in order to overcome the issues highlighted in the example above, the discretization resolution should be selected to be small enough so that the actual flow velocity variation along an edge in the graph is small.

Consider an edge $e_{ij} = (v_i v_j)$ from node $v_i = (\mathbf{x_i}, t_i)$ to node $v_j = (\mathbf{x_j}, t_j)$ with $\mathbf{x_j} = \mathbf{x_i} + \mathbf{dx}$ and $t_j = t_i + dt$. It is assumed that the flow velocity at v_j can be approximated by a first order Taylor series expansion at v_i given by

$$\mathbf{V_f}(v_j) = \mathbf{V_f}(v_i) + \tilde{\nabla}\mathbf{V_f}(v_i)\mathbf{dx_t} \tag{15}$$

where $\tilde{\nabla}\mathbf{V_f}(v_i)$ is the gradient of the flow vector at v_i, and $\mathbf{dx_t} = [\mathbf{dx}^T, dt]^T$. Since the flow is assumed to be constant along an edge, the error in the velocity along an edge is given by

$$\mathbf{V_e} = \mathbf{V_f}(v_j) - \mathbf{V_f}(v_i) = \tilde{\nabla}\mathbf{V_f}(v_i)\mathbf{dx_t}. \tag{16}$$

Thus, the magnitude of the velocity error due to a displacement $\mathbf{dx_t}$ is

$$\|\mathbf{V_e}\| = \sqrt{\mathbf{V_e}^T\mathbf{V_e}} = \sqrt{\mathbf{dx_t}^T\mathbf{H}(v_i)\mathbf{dx_t}},$$

where $\mathbf{H}(v_i) = \tilde{\nabla}\mathbf{V_f}^T(v_i)\tilde{\nabla}\mathbf{V_f}(v_i)$ is a symmetric matrix. As such, it can be shown that

$$\|\mathbf{V_e}\| = \sqrt{\mathbf{dx_t}^T\mathbf{H}(v_i)\mathbf{dx_t}} \leq \lambda_{max}(v_i)\|\mathbf{dx_t}\|, \tag{17}$$

where $\lambda_{max} > 0$ is the maximum eigenvalue of $\mathbf{H}(v_i)$. This maximum error occurs when $\mathbf{dx_t}$ is parallel to $\mathbf{v_{max}}$ which is the direction of the eigenvector corresponding to λ_{max} (see Fig. 5a). Thus, to limit the flow velocity error along the edge, we require

$$\|\mathbf{V_e}\| \leq \lambda_{max}(v_i)\|\mathbf{dx_t}\| \leq p\|\mathbf{V_f}(v_i)\| \tag{18}$$

where $0 < p < 1$ is a ratio that specifies the magnitude of the allowable error in terms of the flow at the base of the edge. To ensure (18) is satisfied by any edge emanating from v_i, we require

$$\|\mathbf{dx_t}\| \leq dx_{t_{max}} = \frac{p\|\mathbf{V_f}(v_i)\|}{\lambda_{max}(v_i)}. \tag{19}$$

Since the edge length limit $dx_{t_{max}}$ in (19) is computed using a first order approximation of the error, it will not be correct for large $\|d\mathbf{x_t}\|$. To address this, Newton's root finding method is used along the $\mathbf{v_{max}}$ direction, with $v_{t_1} = v_i + dx_{t_{max}}\mathbf{v_{max}}$ as the initial point, to find the point $v_p = (\mathbf{x_p}, t_p)$ along $\mathbf{v_{max}}$, that has a velocity error very close to $p\|\mathbf{V_f}(v_i)\|$ (see Fig. 5b). The maximum allowable spatial and temporal displacements at node v_i, dx_{max} and dt_{max} are then respectively selected as

$$dx_{max} = \|\mathbf{x_p} - \mathbf{x_i}\|, \tag{20}$$

$$dt_{max} = |t_p - t_i|. \tag{21}$$

4.3 Adaptive Single Time-step Search (aSTS) method

In case 1 of the illustrative example, we showed that the spatial and temporal discretization given by Δx and Δt need to be selected according to the speed of the flow. If Δx and Δt were chosen such that the resulting edge traversal speed ($\approx \Delta x/\Delta t$) is either too fast or too slow, the results would be incorrect. Here, we present the adaptive Single Time-step Search (aSTS) method which explicitly considers the local flow speeds, in addition to the limits established in (20) when selecting the discretization resolution. Our method ensures that the flow velocities remain relatively constant along edges, and that the resulting edge traversal speeds are commensurate with the underlying flow speeds.

During graph construction, the aSTS method considers the region of space that could be reached from a given node v_i in a single time step Δt, under the influence of both the vehicle actuation and the flow velocity at v_i. This reachable space is demarcated by an n-ball of radius $V_{max}\Delta t$ centered at $\bar{\mathbf{x}} = \mathbf{x_i} + \mathbf{V_f}(v_i)\Delta t$. In 2-D, this reachable

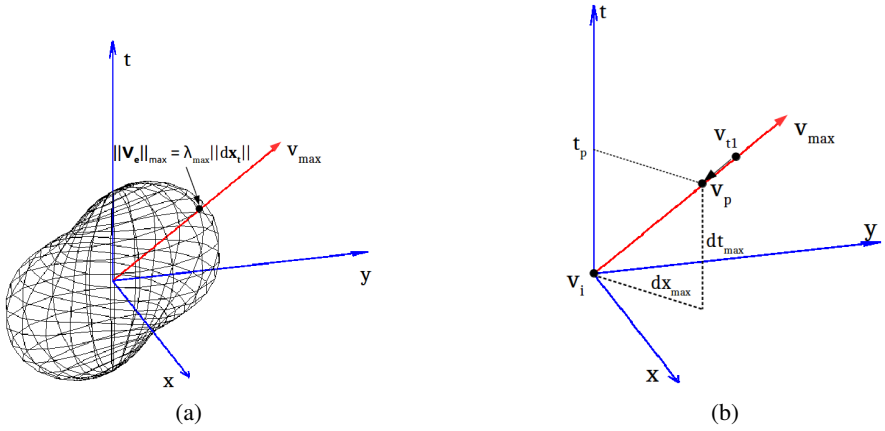
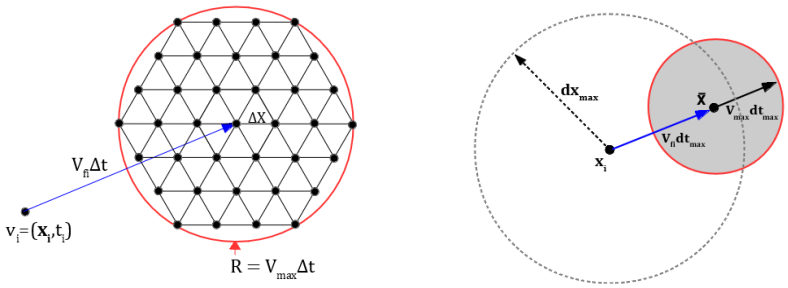


Fig. 5: (a) Velocity error magnitude surface at v_i. The distance of each point on the surface from v_i represents the magnitude of the error in that direction. The maximum velocity error occurs along $\mathbf{v_{max}}$. (b) Newton's root finding method is used along the $\mathbf{v_{max}}$ direction to find the point v_p that has a velocity error of exactly $p\|\mathbf{V_f}(v_i)\|$.

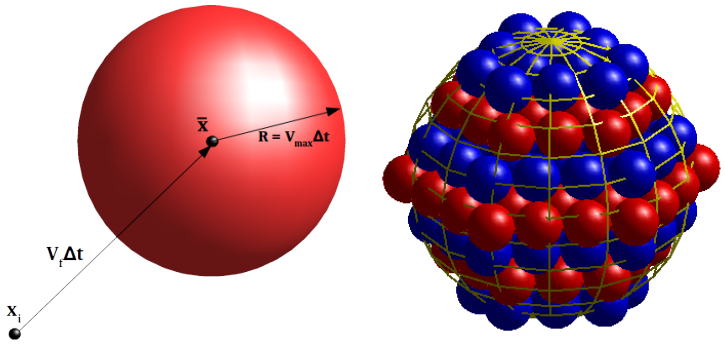
space is a circle and is discretized using a hexagonal lattice of vertices, centered at $\overline{\mathbf{x}}$ with $2q+1$ vertices along the main axis ($3q^2+3q+1$ nodes in total, see Fig. 6a). In 3-D, the reachable space is a sphere (see Fig. 7a). In order to discretize this space, results for the 'sphere packing problem' are used. It has been shown that, densest packing of spheres in three dimensions can be obtained using either the *face centered cubic (FCC)* method or the *hexagonal close packed (HCP)* method [5], and in the current application, the HCP method is used. In the HCP method, the base layer (A) consists of a tightly packed planar layer of spheres. The second layer (B) is constructed on top of the A layer, by placing spheres in the hollows created by adjacent spheres in the A layer. HCP packing is achieved by stacking A and B layers alternatively. Fig. 7b shows the reachable space discretized using the HCP method. The center of each sphere is considered to be a node location is the graph constructed for the aSTS method. The number of nodes in the lattice is decided by the number of spheres used to discretize the horizontal base layer through $\overline{\mathbf{x}}$. In the instance shown in Fig. 7b, the base layer has 37 spheres (this is equivalent to using a grid with $q=3$ to discretize the reachable space in the 2D case), which results in 95 nodes in total. The vertices in the reachable space are added to the neighbor set $\mathcal{N}(v_i)$ of v_i. For each $v_j \in \mathcal{N}(v_i)$, an edge $e_{ij} = (v_i, v_j)$ and a vertex v_j is added to the graph if $\mathbb{O}(v_j) = false$, *i.e.,* if the vertex is not obstructed by an obstacle. All the vertices in $\mathcal{N}(v_i)$ will have the same time coordinate $t_j = t_i + \Delta t$.

The time step Δt at each v_i is selected according to the conditions given in (20). The maximum spatial distance between v_i and any $v_j \in \mathcal{N}(v_i)$ is $(V_f + V_{max})\Delta t$ (at the farthest point on the reachable space). From the discussion in section 4.2, we want $\Delta t \leq dt_{max}$ and $(V_f + V_{max})\Delta t \leq dx_{max}$. Thus, for each v_i, we select Δt according to the



(a) (b)

Fig. 6: (a) The reachable space from v_i is a circle of radius $V_{max}\Delta t$ centered at $\bar{\mathbf{x}}$. A hexagonal lattice of vertices is used to represent this space in the graph. In this case $q = 3$. (b) Since, in this case, $(V_f + V_{max})dt_{max} > dx_{max}$, Δt should be selected as $\Delta t = \frac{dx_{max}}{V_f + V_{max}}$ according to (22).



(a) (b)

Fig. 7: (a) The reachable space in 3D is demarcated by a sphere of radius $V_{max}\Delta t$ centered at $\overline{\mathbf{x_i}} = \mathbf{x_i} + \mathbf{V_f}\Delta t$. (b) The HCP method used to discretize the reachable space with spheres. The A layers are in red while the B layers are in blue. For the case shown, there are 95 nodes in total.

following rule

$$\Delta t = \begin{cases} dt_{max}, & (V_f + V_{max})dt_{max} \leq dx_{max} \\ \frac{dx_{max}}{V_f + V_{max}}, & otherwise \end{cases}. \tag{22}$$

Selection of Δt according to (22) ensures that the edges added to the graph at each node expansion satisfy the conditions specified in (20) (see Fig. 6b). The graph is constructed by repeating this process at each node expansion (see Fig. 8), and the node expansion is guided by the A* algorithm. The aSTS method guarantees that,

1) the maximum velocity error along any edge is less than $pV_f(v_i)$,
2) the net vehicle speed along any edge is commensurate with the flow velocity at the base of the edge.

Note that the only user selected parameters in this method are the edge velocity variation limit p and the number of vertices used to discretize the reachable space. All discretization levels are computed automatically based on these values during runtime. More information about this method can be found in [35]. In conventional cases, where there is only one goal location, the algorithm is terminated when the goal has been reached. However, in cases where optimal paths and path costs are required to multiple goal locations, as in the case of the task allocation phase of the reduced order modelling framework, the graph search could be continued until all goal locations have been reached.

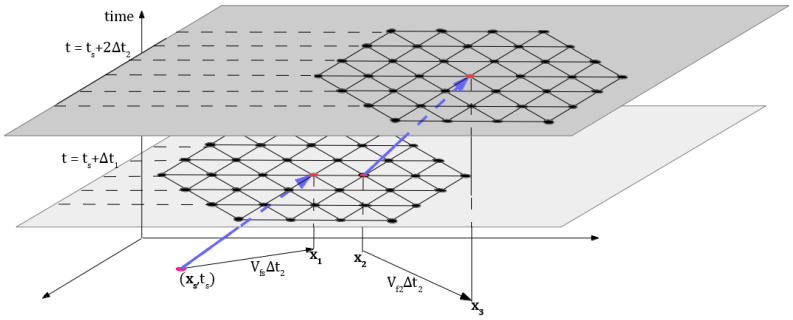


Fig. 8: Construction of the graph for 2-D environments using the aSTS method. All the vertices reachable within a single time step from the base node are considered as neighbors.

4.4 Planning with forecast uncertainties

The reduced order modelling framework described in Section 3 makes it is possible to obtain uncertainty estimates for the flow velocities provided by the ROM. When uncertainty estimates are available, the aSTS method could be used to compute minimum *expected* cost paths. In such cases, the edge cost function used in the aSTS method needs to be modified to be

$$dC_{ij} = \left(K_h + K_d E[\tilde{V}^{\alpha}_{still_j}]\right) dT$$

where

$$E[\tilde{V}^{\alpha}_{still_j}] = \frac{1}{2\pi\sigma_{f1}\cdots\sigma_{fd}} \int\cdots\int \Big(\sum_{k=1}^{d} \hat{x}_k^2\Big)^{\alpha/2} e^{-\left[\sum_{k=1}^{n} \frac{(\hat{x}_k - V_{still_{jk}})^2}{2\sigma_{fk}^2}\right]} d\hat{x}_1\cdots d\hat{x}_n. \quad (23)$$

In (23), $V_{still_{jk}}$ is the k^{th} component of the thrust vector ($\mathbf{V_{still_j}}$) required to reach node v_j from v_i in a deterministic flow, and σ_{fk} is the standard deviation of the flow velocity estimate in the k^{th} direction. More information about this derivation and the associated assumptions can be found in [37].

5 Simulations and Experiments

In this section we present simulation and experimental validation for the methodology presented in the previous sections. We first present a systematic validation of the POD based reduced order method presented in Section 3, and then present simulation based validation for the path planning scheme used to navigate the robots from the current sensing location to the next.

5.1 Results for the POD based reduced order modeling method

Analyses were carried out both in simulation and on physical robots. In simulation, a $1\ m \times 1\ m$ two-dimensional grid space was modeled using video data from an experimental flow tank at low Reynolds number. The fluid experiment was conducted in a $10\ cm \times 10\ cm$ tank with glycerol at a depth of $1-2\ cm$. The tank is equipped with a 4×4 array of equally spaced submerged disks. The mechanism creates a cellular flow where two sets of 8 disks are separately controlled via independent stepper motors and speed controllers. As such, the resulting flow has both a spatially non-uniform and a temporally complex pattern. The dye was strategically placed at the start of the experiment such that the resulting unsteady flow would stretch the dye along dynamically distinct regions in the flow field. 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.

In simulation, the grid space was discretized into 9 non-overlapping regions. 4 robots were simulated in the field. Concentration values of the field were initially gathered for 100 equally spaced times across the time series, and Gaussian noise was then added to these concentration values. These were then used to construct the initial POD basis for the distributed placement algorithm. Data was collected for another 100 sequential times before adapting the POD basis and recomputing the placement. The distributed placement algorithm was compared to the centralized placement algorithm, where all computations occur on a centralized system and are broadcasted to robots. Additionally, the distributed placement algorithm was compared with radial basis function (RBF) interpolation schemes. Two RBF interpolations were computed using the real data: 1) from the regions determined as sensing locations from the distributed method and 2) from randomly selected points across the entire field. All of these methods were compared against the optimal placement that was calculated using noiseless

data across the entirety of the time series. Simulations were carried out for various number of robots, discretizations of the spatial domain, and number of snapshots for the initial construction of the POD basis.

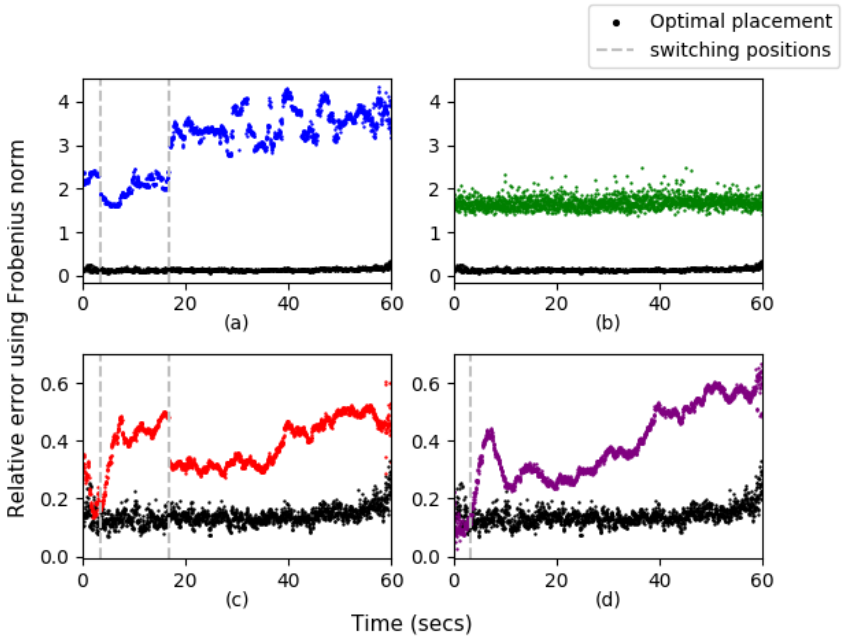


Fig. 9: Norm-wise relative error between field simulation and various field estimation algorithms at each time step. Estimations are calculated using (a) RBF using sensing data, (b) RBF using random points, (c) proposed distributed algorithm, (d) centralized version of proposed algorithm. Black circles represent estimations calculated using the optimal placement determined by using all data to calculate the POD basis. Gray vertical lines indicate robots switching their placement.

The simulation results of the comparison of the various field estimation schemes over the entire time series are shown in Fig. 9. The Frobenius norm-wise error between the actual concentration value and the estimated field computed using various algorithms was computed for each time step. The Frobenius norm-wise error, e, is calculated for field estimate $\hat{\boldsymbol{z}}(\boldsymbol{t})$ using $e = ||\hat{\boldsymbol{z}}(\boldsymbol{t}) - \boldsymbol{z}(\boldsymbol{t})||_F / ||\boldsymbol{z}(\boldsymbol{t})||_F$, where $||\boldsymbol{z}(\boldsymbol{t})||_F = \sqrt{\sum_i |z_i(t)|^2}$. Both RBF interpolation schemes perform significantly worse than placement algorithms using POD. Even in the case of the RBF with randomly selected sensor points, the field estimation is approximately an order of magnitude worse than the three placement algorithms using POD. However, the distributed algorithm and centralized algorithm perform just slightly worse than the optimal placement determined by using all data to calculate the POD basis.

The mean absolute error of the various field estimation schemes is shown in Fig. 10. The RBF interpolation scheme using the data from the sensor measurements results

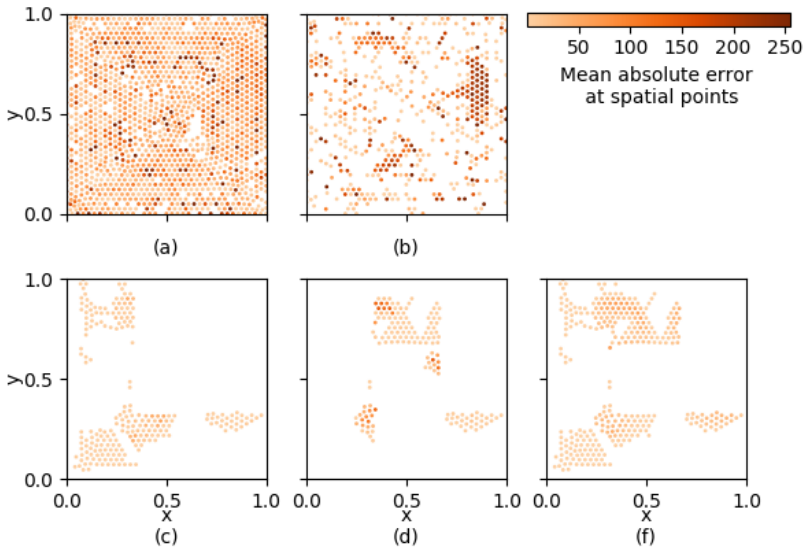


Fig. 10: Mean absolute error at spatial points calculated over time series for various field estimation algorithms. Concentrations at points are calculated using (a) RBF using sensing data, (b) RBF using random points, (c) optimal placement determined by using all data to calculate the POD basis, (d) centralized version of proposed algorithm, and (f) proposed distributed algorithm.

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.

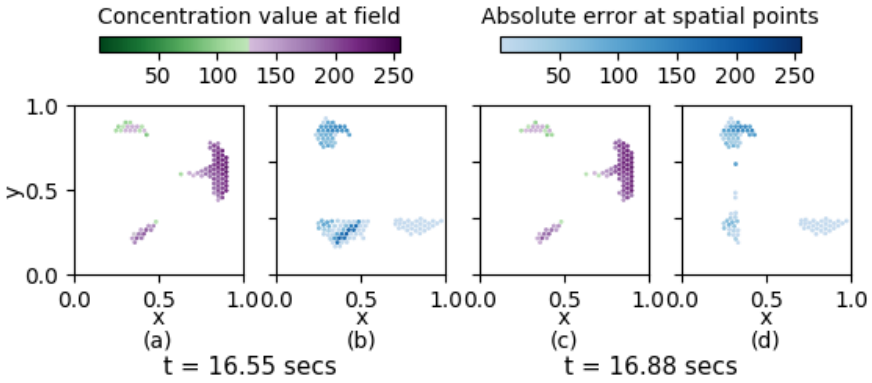


Fig. 11: Concentration field and absolute error at spatial points before and after robots switch locations using distributed placement algorithm. Concentration field (a) and absolute errors (b) are before the switch; concentration field (c) and absolute errors (d) are after assimilating data and switching positions.

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. 11 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. 12 and 13. Mean absolute errors between the estimated field and the actual field for 4 robots with 9 total regions in Fig. 12a and for 8 robots with 25 total regions in Fig. 12c 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. 13a-d. This is again due to the adaptive nature of the algorithm.

Experiments were carried out in a 5 $m \times 3$ m water tank using 4 marine robots, shown in Fig. 14a. The concentration field was mapped and projected onto the tank using the video from the LoRe tank, shown in Fig. 14b. The robots then tracked the pro-

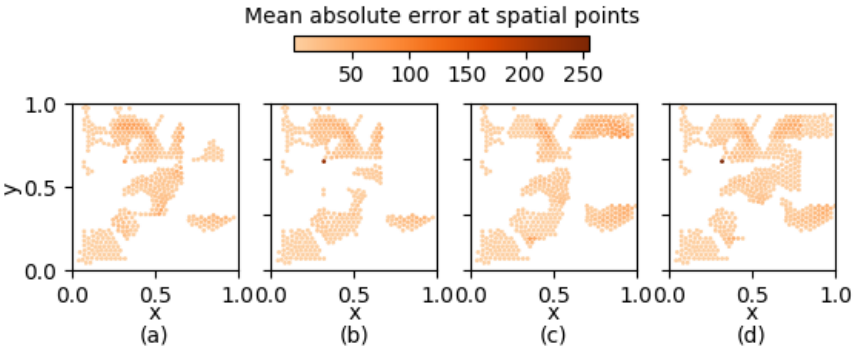


Fig. 12: Mean absolute error at spatial points calculated over time series between field and distributed algorithm for various discretizations of field, numbers of robots, and snapshots used to compute initial POD basis. Algorithm tested for (a) 4 robots, 9 regions, and 100 snapshots, for (b) 4 robots, 9 regions, and 500 snapshots, for (c) 8 robots, 25 regions, and 100 snapshots, and for (d) 8 robots, 25 regions, and 500 snapshots.

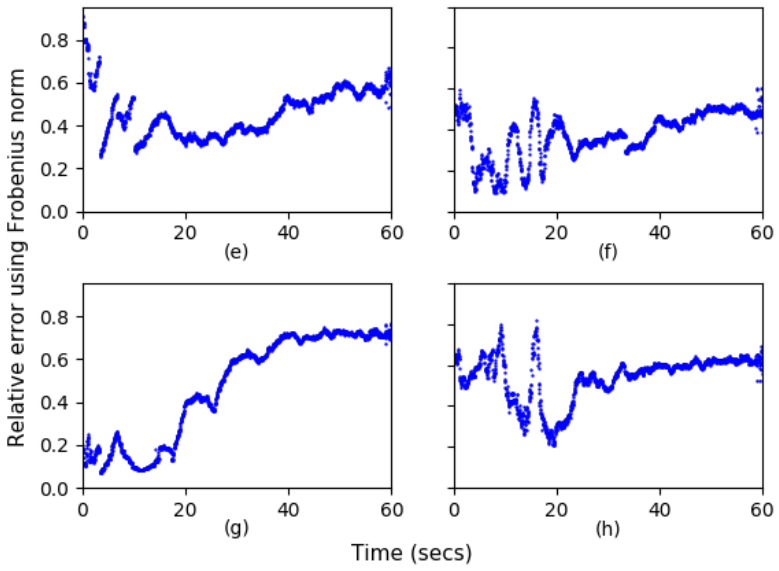
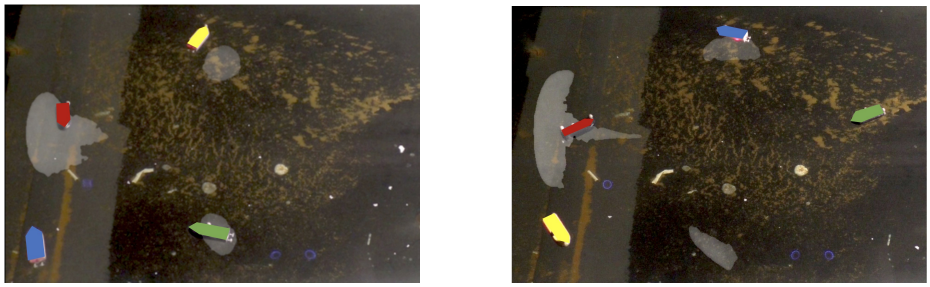


Fig. 13: Norm-wise relative error between field and distributed algorithm for various discretizations of field, numbers of robots, and snapshots used to compute original POD basis. Algorithm tested for (a) 4 robots, 9 regions, and 100 snapshots, for (b) 4 robots, 9 regions, and 500 snapshots, for (c) 8 robots, 25 regions, and 100 snapshots, and for (d) 8 robots, 25 regions, and 500 snapshots.

jected concentration field using the distributed 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. 15.

(a) (b)

Fig. 14: Experimental setup with marine robots. (a) Robot boat equipped with pose information from motion capture system and ability to communicate. (b) Water tank with projection of dynamic process depicted in white and circled in red.



(a) (b)

Fig. 15: Robotic boats tracking dynamic process in water tank. The dynamic process is shown in white, and robots in blue, red, yellow, and green. (a) Robots assume positions based on the initial POD basis. (b) Robots switch positions after collecting sensor measurements and updating their models.

5.2 Simulation Results for aSTS method

In this section the performance of the aSTS method used to compute optimal paths for the sensing agents to move between sensing locations, is verified in simulations. In all simulations, the path parameters were set as $K_h = 0.0005$, $K_d = 1$ and $\alpha = 2$ (linear drag model).

The accuracy of the paths computed by the aSTS method was evaluated by comparing it against a path obtained by solving the corresponding optimal control problem. The optimal control problem involves minimizing the path cost given in (7), subject to the kinematic model in (1), and the constraint $V_{still} \leq V_{max}$. Let $\Gamma^* : [t_s,\ t_g] \mapsto \mathscr{W}$ be the reference path obtained from the optimal control formulation of the problem, and let $\Gamma : [t_s,\ t_g] \mapsto \mathscr{W}$ be the path computed by the proposed method. The mean error (mE) between Γ^* and Γ, defined by

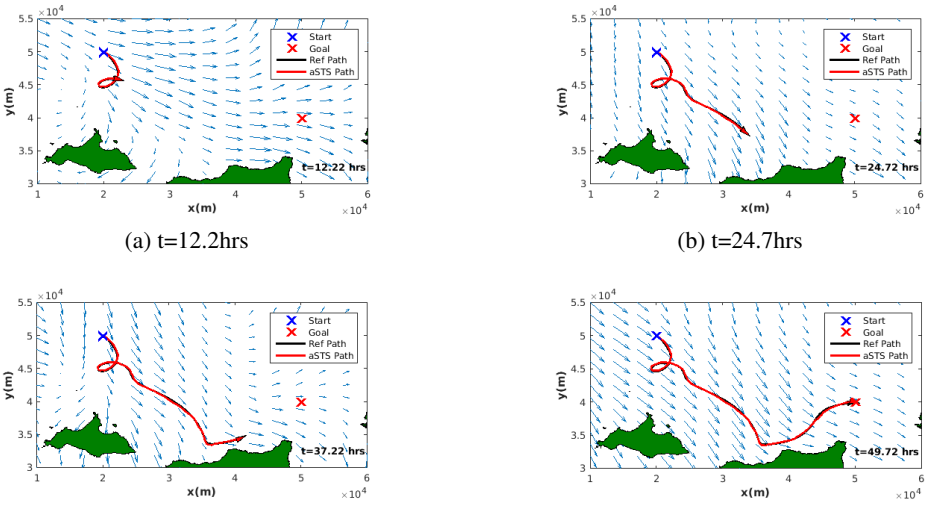
$$mE = \int_{t_s}^{t_g} \frac{\|\Gamma^*(t) - \Gamma(t)\|}{t_g - t_s} dt, \tag{24}$$

was used to evaluate the relative accuracy of computed paths. One could alternatively use the path cost to compare the results. However, the computed path cost is a function of the discretization resolution and thus does not provide a good measure for path quality. For example, a path with only two intermediate nodes might have better cost because it ignores the flow variation at intermediate points along the path.

Simulations using ocean flow data The aSTS method was also used to compute optimal paths in a 2-D ocean environment. The data was obtained from the Regional Ocean Model System (ROMS) for the Santa Barbara Bay area. The Southern California Coastal Ocean Observing System (SCCOOS) [4] generates these hourly ocean current forecasts everyday and each forecast is for 72 hours. The data generated on July 7 and July 8 2016 were used for the simulations. The ROMS data has a $3km \times 3km \times 1hr$ spatio-temporal resolution and linear interpolation is used to obtain flow velocities at intermediate coordinates. The maximum flow speed was $V_{fm} = 0.73m/s$ and as such V_{max} was selected to be $0.5m/s$. A hexagonal lattice with $q = 3$ was used to discretize the reachable space and the error parameter was set as $p = 0.1$. Fig. 16 shows the comparison between the reference path and the path computed using the aSTS method. The reference path computed using the optimal control formulation had a cost of 3775J. It can be seen that the paths match closely with a mean error of $mE = 169$m. The path cost obtained from the aSTS method was 4243J. As before $K_h = 0.0005$ and $K_d = 1$ were used for the simulations, and as a result, more prominence is given to reducing the energy expended to overcome the drag forces. Therefore, the computed path tends to follow the flow as much as possible in order to reduce relative motion between the flow and the vehicle. This leads to the loop structure that can be observed in Fig. 16. The video for this simulation trial can be found on https://youtu.be/6R0RevaAMmY.

Table 1 shows the results of the aSTS method for different p and q values. It shows that increasing q while keeping p constant (paths 1-3) results in more accurate paths at

[4] URL: http://www.sccoos.org/data/roms-3km/



(a) t=12.2hrs (b) t=24.7hrs

(c) t=37.2hrs (d) t=49.7hrs

Fig. 16: Comparison of optimal path computed using the aSTS method (red) with the path computed using the optimal control formulation (black) in an ocean environment. The parameters for the aSTS method were set at $q = 3$, $p = 0.1$. Mean path error for the aSTS path is $mE = 169m$.

the expense of running time. Similarly, decreasing p while keeping q constant (paths 5-3) also results in more accurate paths at the expense of running times. Note that the running time is only a fraction of the total path duration for each case. Table 2 shows the results for paths computed using a fixed-grid discretization scheme. In this fixed discretization method, the user has to specify Δx and Δt, the spatial and temporal discretization resolutions, as well as nHx and nHt, the number of spatial and temporal hops considered as neighbors at each expansion. In the results shown in Table 2, all paths were computed with $\Delta x = 150m$ and $nHx = 3$ so that it has the same spatial discretization as the mean resolution obtained for path 3 of the aSTS method. It can be seen that Path 1 has a similar accuracy as path 3 of the aSTS method, and the computation is faster. However, the results obtained from the fixed discretization method depends heavily on the user defined discretization resolution. For example, a slight increase in the temporal resolution (path 2) results in a less accurate path. The results degrade further when the number of temporal neighbors considered are decreased (path3). Thus, increased accuracy cannot be guaranteed even if the discretization is made finer with such fixed discretization schemes. However, accuracy is guaranteed to improve in the proposed aSTS method by simply increasing q and decreasing p.

Simulations using the double-gyre flow model The aSTS method was used to compute optimal energy paths in a 3-D environment where the horizontal components $V_{f1}(\mathbf{x},t)$ and $V_{f2}(\mathbf{x},t)$ of flow field were given by the double gyre flow in (14), and the vertical component V_{f3} was constant. The parameters for the double gyre flow were

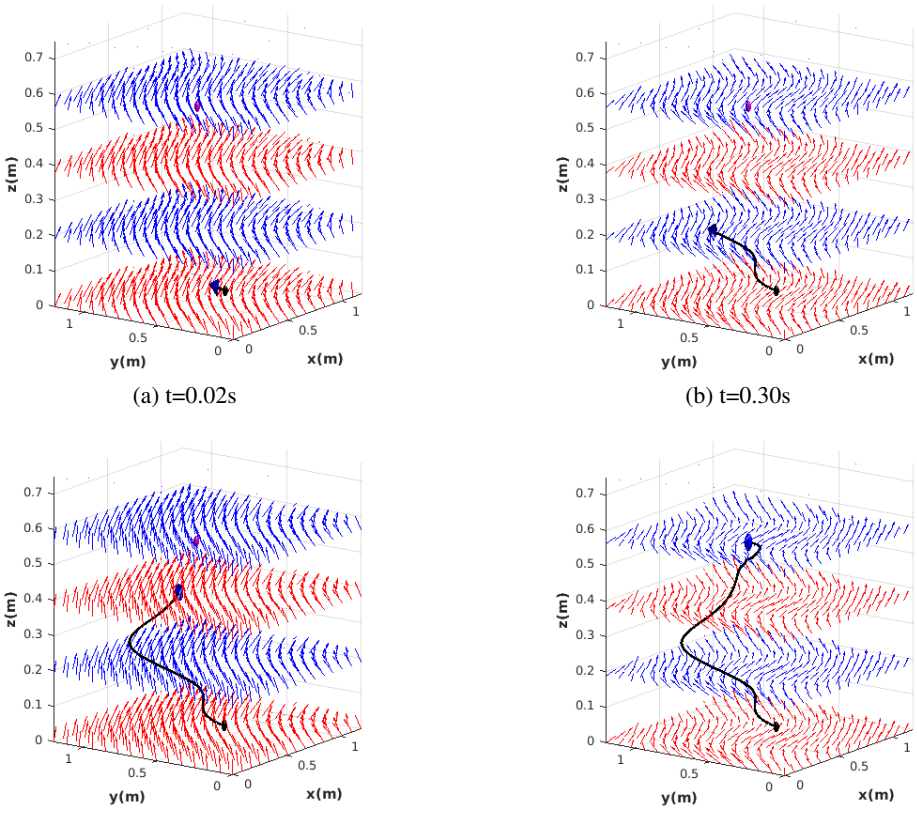
Table 1: Results for the aSTS method for different p and n values.

Path	**1**	**2**	**3**	**4**	**5**
p	0.1	0.1	0.1	0.2	0.3
q	1	2	3	3	3
mErr (m)	6426	595.3	169.2	656.3	1095
cost (J)	9200	5284	4243	4234	4179
running time (s)	385	1074	2009	687	364
path duration (s)	182800	176200	1782000	179400	180600

Table 2: Results for the fixed discretization method.

Path	**1**	**2**	**3**
Δt	150	100	100
nHt	7	10	5
mErr (m)	226.5	326.0	1256
cost (J)	4191	4236	7195
running time (s)	1399	2641	816
path duration (s)	180000	180600	176000

set as $A = 1$, $\mu = 0$, $\omega = 4\pi$ and $\varepsilon = 0.6$ while the constant vertical flow component was set at $V_{f3} = 0.5m/s$. Figure 17 shows snapshots of the path computed by the aSTS method. The computed path cost was 0.0389J and the path duration was 0.84s. When the discretization was done with 37 nodes the path cost was increased to 0.0673J. Thus, as expected, the higher the number of nodes used to discretize the reachable space, the better the computed path is.



(a) t=0.02s (b) t=0.30s

(c) t=0.56s (d) t=0.84s

Fig. 17: Snapshots of the path computed in a 3-D environment subject to a time-varying flow field. The blue and red spheres indicate the start and goal positions respectively.

6 Future Outlook

In this chapter, we presented an energy efficient distributed adaptive sampling strategy for a team of mobile robots to track a dynamic spatio-temporal process. We show

how a team of mobile robots can build a reduced-order model (ROM) of the process using sparse measurements obtained from onboard sensors. The ROM is then used to infer measurements in regions lacking sensor coverage and the inferred data is then used to determine next best locations for new measurements and for planning minimum energy paths to the new locations. In this work, we rely on POD analysis in the derivation of the ROM which identifies the dominant dynamics of the process based on point measurements at specified locations over time. As such, the quality of the ROM is highly dependent on both the spatio-temporal resolution of the measurement data. Consequently, the quality of the minimum energy trajectories computed using the ROM depends on the spatio-temporal resolution of the model. Under these circumstances, the proposed strategies can be significantly improved by incorporating specific geophysical fluid dynamics (GFD) insights.

Geophysical fluid dynamics is the study of natural fluid ows that span large physical scales, such as oceans, the atmosphere, and rivers. GFD flows are naturally stochastic and aperiodic, yet exhibit coherent structure. Recent studies of coherent structures in GFD and other flows have shown that these ubiquitous structures are typically kinematic and play a key role in transport. The Gulf stream is a prominent example of a coherent jet whose heat transport is a critical component of global weather and climate. Using GFD models, details from flows such as the Gulf stream can be used to diagnose the underlying geophysical fluid dynamics. This, in turn, enables the prediction of various physical, chemical, and biological processes in general geophysical flows. While the nature of a particular flow may originate as the solution to some dynamical (force) balance, the characteristics of the flow are an intrinsic property of its velocity vector field. A number of important processes, including transport, can be examined by this purely kinematic description of the flow. It is for this reason that we can witness similar transport features across a broad range of flows and that in the control of vehicles, as for tracers, knowledge of coherent structures may be exploited more effectively than knowledge of the detailed predictions offered by state-of-the-art primitive equation PDE models.

This is further supported by the work of Inanc *et al.* [26] where they showed that minimum energy and time optimal paths in the ocean can coincide with a specific class of coherent structures that are important for quantifying transport phenomena in flows called Lagrangian coherent structures (LCS). LCS are the extensions of stable and unstable manifolds to general time dependent flows [21, 20] and act as separatrices that divide the flow into dynamically distinct regions. Since these separatrices are inherently unstable and denote regions of the flow where more escape events may occur [15], knowing the LCS locations is also important for keeping sensors in specific monitoring regions[42, 24, 22]. These results strongly suggest that it is possible to improve the overall quality of deployment strategies when incorporating such GFD knowledge in the planning, coordination, and controller synthesis process for autonomous marine vehicles. This makes intuitive sense since coherent structures provide a reduced-order description of the complex fluid environment and enable the estimation of the underlying geophysical fluid dynamics. In fact, LCS is one example of a transport controlling feature that can be leveraged to improve the quality ROM of the spatio-temporal process

of interest and simultaneously be leveraged to improve vehicle maneuverability when planning minimum energy paths.

In the past few years, we have developed distributed tracking strategies for teams of robots to track these structures relying solely on onboard measurements obtained by the robots [44, 45, 43, 30, 31]. An immediate direction for future work is to develop adaptive sampling strategies where the team is tasked to track both the coherent structures within the workspace while simultaneously making measurements in specific regions of the process itself. The idea is to improve the ROM by assimilating information about the locations of the LCS in addition to specific measured quantities of the process itself. Since LCS encodes both the stable and unstable modes of the process dynamics, the incorporation of this information can significantly improve the predictive capabilities of the ROM, and thus improve subsequent deployment strategies that employs it.

Another direction for future inquiry is the development of path planning strategies that leverage the locations of the known LCS in the workspace. Since LCS has been shown to be both persistent [47, 46] as well as important for time and energy optimal navigation in the ocean [26, 54], they could potentially be used as optimal paths in regions for which explicit flow descriptions are not available. Furthermore, if the adaptive sampling strategy includes the explicit tracking of LCS locations, the information can significantly reduce the computational burden of computing an energy optimal assignment of robots to the next best location for obtaining new measurements. Lastly, such a strategy can further leverage recent stochastic control strategies that leverage LCS knowledge to enable energy constrained vehicles to efficiently navigate from one LCS-bounded region to another [36]. The ability to develop robust navigation strategies in the presence of model and/or forecast uncertainties is extremely important to ensure prolonged monitoring of dynamic and uncertain environments such as the ocean.

Appendix

Proof of Theorem 1 (See page 14)

Theorem 1. *For sufficient mixing time τ, as determined by the criteria for mixing speeds of Markov Chains [28], $\delta = t_{i+1} - t_i$ the time between collected snapshots, $\left\| \frac{d\boldsymbol{f}(t)}{dt} \right\| \leq \xi$ for $\boldsymbol{f}$ describing the dynamic process, the bound between $(\boldsymbol{P} + \boldsymbol{P}^{\perp})$, the projection corresponding to the dynamic process and $\boldsymbol{P}_{\hat{\boldsymbol{Q}}_i}$, the projection onto the POD basis vectors estimated by robots with high probability is defined as:*

$$\left\| (\boldsymbol{P} + \boldsymbol{P}^{\perp}) - \boldsymbol{P}_{\hat{\boldsymbol{Q}}_i} \right\|_2 \leq e^{\gamma\tau}(2\lambda_{k+1} + \frac{\delta^2}{8}\xi) + \mathscr{O}\left| \frac{\lambda_{k+1}}{\lambda_k} \right|^{\tau} * n + (C+2)\varepsilon^{4\tau} \tag{13}$$

Proof. For POD analysis, m snapshots of the field are collected, either through experimentation or numerical simulations, such that at each time $t = 1, ..., m$, $\boldsymbol{z(t)} = [z_1(t), ..., z_n(t)]^{\top}$, where n is the spatial dimension of some discretization of the flow field. A covariance matrix is constructed as

$$\boldsymbol{K} = \frac{1}{m} \sum_{t=1}^{m} \boldsymbol{z(t)} \boldsymbol{z(t)}^{\top} = \frac{1}{m} \boldsymbol{X} \boldsymbol{X}^{\top}, \tag{25}$$

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

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

$$\boldsymbol{K\phi}_i = \lambda_i \boldsymbol{\phi}_i, \tag{26}$$

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.

In practice, $m < n$ and we assume $rank(\boldsymbol{X}) = m$. For $\boldsymbol{XX}^\top \in \mathbb{R}^{n \times n}$ and $\boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^{m \times m}$, $rank(\boldsymbol{XX}^\top) = rank(\boldsymbol{X}^\top \boldsymbol{X}) \leq m$. Let σ_i, for $i = 1, \ldots, m$ be the singular values $\boldsymbol{X}$. The squares of the m largest singular values of $\boldsymbol{X}$ are the m non-zero eigenvalues of $\boldsymbol{XX}^\top$ and $\boldsymbol{X}^\top \boldsymbol{X}$, as in $\sigma_i^2 = \lambda_i$ is an eigenvalue of $\boldsymbol{XX}^\top$ and $\boldsymbol{X}^\top \boldsymbol{X}$.

Additionally, suppose $\boldsymbol{U}$ is the matrix whose columns are the set of orthonormal eigenvectors of $\boldsymbol{XX}^\top$ and $\boldsymbol{V}$ is the matrix whose columns are the set of orthonormal eigenvectors of $\boldsymbol{X}^\top \boldsymbol{X}$. Let Σ be the matrix with diagonal entries σ_i. Then, from singular value decomposition, we have that

$$\boldsymbol{X} = \boldsymbol{U\Sigma V}^\top. \tag{27}$$

The original basis $\boldsymbol{U}$ 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

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \geq E. \tag{28}$$

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

$$\boldsymbol{z(t)} = \boldsymbol{\Phi c(t)}, \tag{29}$$

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.

Using the SVD, the truncated expression for $\boldsymbol{z(t)}$ with just k vectors in the new basis can then be written as

$$\hat{\boldsymbol{z}}(\boldsymbol{t_j}) = \sum_{i=1}^{k} \sigma_i v_{ji} \boldsymbol{\phi_i}. \tag{30}$$

Define the projection matrix $\boldsymbol{P} = \boldsymbol{\Phi\Phi}^\top$. Define $\hat{\boldsymbol{U}}$ such that the columns are the non-zero eigenvectors in $\boldsymbol{U}$ that are not in $\boldsymbol{\Phi}$. Thus, $\boldsymbol{P}^\perp = \hat{\boldsymbol{U}}\hat{\boldsymbol{U}}^\top$. We know that $\boldsymbol{P}$ is the projection onto $span\{\phi_1, \ldots, \phi_k\}$ and $\boldsymbol{P}^\perp$ is the projection onto $span\{\phi_{k+1} \ldots \phi_m\}$.

We can derive the following upper bounds on the projections of $\boldsymbol{x(t_j)}$ onto the two subspaces, using the representation from Eq. (30).

$$\begin{gathered} \left\|\boldsymbol{Pz(t_j)}\right\|_2 = \left\|\hat{\boldsymbol{z}}(\boldsymbol{t_j})\right\|_2 = \left\|\sum_{i=1}^{k} \sigma_i v_{ji} \boldsymbol{\phi_i}\right\|_2 = \\ \sqrt{\sum_{i=1}^{k} (\sigma_i v_{ji} \boldsymbol{\phi_i})^2} = \sqrt{\sum_{i=1}^{k} (\sigma_i v_{ji})^2} \leq \sigma_1 \sqrt{\sum_{i=1}^{k} (v_{ji})^2} \leq \sigma_1 \end{gathered} \tag{31}$$

$$\left\|P^{\perp}\boldsymbol{z}(\boldsymbol{t_j})\right\|_2 = \left\|\sum_{i=1}^{m-k} \sigma_{i+k} v_{j(i+k)} \boldsymbol{\phi_i}\right\|_2 = \sqrt{\sum_{i=1}^{m-k} (\sigma_{i+k} v_{j(i+k)} \boldsymbol{\phi_i})^2} \leq \sigma_{k+1} \tag{32}$$

.

In Eq. (31) and Eq. (32), σ_i and v_{ji} are constants, and so $(\sigma_i v_{ji} \boldsymbol{\phi_i})^2 = (\sigma_i^2 v_{ji}^2 \boldsymbol{\phi_i}^2)$. Since $\boldsymbol{\phi_i}$ are orthonormal, $\boldsymbol{\phi_i}^2 = 1$. $\sigma_1 \geq \sigma_i$ for all i and can thus be factored out of the summation term. $\sqrt{\sum_{i=1}^{k} (v_{j(i+k)})^2} = \boldsymbol{v_j}^2 = 1$ from the definition of dot product and since v_j are orthonormal.

Given a function $\boldsymbol{f}$ defining the dynamic process of interest, the POD approximation is defined as

$$\begin{aligned} \dot{\hat{\boldsymbol{z}}}(t) &= \boldsymbol{P}\boldsymbol{f}(\hat{z},t) \\ \hat{z}(0) &= \boldsymbol{P}\boldsymbol{z}(0) \end{aligned} \tag{33}$$

The exact solution is defined as

$$\begin{aligned} \dot{\boldsymbol{z}}(t) &= (\boldsymbol{P}+\boldsymbol{P}^{\perp})\boldsymbol{f}(\boldsymbol{z},t) \\ \boldsymbol{z}(0) &= (\boldsymbol{P}+\boldsymbol{P}^{\perp})\boldsymbol{z}(0) \end{aligned} \tag{34}$$

The error between the exact solution and the POD approximation is defined as

$$\begin{aligned} \dot{\boldsymbol{e}}(t) &= \boldsymbol{P}(\boldsymbol{f}(\boldsymbol{x},t) - \boldsymbol{f}(\hat{\boldsymbol{z}},t)) + \boldsymbol{P}^{\perp}\boldsymbol{f}(\boldsymbol{z},t) \\ \boldsymbol{e}(0) &= \boldsymbol{P}^{\perp}\boldsymbol{z}(0) \end{aligned} \tag{35}$$

For $\boldsymbol{f}$ differentiable, $\hat{\boldsymbol{z}}$ is defined by Taylor's Theorem

$$\boldsymbol{f}(\boldsymbol{z},t) - \boldsymbol{f}(\hat{\boldsymbol{z}},t) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}(\hat{\boldsymbol{z}}(t),t)\boldsymbol{e}(t) \tag{36}$$

We can define a bound as

$$\begin{aligned} \boldsymbol{A}(t) &= \boldsymbol{P}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}(\hat{\boldsymbol{z}}(t),t) \\ \|\boldsymbol{A}(t)\|_2 &\leq \gamma, \forall t \in [0,T] \end{aligned} \tag{37}$$

Then, integrating the error Eq. (35)

$$\begin{aligned} \boldsymbol{e}(t) &= \boldsymbol{P}^{\perp}\boldsymbol{z}(0) + \int_0^t \boldsymbol{P}(\boldsymbol{f}(\boldsymbol{z},s) - \boldsymbol{f}(\hat{\boldsymbol{z}},s)) + \boldsymbol{P}^{\perp}\boldsymbol{f}(\boldsymbol{z},s)ds \\ &= \boldsymbol{P}^{\perp}\boldsymbol{z}(0) + \int_0^t \boldsymbol{A}(s)\boldsymbol{e}(s) + \boldsymbol{P}^{\perp}\boldsymbol{f}(\boldsymbol{z},s)ds \\ &= \boldsymbol{P}^{\perp}\boldsymbol{z}(0) + \int_0^t \boldsymbol{A}(s)\boldsymbol{e}(s)ds + \int_0^t \boldsymbol{P}^{\perp}\boldsymbol{f}(\boldsymbol{z},s)ds \end{aligned} \tag{38}$$

By evaluating $\int_0^t \boldsymbol{P}^\perp \boldsymbol{f}(\boldsymbol{z},s)ds$, we have

$$\begin{aligned}
&\int_0^t \boldsymbol{P}^\perp \boldsymbol{f}(\boldsymbol{z},s)ds = \int_0^t \boldsymbol{P}^\perp \dot{\boldsymbol{z}}(s)ds = \boldsymbol{P}^\perp(\boldsymbol{z}(t)-\boldsymbol{z}(0)) \\
\implies &\boldsymbol{e}(t) = \boldsymbol{P}^\perp \boldsymbol{z}(0) + \boldsymbol{P}^\perp(\boldsymbol{z}(t)-\boldsymbol{z}(0)) + \int_0^t \boldsymbol{A}(s)\boldsymbol{e}(s)ds \\
&= \boldsymbol{P}^\perp \boldsymbol{z}(t) + \int_0^t \boldsymbol{A}(s)\boldsymbol{e}(s)ds
\end{aligned} \tag{39}$$

We can bound the error using the following relations

$$\begin{aligned}
\|\boldsymbol{e}(t)\|_2 &= \left\| \boldsymbol{P}^\perp \boldsymbol{z}(t) + \int_0^t \boldsymbol{A}(s)\boldsymbol{e}(s)ds \right\|_2 \\
\|\boldsymbol{e}(t)\|_2 &\le \left\| \boldsymbol{P}^\perp \boldsymbol{z}(t) \right\|_2 + \left\| \int_0^t \boldsymbol{A}(s)\boldsymbol{e}(s)ds \right\|_2 \\
\|\boldsymbol{e}(t)\|_2 &\le \left\| \boldsymbol{P}^\perp \boldsymbol{z}(t) \right\|_2 + \int_0^t \|\boldsymbol{A}(s)\boldsymbol{e}(s)ds\|_2 \\
\|\boldsymbol{e}(t)\|_2 &\le \left\| \boldsymbol{P}^\perp \boldsymbol{z}(t) \right\|_2 + \gamma \int_0^t \|\boldsymbol{e}(s)\|_2 ds
\end{aligned} \tag{40}$$

Theorem 2. (Gronwall's Theorem) *Let α, β, u be real-valued and defined on the interval $[a,b]$. Assume β and u are continuous and α_- is integrable on every closed and bounded subinterval of $[a,b]$. If $\beta \ge 0$ and u satisfies $u(t) \le \alpha(t) + \int_a^t \beta(s)u(s)ds$, then $u(t) \le \alpha(t) + \int_a^t \alpha(s)\beta(s)\exp(\int_s^t \beta(r)dr)ds$, $t \in [a,b]$.*

Applying Gronwall's Theorem to Eq. (40)

$$\begin{aligned}
&u(t) = \|\boldsymbol{e}(t)\|_2, \alpha(t) = \left\| \boldsymbol{P}^\perp \boldsymbol{z}(t) \right\|_2, \beta(t) = \gamma, \\
&\|\boldsymbol{e}(t)\|_2 \le \left\| \boldsymbol{P}^\perp \boldsymbol{z}(t) \right\|_2 + \int_0^t \left\| \boldsymbol{P}^\perp \boldsymbol{z}(t) \right\|_2 * a * e^{\gamma(t-s)} \\
&\|\boldsymbol{e}(t)\|_2 \le e^{\gamma} \max_{t\in[0,T]} \left\| \boldsymbol{P}^\perp \boldsymbol{z}(t) \right\|_2
\end{aligned} \tag{41}$$

The above bound does not take into account the construction of POD matrices, namely how the snapshots are collected. Now, we will investigate the use of snapshot matrix $\boldsymbol{X} = [\boldsymbol{z}(t_1), \dots \boldsymbol{z}(t_m)]$ with projection matrix $\boldsymbol{P}$.

Using Lagrange interpolation

$$\begin{aligned}
\boldsymbol{z}(t) &= \boldsymbol{z}(t_i) * \frac{t-t_{i+1}}{t_i - t_{i+1}} + \boldsymbol{z}(t_{i+1}) \frac{t-t_i}{t_{i+1}-t_i} + \boldsymbol{R}(\boldsymbol{z}(t)), \\
\boldsymbol{R}(\boldsymbol{z}(t)) &= \frac{(t-t_i)(t-t_{i+1})}{2} \frac{d^2\boldsymbol{z}(t)}{dt^2} = \frac{(t-t_i)(t-t_{i+1})}{2} \frac{d\boldsymbol{f}(t)}{dt}
\end{aligned} \tag{42}$$

Multiplying by $\boldsymbol{P}^\perp$

$$\begin{aligned} \boldsymbol{z}(t) &= \boldsymbol{z}(t_i)\frac{t-t_{i+1}}{t_i-t_{i+1}} + \boldsymbol{z}(t_{i+1})\frac{t-t_i}{t_{i+1}-t_i} + \frac{(t-t_i)(t-t_{i+1})}{2}\frac{d\boldsymbol{f}(t)}{dt} \\ \boldsymbol{P}^{\perp}\boldsymbol{z}(t) &= \boldsymbol{P}^{\perp}\boldsymbol{z}(t_i)\frac{t-t_{i+1}}{t_i-t_{i+1}} + \boldsymbol{P}^{\perp}\boldsymbol{z}(t_{i+1})\frac{t-t_i}{t_{i+1}-t_i} + \boldsymbol{P}^{\perp}\frac{(t-t_i)(t-t_{i+1})}{2}\frac{d\boldsymbol{f}(t)}{dt} \end{aligned} \tag{43}$$

For $\delta = t_{i+1} - t_i$, $max_{t\in[t_i,t_{i+1}]}|(t-t_i)(t-t_{i+1})| = \delta/4$. We assume $\frac{d\boldsymbol{f}(t)}{dt} \in C(O,T)$ and $\left\|\frac{d\boldsymbol{f}(t)}{dt}\right\|_2 \leq \xi$. Taking the norm of both sides:

$$\begin{aligned} \left\|\boldsymbol{P}^{\perp}\boldsymbol{z}(t)\right\|_2 &= \left\|\boldsymbol{P}^{\perp}\boldsymbol{z}(t_i)\frac{t-t_{i+1}}{t_i-t_{i+1}}\right\|_2 + \left\|\boldsymbol{P}^{\perp}\boldsymbol{z}(t_{i+1})\frac{t-t_i}{t_{i+1}-t_i}\right\|_2 + \left\|\boldsymbol{P}^{\perp}\frac{(t-t_i)(t-t_{i+1})}{2}\frac{d\boldsymbol{f}(t)}{dt}\right\|_2 \\ \left\|\boldsymbol{P}^{\perp}\boldsymbol{z}(t)\right\|_2 &\leq \left\|\boldsymbol{P}^{\perp}\boldsymbol{z}(t_i)\right\|_2 + \left\|\boldsymbol{P}^{\perp}\boldsymbol{z}(t_{i+1})\right\|_2 + \frac{\delta^2}{8}\left\|\boldsymbol{P}^{\perp}\frac{d\boldsymbol{f}(t)}{dt}\right\|_2 \end{aligned} \tag{44}$$

Using (32), $\left\|\boldsymbol{P}^{\perp}\boldsymbol{z}(t_i)\right\|_2 \leq \sigma_{k+1}$ and $\left\|\boldsymbol{P}^{\perp}\boldsymbol{z}(t_{i+1})\right\|_2 \leq \sigma_{k+1}$

$$\left\|\boldsymbol{P}^{\perp}\boldsymbol{z}(t)\right\|_2 \leq 2\sigma_{k+1} + \frac{\delta^2}{8}\left\|\boldsymbol{P}^{\perp}\right\|\xi \tag{45}$$

Substituting into (41), with

$$\|\boldsymbol{e}(t)\|_2 \leq e^{\gamma t}(2\sigma_{k+1} + \frac{\delta^2}{8}\xi) \tag{46}$$

In the proposed method, we base our algorithm off of the following centralized relationships for orthogonal iteration, used in computing a matrix of eigenvectors $\boldsymbol{Q}$ and eigenvalues contained in the diagonals of $\boldsymbol{R}$. For the design matrix $\boldsymbol{X}$, we have $\boldsymbol{V} = \frac{1}{m}\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{Q} = \boldsymbol{K}\boldsymbol{Q}$, $\boldsymbol{W} = \boldsymbol{V}^T\boldsymbol{V}$, $\boldsymbol{W} \overset{QR}{=} \boldsymbol{R}^T\boldsymbol{R}$, $\boldsymbol{Q}' = \boldsymbol{V}\boldsymbol{R}^{-1}$.

For the decentralized implementation, we start with a matrix $\hat{\boldsymbol{Q}}$, we compute $\hat{\boldsymbol{V}} = \frac{1}{m}\boldsymbol{X}\boldsymbol{X}^T\hat{\boldsymbol{Q}}$. However, instead of $\hat{\boldsymbol{V}}$, for each robot, we have some $\hat{\boldsymbol{V}}_{\boldsymbol{i}} = \hat{\boldsymbol{V}} + \boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}$, since $\hat{\boldsymbol{V}}$ is estimated using the push-sum algorithm. Then, we compute $\hat{\boldsymbol{W}} = \hat{\boldsymbol{V}}_{\boldsymbol{i}}^{\boldsymbol{T}}\hat{\boldsymbol{V}}_{\boldsymbol{i}}$. Again, instead of $\hat{\boldsymbol{W}}$, we have $\hat{\boldsymbol{W}}_{\boldsymbol{i}} = \hat{\boldsymbol{W}} + \boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{W}}$.

We use the following relationships in the proofs below. First, we have that $\|\boldsymbol{V}\|_F = \|\boldsymbol{R}\|_F$ and $\|\boldsymbol{V}\|_2 = \|\boldsymbol{R}\|_2$ since $\boldsymbol{Q}$ is orthonormal. For $\boldsymbol{V} = \boldsymbol{K}\boldsymbol{Q}$, we use the submultiplicativity of norms to get $\|\boldsymbol{V}\|_F \leq \|\boldsymbol{A}\|_F$ and $\|\boldsymbol{V}\|_2 \leq \|\boldsymbol{A}\|_2$. Since $\boldsymbol{W} = \boldsymbol{R}^T\boldsymbol{R}$, $\|\boldsymbol{W}\| \leq \|\boldsymbol{R}\|_F^2$ and $\|\boldsymbol{W}\| = \|\boldsymbol{R}\|_2^2$. From $\boldsymbol{V} = \boldsymbol{K}\boldsymbol{Q}$ and $\hat{\boldsymbol{V}}_{\boldsymbol{i}} = \boldsymbol{K}\hat{\boldsymbol{Q}} + \boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}$, we have that $\left\|\boldsymbol{V} - \hat{\boldsymbol{V}}_{\boldsymbol{i}}\right\|_F \leq \|\boldsymbol{K}\|_2\left\|\boldsymbol{Q} - \hat{\boldsymbol{Q}} - \boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}\right\|_F$. In the proof, we assume that the quantity $\left\|\boldsymbol{Q} - \hat{\boldsymbol{Q}} - \boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}\right\|_F$ is bounded by a constant C.

$$\begin{aligned}
\hat{\boldsymbol{V}}_{\boldsymbol{i}} &= \frac{1}{m}\boldsymbol{X}\boldsymbol{X}^T\hat{\boldsymbol{Q}} + \boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}} \\
\left\|\boldsymbol{V} - \hat{\boldsymbol{V}}_{\boldsymbol{i}}\right\|_F &\leq \|\boldsymbol{K}\|_2 \left\|\boldsymbol{Q} - \hat{\boldsymbol{Q}} - \boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}\right\|_F \\
\left\|\boldsymbol{V} - \hat{\boldsymbol{V}}_{\boldsymbol{i}}\right\|_F &\leq C\,\|\boldsymbol{K}\|_2 \\
\left\|\hat{\boldsymbol{V}}_{\boldsymbol{i}}\right\|_2 - \|\boldsymbol{V}\|_2 &\leq C\,\|\boldsymbol{K}\|_2 \\
\left\|\hat{\boldsymbol{V}}_{\boldsymbol{i}}\right\|_2 &\leq (C+1)\,\|\boldsymbol{V}\|_2
\end{aligned} \tag{47}$$

We can define a bound on error of the matrix $\boldsymbol{W}$ as

$$\begin{aligned}
\left\|\boldsymbol{W} - \hat{\boldsymbol{W}}\right\|_F &= \left\|\boldsymbol{V}^T\boldsymbol{V} - \hat{\boldsymbol{V}}_i^T\hat{\boldsymbol{V}}_i\right\|_F \\
\left\|\boldsymbol{W} - \hat{\boldsymbol{W}}\right\|_F &\leq \left\|\boldsymbol{V}^T\boldsymbol{V} - \hat{\boldsymbol{V}}_i^T\boldsymbol{V}\right\|_F + \left\|\hat{\boldsymbol{V}}_i^T\boldsymbol{V} - \hat{\boldsymbol{V}}_i^T\hat{\boldsymbol{V}}_i\right\|_F \\
\left\|\boldsymbol{W} - \hat{\boldsymbol{W}}\right\|_F &\leq \|\boldsymbol{V}\|_2 \left\|\boldsymbol{V}^T - \hat{\boldsymbol{V}}_i^T\right\|_F + \left\|\hat{\boldsymbol{V}}_i\right\|_2 \left\|\boldsymbol{V} - \hat{\boldsymbol{V}}_i\right\|_F \\
\left\|\boldsymbol{W} - \hat{\boldsymbol{W}}\right\|_F &\leq (C+2)\,\|\boldsymbol{V}\|_2 \left\|\boldsymbol{V} - \hat{\boldsymbol{V}}_i\right\|_F
\end{aligned} \tag{48}$$

Our choice of τ for the number of iterations of the push-sum algorithm, as in [28], to ensure the bound on the error of the projection onto the space spanned by the eigenvectors allows us to establish the bound $\|\boldsymbol{V}\|_F^2 \leq k\,\|\boldsymbol{K}\|_2^2$, where $\boldsymbol{R} \in \mathbb{R}^{k\times k}$.

Next, we an establish the error incurred by estimating $\boldsymbol{W}$ through the decentralized approach.

$$\begin{aligned}
\left\|\boldsymbol{W} - \hat{\boldsymbol{W}}_{\boldsymbol{i}}\right\|_F &= \left\|\boldsymbol{W} - \hat{\boldsymbol{W}}_{\boldsymbol{i}} + \hat{\boldsymbol{W}} - \hat{\boldsymbol{W}}\right\|_F \\
&\leq \left\|\boldsymbol{W} - \hat{\boldsymbol{W}}\right\|_F + \left\|\boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{W}}\right\|_F \\
&\leq \left\|\boldsymbol{W} - \hat{\boldsymbol{W}}\right\|_F + (C+1)^2\varepsilon k\,\|\boldsymbol{K}\|_2^2 \\
&\leq (C+2)\,\|\boldsymbol{K}\|_2^2 \left\|\boldsymbol{Q} - \hat{\boldsymbol{Q}} - \boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}\right\|_F + (C+1)^2\varepsilon k\,\|\boldsymbol{K}\|_2^2 \\
&\leq \max\{(C+2),(C+1)^2\}\,\|\boldsymbol{K}\|_2^2 * \left(\left\|\boldsymbol{Q} - \hat{\boldsymbol{Q}} - \boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}\right\|_F + \varepsilon k\right)
\end{aligned} \tag{49}$$

Using the results presented in [3] and [28], we have that:

$$\begin{aligned}
\left\|\boldsymbol{R} - \hat{\boldsymbol{R}}_{\boldsymbol{i}}\right\|_F &\leq \left\|\boldsymbol{W}^{-1}\right\|_2 \|\boldsymbol{R}\|_2 \left\|\hat{\boldsymbol{W}} - \hat{\boldsymbol{W}}_{\boldsymbol{i}}\right\|_F \\
&\leq \max\{(C+2),(C+1)^2\}\,\left\|\boldsymbol{R}^{-1}\right\|_2^2 \|\boldsymbol{K}\|_2^3 \left(\left\|\boldsymbol{Q} - \hat{\boldsymbol{Q}} - \boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}\right\|_F + \varepsilon k\right)
\end{aligned} \tag{50}$$

Applying Wedin's Theorem [3], we have for invertible matrices $\boldsymbol{R}$, $\hat{\boldsymbol{R}}_{\boldsymbol{i}}$

$$\left\|\boldsymbol{R}^{-1} - \hat{\boldsymbol{R}}_{\boldsymbol{i}}^{-1}\right\|_2 \leq \frac{1+\sqrt{5}}{2}\left\|\boldsymbol{R} - \hat{\boldsymbol{R}}_{\boldsymbol{i}}\right\|_2 \max\left\{\left\|\boldsymbol{R}^{-1}\right\|_2^2, \left\|\hat{\boldsymbol{R}}_{\boldsymbol{i}}^{-1}\right\|_2^2\right\} \tag{51}$$

Then, applying (50) and (51)

$$\begin{aligned}\left\|\boldsymbol{R}^{-1}-\hat{\boldsymbol{R}}_{\boldsymbol{i}}^{-1}\right\|_2 &\leq \left\|\boldsymbol{R}^{-1}\right\|_2^2\|\boldsymbol{K}\|_2^3\left(\left\|\boldsymbol{Q}-\hat{\boldsymbol{Q}}-\boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}\right\|_F+\varepsilon k\right)\\ &\leq \max\{(C+2),(C+1)^2\}*(1+\sqrt{5})\|\boldsymbol{K}\|_2^3\left\|\boldsymbol{R}^{-1}\right\|_2^4\left(\left\|\boldsymbol{Q}-\hat{\boldsymbol{Q}}-\boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}\right\|_F+\varepsilon k\right)\end{aligned} \tag{52}$$

Using results from [28] related to analyzing the effects of the node on the matrix $\boldsymbol{Q}$, we have that:

$$\begin{aligned}\left\|\boldsymbol{Q}'-\hat{\boldsymbol{Q}}_{\boldsymbol{i}}'\right\|_F &\leq \left\|\hat{\boldsymbol{V}}_{\boldsymbol{i}}-\boldsymbol{V}\right\|_F*\max_i\left\|\hat{\boldsymbol{R}}_i\right\|_2^{-1}+\|\boldsymbol{V}\|_F*\max_i\left\|\boldsymbol{R}^{-1}-\hat{\boldsymbol{R}}_{\boldsymbol{i}}^{-1}\right\|_2\\ &\leq \|\boldsymbol{K}\|_2\left\|\boldsymbol{Q}-\hat{\boldsymbol{Q}}_{\boldsymbol{i}}\right\|_F*\sqrt{2}\left\|\boldsymbol{R}^{-1}\right\|_2\\ &\quad+\sqrt{k}\|\boldsymbol{K}\|_2*\max\{(C+2),(C+1)^2\}*(1+\sqrt{5})\|\boldsymbol{K}\|_2^3\left\|\boldsymbol{R}^{-1}\right\|_2^4\left(\left\|\boldsymbol{Q}-\hat{\boldsymbol{Q}}-\boldsymbol{E}_{\boldsymbol{i}}^{\boldsymbol{V}}\right\|_F+\varepsilon k\right)\\ &\leq \|\boldsymbol{K}\|_2\left\|\boldsymbol{Q}-\hat{\boldsymbol{Q}}_{\boldsymbol{i}}\right\|_F*\sqrt{2}\left\|\boldsymbol{R}^{-1}\right\|_2\\ &\quad+\sqrt{k}\|\boldsymbol{K}\|_2*\max\{(C+2),(C+1)^2\}*(1+\sqrt{5})\|\boldsymbol{K}\|_2^3\left\|\boldsymbol{R}^{-1}\right\|_2^4\left(\left\|\boldsymbol{Q}-\hat{\boldsymbol{Q}}_{\boldsymbol{i}}\right\|_F+\varepsilon k\right)\\ &\leq C^4\sqrt{k}\left\|\boldsymbol{R}^{-1}\right\|_2^4\|\boldsymbol{K}\|_2^4\left(\left\|\boldsymbol{Q}-\hat{\boldsymbol{Q}}_{\boldsymbol{i}}\right\|_F+\varepsilon k\right)\end{aligned} \tag{53}$$

This shows that the error is a factor of $\sqrt{k}(C\left\|\boldsymbol{R}^{-1}\right\|_2\|\boldsymbol{K}\|_2)^4$. We can calculate the total error, defined as $\left\|(\boldsymbol{P}+\boldsymbol{P}^{\perp})-\boldsymbol{P}_{\hat{\boldsymbol{Q}}_{\boldsymbol{i}}}\right\|$, using (53) and the analysis set forth in [28]:

$$\begin{aligned}\left\|(\boldsymbol{P}+\boldsymbol{P}^{\perp})-\boldsymbol{P}_{\hat{\boldsymbol{Q}}_{\boldsymbol{i}}}\right\|_2 &= \left\|(\boldsymbol{P}+\boldsymbol{P}^{\perp})-\boldsymbol{P}_{\hat{\boldsymbol{Q}}}+\boldsymbol{P}_{\hat{\boldsymbol{Q}}}-\boldsymbol{P}_{\hat{\boldsymbol{Q}}_{\boldsymbol{i}}}\right\|_2\\ &\leq \left\|(\boldsymbol{P}+\boldsymbol{P}^{\perp})-\boldsymbol{P}_{\boldsymbol{Q}}\right\|_2+\left\|\boldsymbol{P}_{\boldsymbol{Q}}-\boldsymbol{P}_{\hat{\boldsymbol{Q}}_{\boldsymbol{i}}}\right\|_2\\ &= \left\|(\boldsymbol{P}+\boldsymbol{P}^{\perp})-\boldsymbol{P}_{\boldsymbol{Q}}\right\|_2+\left\|\boldsymbol{Q}\boldsymbol{Q}^T{}_2-\hat{\boldsymbol{Q}}_{\boldsymbol{i}}\hat{\boldsymbol{Q}}_{\boldsymbol{i}}^T\right\|_2\\ &\leq \left\|(\boldsymbol{P}+\boldsymbol{P}^{\perp})-\boldsymbol{P}_{\boldsymbol{Q}}\right\|_2+\left\|\boldsymbol{Q}\boldsymbol{Q}^T-\boldsymbol{Q}\hat{\boldsymbol{Q}}_{\boldsymbol{i}}^T\right\|_F+\left\|\boldsymbol{Q}\hat{\boldsymbol{Q}}_{\boldsymbol{i}}^T-\hat{\boldsymbol{Q}}_{\boldsymbol{i}}\hat{\boldsymbol{Q}}_{\boldsymbol{i}}^T\right\|_F\\ &\leq \left\|(\boldsymbol{P}+\boldsymbol{P}^{\perp})-\boldsymbol{P}_{\boldsymbol{Q}}\right\|_2+(\|\boldsymbol{Q}\|_2+\left\|\hat{\boldsymbol{Q}}_i\right\|_2)\left\|\boldsymbol{Q}-\hat{\boldsymbol{Q}}_{\boldsymbol{i}}\right\|_F\\ &\leq e^{\gamma t}(2\lambda_{k+1}+\frac{\delta^2}{8}\xi)+(C+2)\left\|\boldsymbol{Q}-\hat{\boldsymbol{Q}}_{\boldsymbol{i}}\right\|_F\\ &\leq e^{\gamma t}(2\lambda_{k+1}+\frac{\delta^2}{8}\xi)+\mathscr{O}\left|\frac{\lambda_{k+1}}{\lambda_k}\right|^{\tau}*n+(C+2)\varepsilon^{4\tau}\end{aligned} \tag{54}$$

Acknowledgements

TS, DK, and MAH are supported by ONR Award Nos. N00014-16-1-2216 and N00014-17-1-2690 and National Science Foundation (NSF) grants IIS-1253917 and IIS-1812319. EF is supported by National Science Foundation (NSF) grant DMS-1418956.

Bibliography

[1] Alonso AA, Kevrekidis IG, Banga JR, Frouzakis CE (2004) Optimal sensor location and reduced order observer design for distributed process systems. Computers and Chemical Engineering 28(1-2):27–35, DOI 10.1016/S0098-1354(03)00175-3

[2] Caron D, Stauffer B, Moorthi S, Singh A, Batalin M, Graham E, Hansen M, Kaiser W, Das J, de Menezes Pereira A, A Dhariwal BZ, Oberg C, Sukhatme G (2008) Macro- to fine-scale spatial and temporal distributions and dynamics of phytoplankton and their environmental driving forces in a small subalpine lake in southern California, USA. Journal of Limnology and Oceanography 53(5):2333–2349

[3] Chen KK, Tu JH, Rowley CW (2012) Variants of dynamic mode decomposition: boundary condition, koopman, and fourier analyses. Journal of nonlinear science 22(6):887–915

[4] Chen V, Batalin M, Kaiser W, Sukhatme G (2008) Towards spatial and semantic mapping in aquatic environments. In: IEEE International Conference on Robotics and Automation, Pasadena, CA, pp 629–636

[5] Conway JH, Sloane NJA (1999) Certain Important Lattices and Their Properties, Springer New York, New York, NY, pp 94–135. DOI 10.1007/978-1-4757-6568-7_4, URL https://doi.org/10.1007/978-1-4757-6568-7_4

[6] Cortés J, Martínez S, Karatas T, Bullo F, Member S (2004) Coverage Control for Mobile Sensing Networks. IEEE Transactions on Robotics and Automation 20(2):243–255, DOI 10.1109/TRA.2004.824698

[7] Das J, Py F, Maughan T, OReilly T, Messi M, Ryan J, Sukhatme GS, Rajan K (2012) Coordinated sampling of dynamic oceanographic features with underwater vehicles and drifters. The International Journal of Robotics Research 31(5):626–646, DOI 10.1177/0278364912440736

[8] Das J, Py F, Maughan T, O'Reilly T, Messié M, Ryan J, Rajan K, Sukhatme GS (2014) Simultaneous Tracking and Sampling of Dynamic Oceanographic Features with Autonomous Underwater Vehicles and Lagrangian Drifters, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 541–555. DOI 10.1007/978-3-642-28572-1_37, URL https://doi.org/10.1007/978-3-642-28572-1_37

[9] Das J, Py F, Harvey JB, Ryan JP, Gellene A, Graham R, Caron DA, Rajan K, Sukhatme GS (2015) Data-driven robotic sampling for marine ecosystem monitoring. International Journal of Robotics Research 34(12):1435–1452, DOI 10.1177/0278364915587723

[10] Delaney JR, Barga RS (2009) A 2020 vision for ocean science. In: Hey T, Tansley S, Tolle K (eds) The Fourth Paradigm: Data-Intensive Scientific Discovery, Microsoft Research, pp 27–38

[11] DeVries L, Paley DA (2012) Multivehicle control in a strong flowfield with application to hurricane sampling. Journal of Guidance, Control, and Dynamics 35(3):794–806, DOI 10.2514/1.55580, URL https://doi.org/10.2514/1.55580

[12] Durrant-Whyte HF (1988) Sensor Models and Multisensor Integration. International Journal of Robotics Research 7(6)

[13] Edwards CR, Seim HE (2008) Complex EOF analysis as a method to separate barotropic and baroclinic velocity structure in shallow water. Journal of Atmospheric and Oceanic Technology 25(5):808–821, DOI 10.1175/2007JTECHO562.1

[14] Everson R, Sirovich L (1995) Karhunen–Loève procedure for gappy data. Journal of the Optical Society of America 12(8):1657, DOI 10.1364/JOSAA.12.001657

[15] Forgoston E, Billings L, Yecko P, Schwartz IB (2011) Set-based corral control in stochastic dynamical systems: Making almost invariant sets more invariant. Chaos 21(1):013,116

[16] Frolov S, Paduan J, Cook M, Bellingham J (2012) Improved statistical prediction of surface currents based on historic HF-radar observations. Ocean Dynamics 62(7):1111–1122, DOI 10.1007/s10236-012-0553-5

[17] García MR, Vilas C, Banga JR, Alonso AA (2007) Optimal field reconstruction of distributed process systems from partial measurements. Industrial and Engineering Chemistry Research 46(2):530–539, DOI 10.1021/ie0604167

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

[19] Golub GH, Loan CFV (1996) Matrix Computations, 3rd edn. The Johns Hopkins University Press, Baltimore, MD

[20] Haller G (2011) A variational theory of hyperbolic lagrangian coherent structures. Phys D 240:574–598

[21] Haller G, Yuan G (2000) Lagrangian coherent structures and mixing in two-dimensional turbulence. Phys D 147:352–370

[22] Heckman CR, Schwartz IB, Hsieh MA (2015) Toward efficient navigation in uncertain gyre-like flows. The International Journal of Robotics Research 34(13):1590–1603, DOI 10.1177/0278364915585396

[23] Hollinger GA, Pereira AA, Binney J, Somers T, Sukhatme GS (2016) Learning Uncertainty in Ocean Current Predictions for Safe and Reliable Navigation of Underwater Vehicles. Journal of Field Robotics 33(1):47–66, DOI 10.1002/rob.21613

[24] Hsieh MA, Mallory K, Schwartz IB (2014) Distributed allocation of mobile sensing agents in geophysical flows. In: Proc. of the 2014 American Controls Conference, Portland, OR

[25] Huynh V, Dunbabin M, Smith R (2015) Predictive motion planning for auvs subject to strong time-varying currents and forecasting uncertainties. In: Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp 1144–1151, DOI 10.1109/ICRA.2015.7139335

[26] Inanc T, Shadden S, Marsden J (2005) Optimal trajectory generation in ocean flows. In: American Control Conference, 2005. Proceedings of the 2005, pp 674 – 679, DOI 10.1109/ACC.2005.1470035

[27] Kemna S, Rogers JG, Nieto-Granda C, Young S, Sukhatme GS (2017) 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, DOI 10.1109/ICRA.2017.7989245

[28] Kempe D, McSherry F (2004) A Decentralized Algorithm for Spectral Analysis. Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing pp 561–568, DOI 10.1145/1007352.1007438

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

[30] Kularatne D, Hsieh MA (2015) Tracking attracting lagrangian coherent structures in flows. In: Proceedings of Robotics: Science and Systems, Rome, Italy

[31] Kularatne D, Hsieh MA (2017) Tracking attracting manifolds in flows. Autonomous Robots 41(8):15751588, DOI 10.1007/s10514-017-9628-y, URL https://doi.org/10.1007/s10514-017-9628-y

[32] Kularatne D, Bhattacharya S, Hsieh MA (2016) Computing energy optimal paths in time-varying flows. Tech. rep., Drexel University, Philadelphia, USA, doi:10.17918/D8B66V

[33] Kularatne D, Bhattacharya S, Hsieh MA (2016) Time and energy optimal path planning on a flow field. In: submitted to Robotics: Science and Systems, Ann Arbor, MI USA

[34] Kularatne D, Bhattacharya S, Hsieh MA (2018) Going with the flow: a graph based approach to optimal path planning in general flows. Autonomous Robots 42(7):1369–1387, DOI 10.1007/s10514-018-9741-6, URL https://doi.org/10.1007/s10514-018-9741-6

[35] Kularatne D, Bhattacharya S, Hsieh MA (2018) Optimal path planning in time-varying flows using adaptive discretization. IEEE Robotics and Automation Letters 3(1):458–465, DOI 10.1109/LRA.2017.2761939

[36] Kularatne D, Forgoston E, Hsieh MA (2018) Exploiting stochasticity for the control of transitions in gyre flows. In: Robotics: Science and Systems, Pittsburgh, PA

[37] Kularatne D, Hajieghrary H, Ani Hsieh M (2018) Optimal path planning in time-varying flows with forecasting uncertainties. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp 1–8, DOI 10.1109/ICRA.2018.8460221

[38] Kumar S, Deshpande A, Sarma SE (2012) Stable arrangements of mobile sensors for sampling physical fields. In: American Control Conference (ACC), 2012, IEEE, pp 324–331

[39] Le Ny J, Pappas GJ (2010) Sensor-Based Robot Deployment Algorithms. Proceedings of the IEEE Conference on Decision and Control pp 5486–5492, DOI 10.1109/CDC.2010.5716979

[40] Lolla T, Ueckermann MP, Haley P, Lermusiaux PFJ (2012) Path planning in time dependent flow fields using level set methods. In: in the Proc. IEEE International Conference on Robotics and Automation, Minneapolis, MN USA

[41] Lynch KM, Schwartz IB, Yang P, Freeman RA (2008) Decentralized environmental modeling by mobile sensor networks. IEEE Transactions on Robotics and Automation 24(3):710–724

[42] Mallory K, Hsieh MA, Forgoston E, Schwartz IB (2013) Distributed allocation of mobile sensing swarms in gyre flows. Nonlin Processes Geophys 20(5):657–668

[43] Michini M, Hsieh MA, Forgoston E, Schwartz IB (2014) Experimental validation of robotic manifold tracking in gyre-like flows. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) 2014, Chicago, IL USA
[44] Michini M, Hsieh MA, Forgoston E, Schwartz IB (2014) Robotic tracking of coherent structures in flows. IEEE Trans on Robotics 30(3):593–603
[45] Michini M, Rastgoftar H, Hsieh MA, Jayasuriya S (2014) Distributed formation control for collaborative tracking of manifolds in flows. In: Proc. of the 2014 American Control Conference (ACC 2014), Portland, OR
[46] Olascoaga MJ, Rypina II, Brown MG, Beron-Vera FJ, Kocak H, Brand LE, Halliwell GR, Shay LK (2006) Persistent transport barrier on the West Florida Shelf. Geophys Res Lett 33(22):L22,603
[47] Peacock T, Haller G (2013) Lagrangian coherent structures: The hidden skeleton of fluid flows. Physics Today 66(2):41–47
[48] Peherstorfer B, Willcox K (2016) Dynamic data-driven model reduction: adapting reduced models from incomplete data. Advanced Modeling and Simulation in Engineering Sciences 3(1), DOI 10.1186/s40323-016-0064-x
[49] Penna F, Stanczak S (2015) Decentralized Eigenvalue Algorithms for Distributed Signal Detection in Cognitive Networks. IEEE Transactions on Signal Processing 63(2):427–440
[50] Rowley CW, Mezić I, Bagheri S, Schlatter P, Henningson DS (2009) Spectral analysis of nonlinear flows. Journal of fluid mechanics 641:115–127
[51] Salam T, Hsieh MA (2019) Adaptive sampling and reduced-order modeling of dynamic processes by robot teams. IEEE Robotics and Automation Letters 4(2):477–484, DOI 10.1109/LRA.2019.2891475
[52] Schmid PJ (2010) Dynamic mode decomposition of numerical and experimental data. Journal of fluid mechanics 656:5–28
[53] Schwager M, Vitus MP, Powers S, Rus D, Tomlin CJ (2017) Robust Adaptive Coverage Control for Robotic Sensor Networks. IEEE Transactions on Control of Network Systems 4(3):462–476
[54] Senatore C, Ross S (2008) Fuel-efficient navigation in complex flows. In: American Control Conference, 2008, pp 1244 –1248, DOI 10.1109/ACC.2008.4586663
[55] Shchepetkin A, McWilliams J (1998) Quasi-monotone advection schemes based on explicit locally adaptive dissipation. Monthly Weather Review 126:1541–1580
[56] Shchepetkin AF, McWilliams JC (2005) The regional oceanic modeling system (roms): a split-explicit, free-surface, topography-following-coordinate oceanic model. Ocean Modeling 9:347–404
[57] Sirovich L (1987) Turbulence and the dynamics of coherent structures. I. Coherent Structures. Quarterly of Applied Mathematics 45(3):561–571, DOI 10.1090/qam/910463
[58] Smith R, Pereira AM, Chao Y, Li P, Caron DA, Jones BH, Sukhatme G (2010) Autonomous underwater vehicle trajectory design coupled with predictive ocean models: A case study. In: Proc. of the 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, pp 4770–4777
[59] Smith R, Kelly J, Sukhatme G (2012) Towards improving mission execution for autonomous gliders with an ocean model and kalman filter. In: Proc. of the 2012 IEEE International Conference on Robotics and Automation, Minneapolis, MN

[60] Smith RN, Chao Y, Li PP, Caron DA, Jones BH, Sukhatme GS (2010) Planning and implementing trajectories for autonomous underwater vehicles to track evolving ocean processes based on predictions from a regional ocean model. International Journal of Robotics Research 29(12):1475–1497, DOI http://dx.doi.org/10.1177/0278364910377243, URL http://cres.usc.edu/cgi-bin/print_pub_details.pl?pubid=646

[61] Sydney N, Paley DA (2011) Multi-vehicle control and optimization for spatiotemporal sampling. In: IEEE Conf. Decision and Control, Orlando, FL, pp 5607–5612

[62] Veronis G (1966) Wind-driven ocean circulation, part i and part ii. Deep-Sea Res 13(31)

[63] Wang D, Lermusiaux P, Haley P, Eickstedt D, Leslie W, Schmidt H (2009) Acoustically focused adaptive sampling and on-board routing for marine rapid environmental assessment. Journal of Marine Systems 78:393–407

[64] Willcox K (2006) Unsteady flow sensing and estimation via the gappy proper orthogonal decomposition. Computers and Fluids 35(2):208–226, DOI 10.1016/j.compfluid.2004.11.006

[65] Williams RK, Sukhatme GS (2012) Probabilistic spatial mapping and curve tracking in distributed multi-agent systems. In: 2012 IEEE International Conference on Robotics and Automation, pp 1125–1130, DOI 10.1109/ICRA.2012.6224689

[66] Wu W, Zhang F (2011) Cooperative exploration of level surfaces of three dimensional scalar fields. Automatica, the IFAC Journall 47(9):2044–2051

[67] Zhang F, Fratantoni DM, Paley D, Lund J, Leonard NE (2007) Control of coordinated patterns for ocean sampling. International Journal of Control 80(7):1186–1199