Distributed Environmental Monitoring with Finite Element Robots

Matthew L. Elwin, *Member, IEEE,* Randy A. Freeman, *Member, IEEE,* and Kevin M. Lynch, *Fellow, IEEE*

***Abstract*—We introduce a distributed finite element algorithm that allows swarms of mobile robots to persistently monitor environmental quantities such as temperature or salinity. The robots deploy themselves into the environment, covering the domain and dividing it into non-overlapping regions. Each robot estimates the environment over its own region using local measurements and communication with nearby robots. The algorithm ensures that each robot's estimate constitutes a piece of a global estimate that spans the entire domain, fuses the whole swarm's measurements, and accounts for the spatial correlation between measurement and estimation locations. By incorporating spatial correlation without requiring the transmission of measurements or measurement locations, the algorithm decouples its communication requirements from the spatial statistics of the environment and enables robots with fixed capabilities to monitor environments with different spatial correlation lengths. Analysis and simulation demonstrate that, as the number of robots increases, the memory and communication requirements of each individual robot decrease until reaching a minimum, after which the resolution of the environmental model increases. Additional robots, therefore, add computational resources to the swarm rather than introducing extra computational burdens.**

***Index Terms*—Distributed Robot Systems, Sensor Networks, Networked Robots, Environment Monitoring and Management**

I. Introduction

STUDYING environmental fields such as temperature, chemical concentration, or radiation intensity traditionally requires manually deploying sensors, collecting data at a central computer, and estimating the field from sparse, irregular measurements. Such methods have been studied extensively, for example in oceanography [1], [2]. Using mobile robots to collect these measurements, rather than manually deployed fixed-location sensors, provides more relevant data to scientists, enabling them to create detailed and accurate environmental models.

When estimating an environment, typically a central computer must store all measurements and environmental model states in its memory; therefore, its memory use increases with the number of measurements and model states. To obtain the measurements, the central computer receives transmissions from the robots. As the number of robots and measurements grows, however, limited communication bandwidth reduces the feasibility of this centralized communication scheme, especially when monitoring remote environments such as outer space, the ocean, and underground.

Our environmental monitoring approach is based on a distributed implementation of a finite element method (FEM) (see [3], [4]). We focus on reducing the memory and communication requirements of individual robots, especially as the number of robots and their density in the environment (and therefore the density of the measurements) increases. In particular, we evaluate how much memory each robot needs to store the estimate, how much data the swarm transmits, and the algorithm's communication topology requirements. All three of these criteria directly influence the cost and power consumption of the robots.

Compared to related work, our method significantly reduces the memory consumption and communication requirements of the robots, especially as their density in the environment (relative to a characteristic length scale) increases. Our algorithm also provides additional flexibility over existing methods: its communication requirements remain independent of intrinsic characteristics of the environment, allowing the same communication hardware to be used in multiple settings.

To reduce memory requirements, the robots partition the domain into non-overlapping subdomains. Each robot estimates the environment over its own partition, storing only a piece of the full environmental model; therefore, collectively the robots can estimate environments whose models do not fit inside a single robot's memory. Initially, as the number of robots increases, each individual's memory use decreases because it stores a diminishing share of the full environmental estimate. Eventually, each robot stores a minimum number of states, determined by the subdomain geometry. After reaching this minimum, adding more robots increases the resolution of the environmental model without increasing the memory use of the individual robots.

To fuse all of the swarm's measurements into their estimates, the robots execute a distributed algorithm based on the variational inverse method (VIM) [5]–[12]. This algorithm generates estimates by using FEM to minimize a cost functional that penalizes the estimate's non-smoothness and inconsistency with the measurements. FEM transforms this continuous infinite-dimensional optimization into a discrete finite-dimensional optimization and guarantees convergence.

In our distributed version of VIM, the robots determine their own FEM meshes and solve for their local environmental states using the distributed alternating direction method of multipliers (ADMM) [13], [14]. Despite communicating only with its Voronoi neighbors, each robot's estimate converges to

This work is supported by the Office of Naval Research, Grant N00014-13-1-0331. The authors are with the Center for Robotics and Biosystems, the Department of Mechanical Engineering (Elwin and Lynch), the Northwestern Institute on Complex Systems (Freeman and Lynch), and the Department of Electrical Engineering and Computer Science (Freeman), Northwestern University, Evanston, IL 60208 USA. Emails: elwin@northwestern.edu, freeman@eecs.northwestern.edu, kmlynch@northwestern.edu.

a piece of the global VIM solution.

Assuming that the environment is a spatial random field with known mean and covariance functions (a common assumption in environmental monitoring: see e.g., [1], [2]) the VIM estimate, under certain technical conditions, matches the best linear unbiased estimate (BLUE) of the environment [2], [6], [15], [16]. The BLUE results from taking a linear combination of the measurements, weighted to minimize the mean-squared estimation error. Thus, by matching the global VIM solution within its region, each robot's estimate implicitly incorporates every measurement from the swarm while, due to the FEM approach, its memory and communication requirements remain approximately independent of the number of measurements.

Generally, the spatial correlation function of the random environmental field determines the relative importance of measurements when estimating the field at a given location: measurements whose locations are highly correlated with the estimation location are more important than those with little correlation. For non-increasing correlation functions, the correlation length determines a distance beyond which measurements no longer meaningfully contribute to an estimate. Our algorithm's communication requirements remain independent of the correlation length despite incorporating all relevant measurements into the estimate. This decoupling enhances the applicability of the algorithm, enabling the same robots to measure fields with vastly different spatial correlation lengths.

Overall, distributed coordination of multiple robots monitoring an environment consists of four tasks: deployment, patrol, estimation, and querying (see Figure 1). During deployment, the robots move to cover the environment and partition it into subdomains. Within each subdomain, the robots form local finite element meshes. Next, the robots simultaneously patrol and estimate the environment. To perform the patrol task, the robots move according to a control law and gather measurements. In the estimation task the robots assemble local FEM equations and solve them using distributed ADMM. Estimation provides an approximation of the field value and estimation uncertainty. To track time-varying fields, the estimation task is periodically restarted with new measurements from the patrol task. At any time, the robots may receive a query, which allows them to provide the estimate to a user. The robots may periodically redeploy to obtain coverage with higher densities in areas of greater importance or uncertainty.

This paper introduces a distributed environmental estimation algorithm focusing on the deployment and estimation tasks. In particular, we examine the estimation algorithm's memory and communication requirements, analyze its desirable scaling properties, and demonstrate its reduced requirements relative to prior methods. The patrol and query tasks are not discussed here; see [17]. The decision on when to patrol versus when to redeploy is also not covered in this work. The deployment task, however, can be informed by an uncertainty distribution, which can be obtained using a slight modification to the estimation algorithm presented here; see [17] and Appendix A.

A. Related Work

In some decentralized environmental monitoring approaches, every robot estimates the whole environment. Typically, the robots estimate this global information using consensus algorithms, such as average consensus estimators [18]–[21]. For instance in [22], the environment is represented as a linear combination of basis functions whose coefficients are estimated using a distributed Kalman filter [23], [24]. The approach of [25] represents the environment as samples from probability distributions and uses consensus algorithms to determine the estimate. These consensus-based methods require all robots to store and agree upon the full environment state; therefore, their memory usage and communication requirements increase with the complexity of the environment.

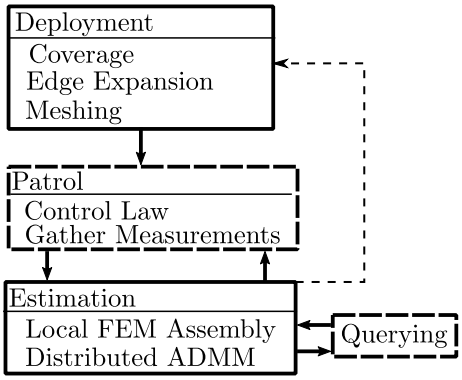

Fig. 1. Overview of tasks required for environmental monitoring. During deployment, the robots move to partition the domain and prepare the finite element mesh. The robots then patrol, using a control law to move and gather measurements. These measurements are used to estimate the environment. Periodically, the robots restart the estimation process with new measurements, allowing them to respond to time-varying fields. The robots can also redeploy to gain better coverage of the environment. If a robot receives a query, the robot responds and continues with its patrol and estimation tasks. Dashed boxes and arrows indicate topics not explored in this paper.

In contrast, methods that use distributed environmental representations do not require every robot to store the full environmental estimate. For example, the method of [26] uses FEM and a Kalman filter to estimate an environment. Unlike our method, this approach is not fully distributed because it assumes that the robots know their pre-assigned subregions and the FEM mesh beforehand. Additionally, the method requires knowledge of an accurate partial differential equation (PDE) model for the environment, which may be unavailable.

Conceptually, both VIM and [26] solve a PDE to determine an environmental estimate. However, in VIM, the PDE model itself incorporates the measurements and measurement locations, enabling it to directly account for the spatial statistics of the environment. The PDE used by VIM can include physics-based models in addition to the spatial statistics: see [27].

In [28], the robots use local interpolation rules to estimate the environment in their own regions. The distributed Kriged Kalman filter of [29] uses average consensus estimators and distributed Jacobi over-relaxation to estimate the environment.[1] Like our method, these algorithms require less communication and memory than consensus-based approaches. However, unlike our approach, both [28] and [29] require the

[1]The environmental representation in [29] has consensus-based and distributed components; for comparison purposes we ignore the consensus-based aspect of the environment representation and the consensus-based part of the algorithm.

robots to communicate with all other robots within a radius determined by the environment's spatial correlation length. The communication requirements of these algorithms also increase with increasing density of the robots.

The work of [30] studies a different estimation problem than ours: tracking and searching for multiple targets. However, the robots use Voronoi-based control algorithms and a distributed Probability Hypothesis Density (PHD) filter to maintain estimates in local Voronoi cells that match a centralized solution.

Although this paper focuses on estimation, designing control laws to efficiently collect measurements is also an important task for environmental monitoring. The method of [31] introduces rapidly exploring random cycles, which create persistent trajectories that robots use to estimate the environment. The ergodic exploration methods of [32]–[34] use an information density map to allow robots to spend more time in areas of high information while still exploring areas with low information. In [35], robotic ocean gliders travel along parameterized paths to collect better information. The control strategy of [36] balances adherence to a trajectory and gathering better information along closed paths.

To generate an estimate, our method uses the over-relaxed pre-conditioned distributed ADMM method of [14] to solve the FEM problem in a distributed manner. Distributed ADMM (see e.g., [13]) is an iterative method that requires three steps: a primal update, a dual update, and a Lagrange multiplier update. The primal and Lagrange multiplier updates are performed locally, while the dual update requires communication.

Theoretically, we can use any distributed optimization method to solve the FEM problem. The algorithm choice, however, significantly affects performance, particularly the total amount of communication needed. We use ADMM because it has been shown to converge in fewer iterations than other first-order methods such as distributed gradient descent [37], it does not require the communication of individual stiffness matrix rows needed by Jacobi over-relaxation [38], and it does not require any global communication steps needed by methods such as distributed conjugate gradient [39], the massively parallel method of [40], or a parallel skyline solver [41].

B. Contribution Summary

We introduce a new distributed algorithm, the distributed variational inverse method (DVIM), for estimating environments from sparse measurements. Our algorithm distributes the environmental representation across the swarm, lowering the communication and memory requirements of the individual robots. Our approach is unique because it maintains an estimate that accounts for the spatial correlation between measurements without requiring direct communication between the robots storing the relevant measurements. This feature relaxes an important communication topology requirement and reduces the memory used by each robot, especially as the density of the robots increases. Additionally, our deployment method ensures that the robots create a properly constructed and well-conditioned FEM mesh in a distributed manner.

This work significantly expands upon our preliminary work in [42]. In contrast to [42], we explicitly consider how the robots are deployed and use their deployment to improve the conditioning (and therefore the convergence speed) of the estimation. We also explicitly quantify the memory and communication requirements of this algorithm and contrast these properties with existing methods. Our expanded analysis and numerical simulations provide evidence of the efficacy of our algorithm in reducing the memory and communication requirements for environmental estimation without sacrificing estimation fidelity relative to a centrally computed solution.

Paper Structure: Section II formulates the environmental estimation problem in a centralized setting. Section III describes the decentralized deployment process. Section IV describes the distributed field estimation method. Section V analyzes the performance of our method and compares it to previous work. Section VI presents agent-based Monte-Carlo simulations, which validate our algorithm.

II. Centralized Problem

A. Environment Model

Table I summarizes the notation used in this section. We model the environment $\phi : \Omega \to \mathbb{R}$ as a spatial random field over the domain $\Omega \subset \mathbb{R}^2$. The field has zero mean and covariance function

$$\mathrm{E}[\phi(x_1)\phi(x_2)] = C(x_1, x_2) = \gamma R(x_1, x_2), \tag{1}$$

where $\mathrm{E}[\cdot]$ is the expected value, $\gamma \in \mathbb{R}$ is the background variance and $R(x_1, x_2)$ is the spatial correlation function, with $x_1, x_2 \in \Omega$. Usually, spatial correlation depends only on the distance between points, so we let $R(x_1, x_2) = R(\|x_1 - x_2\|)$, where $\|\cdot\|$ denotes the Euclidean norm.

We consider n robots, each taking a measurement $z_i \in \mathbb{R}$ at location X_i. The measurement vector $z \in \mathbb{R}^n$ and measurement location matrix $X \in \mathbb{R}^{n\times 2}$ contain the individual measurements and their locations, respectively.

Remark 1. *To simplify our presentation, we have made some assumptions that can be relaxed. Our method handles fields with nonzero mean (by treating the mean as the best estimate of the field in the absence of measurements) and time variation (by incorporating a time-dependent term into the covariance function and repeatedly running the estimation process) (see [17]). Multiple measurements per robot can be allowed with minimal modification (see [17]). Much of this work extends to 3-D, with the exception the edge expansion algorithm used to improve convergence (see Section III-B).*

B. Sensor Model

The measurement model for an individual robot is

$$z_i = \mathcal{H}_i\phi + v_i, \tag{2}$$

where $\mathcal{H}_i$ is a functional that evaluates ϕ at measurement position i (i.e., $\mathcal{H}_i\phi = \phi(X_i)$) and $v_i \sim N(0, \epsilon_i)$ is zero-mean Gaussian noise with variance ϵ_i.

The measurements for the whole swarm are modeled as

$$z = \mathcal{H}\phi + v, \tag{3}$$

TABLE I
SYMBOL DEFINITIONS FOR SECTION II

Symbol	Description
$\Omega \subset \mathbb{R}^2$	Environmental domain
$x \in \Omega$	A position within the domain Ω
$\phi(x) \in \mathbb{R}$	The actual field value at position x
$\gamma \in \mathbb{R}$	Background variance
$R(x_1, x_2) \in \mathbb{R}$	Spatial correlation function
$C(x_1, x_2) \in \mathbb{R}$	Spatial covariance function
n	The number of robots
$z \in \mathbb{R}^n$	Measurement vector
$X \in \mathbb{R}^{n\times 2}$	Measurement positions
$\epsilon \in \mathbb{R}^{n\times n}$	Sensor covariance matrix (diagonal)
$v \sim N(0, \epsilon)$	Sensor noise: Gaussian with covariance ϵ
$\mathcal{H}\phi \in \mathbb{R}^n$	Field evaluated at the measurement locations
$J[\phi] \in \mathbb{R}$	Cost functional used for estimation
$\mu \in \mathbb{R}^n$	Measurement weights in the cost functional
$f(\phi) \in \mathbb{R}$	Smoothness norm used in the cost functional
$L \in \mathbb{R}$	Spatial correlation length
$e(x) \in \mathbb{R}$	Estimation error at position x
$\hat{\phi}(x) \in \mathbb{R}$	Estimate of the field at position x
$\mathbb{K}_1(\cdot)$	First modified Bessel function of the second kind
m	Number of shape functions per element
$w(x) \in \mathbb{R}^m$	Shape function vector
M	Total number of nodes in the FEM mesh
$q_e \in \mathbb{R}^m$	Local node vector for element e
$q \in \mathbb{R}^M$	Global node vector
$B_e \in \mathbb{R}^{m\times M}$	Gather matrix for element e
$K_e \in \mathbb{R}^{m\times m}$	Local stiffness matrix for element e
$K \in \mathbb{R}^{M\times M}$	Global stiffness matrix
$g_e \in \mathbb{R}^m$	Local load vector for element e
$g \in \mathbb{R}^M$	Global load vector
$\mathrm{E}[\cdot]$	Expected value
N	The number of elements in the FEM mesh
$[\cdot]_i$	Vector element i
$[\cdot]_{ij}$	Matrix element (i, j)

where $\mathcal{H}$ is a functional that evaluates ϕ at the measurement positions (i.e., $\mathcal{H}\phi = \begin{bmatrix}\phi(X_1) & \cdots & \phi(X_n)\end{bmatrix}^T$) and $v \sim N(0, \epsilon)$ is zero-mean Gaussian noise with a diagonal covariance matrix $\epsilon \in \mathbb{R}^{n\times n}$. The diagonal components of ϵ are the individual sensor variances ϵ_i. The sensor noise and field value are uncorrelated (i.e., $\mathrm{E}[\phi(x)v] = 0$).

C. Variational Inverse Method

The variational inverse method (VIM) is an interpolation technique closely related to smoothing splines [2], [15], [16]. It generates an environmental estimate by minimizing a cost functional penalizing the estimate's non-smoothness and its inconsistency with the measurements [6], [7], [9]–[11].

The cost functional to minimize is

$$J[\phi] = f(\phi)^2 + \sum_{i=1}^{n} \mu_i (\phi(X_i) - z_i)^2, \tag{4}$$

where $\mu_i \in \mathbb{R}$ is the weight of the i-th measurement (noisier measurements have lower weights) and $f(\phi)$ is a norm used to measure non-smoothness. The vector $\mu \in \mathbb{R}^n$ stacks all the measurement weights.

The spatial coordinates of the two-dimensional domain are denoted $x = ([x]_1, [x]_2) \in \Omega$ (indexing into vectors and matrices is denoted with $[\cdot]_i$ and $[\cdot]_{ij}$, respectively). We use the non-smoothness measure from [6], [9]:

$$f(\phi)^2 = \int_\Omega \left(\left(\frac{\partial^2 \phi}{\partial [x]_1^2}\right)^2 + \left(\frac{\partial^2 \phi}{\partial [x]_2^2}\right)^2 + 2\left(\frac{\partial^2 \phi}{\partial [x]_1 \partial [x]_2}\right)^2 + \frac{2}{L^2}\left(\left(\frac{\partial \phi}{\partial [x]_1}\right)^2 + \left(\frac{\partial \phi}{\partial [x]_2}\right)^2\right) + \frac{1}{L^4}\phi^2 \right) d\Omega. \tag{5}$$

Here, L is the correlation length, a distance scaling parameter that establishes the decay rate of the covariance between two positions as a function of the distance between them.

Remark 2. *The summation in Equation* (4) *penalizes the deviation of the estimates from the measurements and the norm $f(\phi)$ penalizes non-smoothness of the estimate. Technical restrictions on the choice of norm are discussed in [15]. A suitable norm for three dimensions is discussed in [5], [8].*

Remark 3. *The term ϕ^2 in $f(\phi)^2$ penalizes the magnitude of the field, which works for fields with zero mean. When the mean is nonzero, subtracting it from the measurements in Equation* (4) *approximates the anomaly field. The actual field is then recovered from the anomaly field (see [9], [17]).*

Estimate Quality: We evaluate estimate quality by using the mean-squared error at each estimate location:

$$e(x) = \mathrm{E}\left[\left(\phi(x) - \hat{\phi}(x)\right)^2\right], \tag{6}$$

where $\hat{\phi}(x)$ is the estimate of $\phi(x)$.

D. Best Linear Unbiased Estimator

The best linear unbiased estimator (BLUE) generates estimates using a weighted sum of the measurements, with spatially-dependent weights that minimize the mean-squared error at every location. VIM and BLUE are equivalent under certain circumstances (see [2], [6], [15], [16] and Appendix A).

Essentially, choosing the norm $f(\phi)$ in Equation (4) determines the spatial correlation function $R(x_1, x_2)$ in Equation (1) used to generate the equivalent BLUE estimate. By choosing the weights in Equation (4) according to

$$\mu_i = \frac{\gamma}{\epsilon_i}, \tag{7}$$

and satisfying the conditions in Appendix A, the VIM estimate matches the BLUE estimate computed using the spatial correlation function entailed by the norm $f(\phi)$.

For the norm $f(x)$ in Equation (5) and the domain $\Omega = \mathbb{R}^2$, the equivalent spatial correlation function is

$$R(x_1, x_2) = \frac{\|x_1 - x_2\|}{L} \mathbb{K}_1\left(\frac{\|x_1 - x_2\|}{L}\right), \tag{8}$$

where $\mathbb{K}_1$ is the first modified Bessel function of the second kind [8]. Figure 2 plots Equation (8) versus $\|x_1 - x_2\|$.

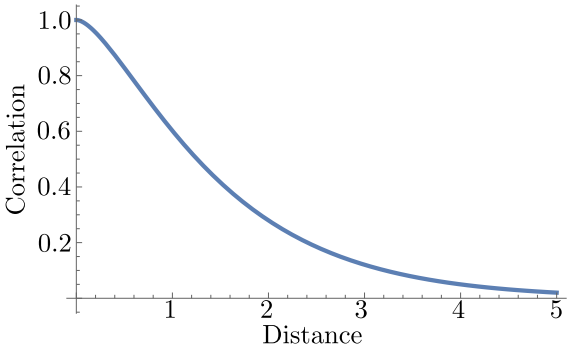

Fig. 2. Spatial correlation $R(x_1, x_2)$ (as given in Equation (8)) versus distance $\|x_1 - x_2\|$, with $L = 1$.

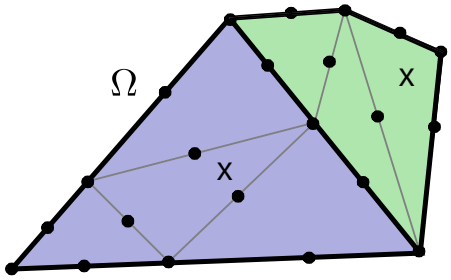

Fig. 3. Finite element mesh with seven triangular elements, divided between two robots. The elements composing the mesh meet only along full edges or at vertices. Robot locations are denoted "x". Dots represent nodes. Each robot's subdomain (blue and green shaded regions) consists of several triangular elements. Within each subdomain, the triangular elements compose a collective element. Thus, the mesh can be viewed as triangular, with each subdomain consisting of multiple triangular elements, or as polygonal, with each subdomain corresponding to a collective element.

E. Finite Element Method

We use the finite element method (FEM) to minimize the cost functional in Equation (4), as established in [6], [7], [9]–[12]. In FEM, the domain is partitioned into a mesh of N non-overlapping subdomains Ω_e called elements (see Figure 3). In a valid mesh, the subdomains intersect only at vertices or along full edges. Within each element, the field is approximated as a linear combination of m shape functions $w_j(x) \in \mathbb{R}$ and local nodal values $[q_e]_j \in \mathbb{R}^m$, $j = 1$ to m. Thus,

$$\phi_e(x) = w^T(x) q_e, \tag{9}$$

where $w(x) \in \mathbb{R}^m$ and $q_e \in \mathbb{R}^m$ are the shape functions and local node values stacked, respectively, into vectors.

The shape functions $w(x)$ are chosen in advance and the local node vectors q_e are unknown (see Appendix B).

Each nodal vector q_e is local to element e; however, nodes on adjacent element edges at the same location correspond to the same global nodal value. Generally, there are $M < mN$ nodes in the finite element mesh. The global node vector $q \in \mathbb{R}^M$ is related to the local node vectors by the gather matrices $B_e \in \mathbb{R}^{m \times M}$ such that

$$q_e = B_e q. \tag{10}$$

Each row of B_e contains exactly one nonzero element, equal to one, and each column of B_e has either zero or one nonzero element; thus every local nodal value corresponds to exactly one global nodal value.

By defining $\phi_e(x) = 0$ when $x \notin \Omega_e$, the estimate of the entire field is

$$\hat{\phi}(x) = \sum_{i=1}^{N} \phi_e(x). \tag{11}$$

We define the local load vector

$$g_e = \sum_{\{k : X_k \in \Omega_e\}} \mu_k w(X_k) z_k \tag{12}$$

and local stiffness matrix

$$\begin{aligned} K_e = \int_{\Omega_e} \Bigg[& \left(\frac{\partial^2 w}{\partial [x]_1^2}\right)\left(\frac{\partial^2 w}{\partial [x]_1^2}\right)^T + \left(\frac{\partial^2 w}{\partial [x]_2^2}\right)\left(\frac{\partial^2 w}{\partial [x]_2^2}\right)^T \\ & + 2\left(\frac{\partial^2 w}{\partial [x]_1 \partial [x]_2}\right)\left(\frac{\partial^2 w}{\partial [x]_1 \partial [x]_2}\right)^T \\ & + \frac{2}{L^2}\left(\frac{\partial w}{\partial [x]_1}\right)\left(\frac{\partial w}{\partial [x]_1}\right)^T + \frac{2}{L^2}\left(\frac{\partial w}{\partial [x]_2}\right)\left(\frac{\partial w}{\partial [x]_2}\right)^T \\ & + \frac{1}{L^4} w w^T \Bigg] d\Omega_e + \sum_{\{k : X_k \in \Omega_e\}} \mu_k w(X_k) w^T(X_k). \end{aligned} \tag{13}$$

Substituting Equations (11) to (13) into the cost functional (Equation (4)) and manipulating yields

$$J[\hat{\phi}] = \sum_{e=1}^{N} (q_e^T K_e q_e - 2 q_e^T g_e) + \sum_{j=1}^{n} \mu_j z_j^2. \tag{14}$$

Substituting Equation (10) into Equation (14) and minimizing with respect to q yields the global FEM equation

$$Kq = g, \tag{15}$$

with global stiffness matrix

$$K = \sum_{e=1}^{N} B_e^T K_e B_e \tag{16}$$

and global load vector

$$g = \sum_{e=1}^{N} B_e^T g_e. \tag{17}$$

The measurement values enter the FEM equations only through g, although the measurement positions affect both K and g.[2] Solving Equation (15) for q provides the field estimate.

Remark 4. *As in [7] we use natural (or Neumann) boundary conditions. Thus the derivatives of $\hat{\phi}(x)$ on the boundary of Ω and normal to it are zero and do not explicitly appear (see [3], [6], [7]). Using the Veubeke triangular elements described in Appendix B with these boundary conditions results in a well-posed FEM problem: see [4], [43], [44] for details.*

TABLE II
SYMBOL DEFINITIONS FOR SECTION III

Symbol	Description
$\Omega_i \subset \Omega$	Robot i's subdomain
$p \in \mathbb{R}^2$	Position in the field
$p_i \in \mathbb{R}^2$	Robot i's position
V_i	Robot i's Voronoi neighbors
T_{v}	Number of iterations for Voronoi coverage
τ_{ijk}	Delaunay triangle with vertices p_i, p_j, and p_k
$d_{ijk} \in \mathbb{R}$	Distance between circumcenter and incenter of τ_{ijk}
$R_{ijk} \in \mathbb{R}$	Circumradius of Delaunay triangle τ_{ijk}
$\bar{r}_{ijk} \in \mathbb{R}$	Inradius of Delaunay triangle τ_{ijk}
$A_{ijk} \in \mathbb{R}$	Area of Delaunay triangle τ_{ijk}
$d_{ij} \in \mathbb{R}$	Distance between robots i and j
τ_i	Set of sets $\{j,k\}$ such that τ_{ijk} is a Delaunay triangle
$G(p_1, \dots p_n)$	Cost function penalizing the Voronoi edge length
$\beta \in \mathbb{R}$	Step size for Voronoi edge expansion algorithm
T_{s}	Number of iterations for edge expansion algorithm
$\ell_{\max} \in \mathbb{R}$	Maximum side length for the FEM mesh

III. DEPLOYMENT

Table II summarizes the notation for this section. Our distributed variational inverse method (DVIM) allows robot groups to estimate the environment using VIM. Each robot communicates bidirectionally with neighbors and estimates only a subset of the environment, yet each local estimate converges to a piece of the global VIM estimate.

The algorithm consists of two basic tasks: deployment and estimation. Deployment consists of three stages: coverage, edge expansion, and meshing. During coverage, the robots spread out across the domain and establish their subdomains. The edge expansion algorithm improves the conditioning of the resulting FEM mesh. Finally, during the meshing phase, the robot establishes an FEM mesh in its own subdomain, that is a piece of a global FEM mesh. During estimation, a static condensation process is used, enabling each robot's mesh to be treated as a single element in a global FEM mesh.

After deployment, the robots take their measurements and then begin the estimation task. To estimate the environment, the robots solve the FEM problem using the alternating direction method of multipliers (ADMM).

For time-varying fields, the robots run the estimation process repeatedly, with newly acquired measurements. Robots can also patrol their own subdomains to obtain measurements at different locations (see e.g., [17]); however, for simplicity we assume they remain stationary after deployment. Occasionally, the robots may rerun the coverage algorithm using the estimation uncertainty or a user-specified interest function as a density function. This redeployment requires reforming the mesh. See Appendix A for how the DVIM algorithm for field estimation can also approximate the estimation uncertainty. Determining when to patrol the existing subdomain versus when to rerun the coverage algorithm is left for future work.

A. Coverage

During the coverage phase, each robot establishes a subdomain $\Omega_i \subset \Omega$. Collectively, the robots partition Ω into n non-overlapping subdomains that cover Ω.

[2]The terms *stiffness matrix* and *load vector* originate from solid mechanics, where FEM is used to analyze mechanical stress and strain.

We use Voronoi cells for the subdomains. The *Voronoi cell* for robot i consists of all points in Ω closer to robot i than to any other robot:

$$\{p \in \Omega : \|p - p_i\| \leq \|p - p_i\|, \text{ for all } i \neq j\}, \tag{18}$$

where p_i is the position of robot i. The collection of all such Voronoi cells is the *bounded Voronoi diagram* [45]. Two robots are *Voronoi neighbors* if their Voronoi cells share an edge.[3] The set of robot i's Voronoi neighbors is V_i.

If the robots can communicate with their Voronoi neighbors and know their absolute position and the domain boundary, the algorithms of [46], [47] allow distributed computation of the Voronoi cell.

Domain Coverage: To achieve subdomains with similar areas, the robots apply a Voronoi coverage algorithm based on centroidal Voronoi diagrams (CVD). In a CVD, the robots are positioned at the centroid of their Voronoi cells [45]. Upon initial deployment, the robots execute the coverage algorithm for T_{v} timesteps.

Remark 5. *The distribution of the robots can be weighted by a density function, allowing there to be more robots deployed in areas of greater uncertainty or interest. For simplicity, our implementation operates in a convex domain, using the algorithm of [48] with a uniform density. For non-convex regions the algorithm of [49] or [50] can achieve a similar configuration.*

Although the Voronoi cells of the CVD have similar areas (and thus each robot has approximately equal estimation responsibility), the cells in a CVD may (and often do) contain arbitrarily short edges [51]. Short edges lead to ill-conditioned FEM problems, making the estimation algorithm converge more slowly [52]. To avoid short edges, we introduce a distributed edge expansion algorithm that runs after the CVD algorithm.

B. Edge Expansion

Our edge expansion algorithm extends the work of [51] by decentralizing it. First, we require some definitions.

Definition 1. *The* circumcircle *of a triangle is the circle passing through all three triangle vertices, and its center is the* circumcenter.

Definition 2. *The* incircle *of a triangle is the largest circle that can be inscribed inside that triangle, and the* incenter *is the center of the incircle (see Figure 4).*

Definition 3. *The* Delaunay triangle τ_{ijk} *is the triangle whose vertices are at the positions of robots i, j, and k that are Voronoi neighbors of each other (i.e., $i \in V_j \cap V_k$, $j \in V_i \cap V_k$, and $k \in V_i \cap V_j$).*

Definition 4. *A Voronoi diagram where every Voronoi vertex is shared by exactly three Voronoi cells is* non-degenerate.

[3]The degenerate case of two regions intersecting at only a single vertex occurs with probability zero for randomly distributed robots [46]. Additionally, our deployment algorithm drives the robots away from such configurations.

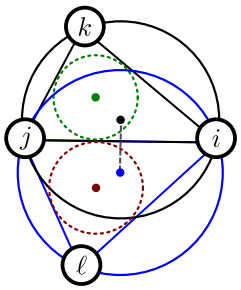

Fig. 4. Robots i, j, and k are Voronoi neighbors of each other and therefore are vertices of Delaunay triangle τ_{ijk} (black). Robots i, j, and ℓ are Voronoi neighbors of each other and therefore are vertices of Delaunay triangle $\tau_{ij\ell}$ (blue). The circumcircles (solid) and incircles (dashed) are also shown. The black and blue dots (connected by the dashed line), are the circumcenters of Delaunay triangles, and are therefore Voronoi vertices. The gray dashed line is therefore a Voronoi edge. The incircles are denoted by the dashed circles. As the circumcenters move toward the incenters, the length of the Voronoi edge between them increases.

The vertex shared by regions Ω_i, Ω_j, and Ω_k is the circumcenter of Delaunay triangle τ_{ijk} [45]. Therefore, a short Voronoi edge occurs when two Delaunay triangle circumcenters are near each other. To avoid this circumstance, the method of [51] minimizes the distance d_{ijk} between every Delaunay triangle's circumcenter and incenter (see Figure 4).

Let $\bar{R}_{ijk}$ be the circumradius, $\bar{r}_{ijk}$ be the inradius, and A_{ijk} be the area of Delaunay triangle τ_{ijk}. Let d_{ij} be the Euclidean distance between robots i and j. Then, as per [51],

$$d_{ijk}^2 = \bar{R}_{ijk}(\bar{R}_{ijk} - 2\bar{r}_{ijk}), \tag{19}$$

with

$$\bar{R}_{ijk} = \frac{d_{ij}d_{ik}d_{jk}}{4A_{ijk}} \tag{20}$$

and

$$\bar{r}_{ijk} = \frac{2A_{ijk}}{d_{ij} + d_{ik} + d_{jk}}. \tag{21}$$

Let τ_i be the set of two-element sets $\{j, k\}$ such that τ_{ijk} is a Delaunay triangle. A cost function penalizing short distances between Delaunay incenters and circumcenters is (see [51])

$$G(p_1, \ldots, p_n) = \frac{1}{6}\sum_{i=1}^{n}\sum_{\{j,k\}\in\tau_i} \bar{R}_{ijk}(\bar{R}_{ijk} - 2\bar{r}_{ijk}). \tag{22}$$

Let $p^\perp = (-[p]_2, [p]_1)$ be the rotation of point p by $90°$ about $(0, 0)$. Noting that every Delaunay triangle appears in Equation (22) three times, the gradient of $G(\ldots)$ with respect to the robot positions is[4]

$$\frac{\partial G}{\partial p_i} = \sum_{(j,k)\in\tau_i}\left[(\bar{R}_{ijk} - \bar{r}_{ijk})\frac{\partial \bar{R}_{ijk}}{\partial p_i} - \bar{R}_{ijk}\frac{\partial \bar{r}_{ijk}}{\partial p_i}\right], \tag{23}$$

where

$$\frac{\partial \bar{R}_{ijk}}{\partial p_i} = \bar{R}_{ijk}\left[\frac{1}{d_{ij}^2}(p_i - p_j) + \frac{1}{d_{ik}^2}(p_i - p_k) - \frac{1}{2A_{ijk}}(p_k - p_j)^\perp\right], \tag{24}$$

[4]Each Delaunay triangle appears three times because $\{j, k\} \in \tau_i$ implies that $\{i, k\} \in \tau_j$ and $\{i, j\} \in \tau_k$.

and

$$\frac{\partial \bar{r}_{ijk}}{\partial p_i} = \frac{-2}{(d_{ij} + d_{ik} + d_{jk})^2}\left[\frac{A_{ijk}}{d_{ij}}(p_i - p_j) + \frac{A_{ijk}}{d_{ik}^2}(p_i - p_k) - \frac{d_{ij} + d_{ik} + d_{jk}}{2}(p_k - p_j)^\perp\right]. \tag{25}$$

Remark 6. *As written here, the derivative in Equation* (23) *depends only on the positions of robot i's Voronoi neighbors, rather than, as derived in [51], depending on every Delaunay triangle. Writing the derivative in this form allows the robots to compute the gradient without global information.*

To compute Equation (23), the robots send their positions to their neighbors, compute their Voronoi neighbors, determine the side lengths of neighboring Delaunay triangles, and compute $\frac{\partial G}{\partial p_i}$.

In our implementation (see Algorithm 1), generally the robots move according to the gradient descent update rule

$$p_i^{k+1} = p_i^k - \beta\frac{\partial G}{\partial p_i}, \tag{26}$$

where $\beta \in \mathbb{R}$ is a fixed step size. After all the robots move according to Equation (26) for T_s steps, they stop and begin the meshing phase of the initialization task.

Edge Expansion Control Law Adjustments: There are two adjustments that we make to the control law, both involving the boundary of Ω. To avoid leaving the domain, each robot uses Equation (26) to predict its next position; if that position lies outside Ω, it halves the step-size and recomputes the control law. This process is repeated until p_i^{k+1} is inside Ω.

The second adjustment depends on the edges of the robot's Voronoi cell Ω_i. For every edge of Ω_i that coincides with an edge of the global domain Ω, the gradient $\frac{\partial G}{\partial p_i}$ is projected onto a vector pointing in the direction of the edge. This adjustment compensates for the lack of fixed Delaunay vertices on the boundary of Ω, as assumed in [51].

C. Meshing

After deployment, each robot forms an FEM mesh consistent with its neighbors' meshes, so that the union of each robot's mesh forms a valid FEM mesh (i.e., the elements cover Ω and only intersect along full edges or at vertices).

Algorithm 2 outlines the meshing process. To ensure a well-conditioned mesh, the meshing algorithm should produce triangles without small angles or long edges. We use the algorithm of [53], which ensures that all triangles have a smallest angle larger than $20.7°$ (this constraint is locally violated if the boundary of Ω has angles smaller than $20.7°$). We also set the algorithm so that no edge length exceeds $\ell_{\max} = \frac{L}{3}$ (as per [7]).

To ensure a consistent FEM mesh, robots that share edges must place any mesh nodes on those shared edges at the same position as their neighbors' node placements. Therefore, prior to meshing, every robot divides its edges into the minimum number of segments such that no segment exceeds $\ell_{\max}$. The nodes subdividing these edges will be nodes in the resulting

Algorithm 1 Distributed Coverage and Edge Expansion

Input:
The number of Centroidal Voronoi iterations T_{v}
The number of Edge Expansion iterations T_{s}
The Edge Expansion step size β

repeat ▷ The centroidal Voronoi algorithm (e.g., [48])
  Transmit p_i
  Receive p_j from communication neighbors
  Determine Voronoi neighbors and compute Ω_i
  Move to centroid of Ω_i
until T_{v} iterations have occurred
repeat ▷ Edge Expansion
  Transmit p_i
  Receive p_j from communication neighbors
  Determine Voronoi neighbors
  Compute $\frac{\partial G}{\partial p_i}$ using Equation (23)
  Project $\frac{\partial G}{\partial p_i}$ onto any Ω_i edges shared with Ω
  while p_i^{k+1} (from Equation (26)) not in Ω **do**
    $\beta \to \beta/2$
  end while
  Move according to Equation (26) using the
  updated step size and projected gradient
until T_{s} iterations have occurred

mesh, and, due to symmetry, they will be at the same absolute location for adjacent robots.

The robots then run the algorithm of [53] over their own domains to mesh the interior. This algorithm guarantees that the nodes previously added to subdivide the boundary of the Voronoi cell Ω_i are included in the mesh. However, the algorithm may also add additional nodes to the boundary. The robots remove these vertices so that only those vertices created during the edge subdivision process are shared, resulting in a consistent mesh between neighbors.

Algorithm 2 Distributed Meshing

Input: Voronoi cell Ω_i and the maximum edge length $\ell_{\max}$

Subdivide the edges of Ω_i, as follows:
for edge $e_i \in \Omega_i$ **do**
  $\ell \leftarrow$ length of edge e_i
  $n_{\mathrm{segs}} \leftarrow \lceil \frac{\ell}{\ell_{\max}} \rceil$ ▷ Number of subdivided segments
  $\ell_{\mathrm{act}} \leftarrow \frac{\ell}{n_{\mathrm{segs}}}$ ▷ Length of each subdivided segment
  Add nodes to e_i so edge e_i has n_{segs} of length ℓ_{act}
end for
Mesh the subdivided Ω_i, using the algorithm of [53]
Remove nodes added to edges of Ω_i by algorithm of [53]

IV. Distributed Estimation

To perform distributed estimation, the robots must assemble their local finite element meshes and then execute a distributed optimization algorithm to solve for their estimate. Table III summarizes the notation for this section.

TABLE III
Symbol definitions for Section IV

Symbol	Description
m_i	Number of nodes on robot i's domain boundary
$\widetilde{M}$	Total number of all robots' nodes
$\widetilde{q} \in \mathbb{R}^{\widetilde{M}}$	All robot node vectors, stacked
$Q \in \mathbb{R}^{\widetilde{M} \times \widetilde{M}}$	ADMM preconditioning matrix
$\mathcal{N}_\ell$	Index set containing local indexes into expanded node vector
$\mathcal{M}_i$	Index set local to robot i
$\hat{q}_i \in \mathbb{R}^{m_i}$	Robot i's primal vector
$\nu_i \in \mathbb{R}^{m_i}$	Robot i's dual vector
$u \in \mathbb{R}^{m_i}$	Robot i's Lagrange multiplier vector
$\rho \in \mathbb{R}$	ADMM optimization parameter
$\alpha \in \mathbb{R}$	ADMM relaxation paramter
T_{a}	Number of steps to run the ADMM algorithm
$\bar{\nu}_i \in \mathbb{R}^{m_i}$	Dual vector sub-calculation that robot i transmits

A. Local FEM Assembly

To assemble its local FEM equations, each robot follows the procedure outlined in Section II-E. However, instead of global quantities, the robots use their local domains, local meshes (from the deployment step), and local measurement information. Each robot eliminates nodes that are not shared with other robots (i.e., the nodes not on the boundary of Ω_i) using static condensation (see Appendix B).

This assembly procedure requires no communication and effectively results in each robot's domain corresponding to a single element in a global finite element mesh. Thus each robot has a local node vector $q_i \in \mathbb{R}^{m_i}$, local stiffness matrix $K_i \in \mathbb{R}^{m_i \times m_i}$, and local load vector $g_i \in \mathbb{R}^{m_i}$, where m_i is the number of nodes on the boundary of robot i's domain.

B. Distributed ADMM

After each robot has determined its own stiffness matrix and node vector, it uses the distributed alternating directions method of multipliers (ADMM) to solve the FEM problem. This constrained optimization method enables each robot to determine the local node vector q_i, which constitutes a piece of the solution to the global FEM problem.

To use ADMM, we first group the nodal values into three vectors: the global node vector $q \in \mathbb{R}^M$, the local node vector $q_i \in \mathbb{R}^{m_i}$, and the expanded node vector $\widetilde{q} = [q_1, \dots q_N] \in \mathbb{R}^{\widetilde{M}}$, where $\widetilde{M} = \sum_{i=1}^{N} m_i$ is the total number of local nodes.

As in regular FEM, the local vectors are related to the global vector by the gather matrix: $q_i = B_i q$. The expanded node vector is formed by concatenating a copy of each robot's local node vectors and can be expressed as $\widetilde{q} = Bq$, where $B = [B_1^T, \dots, B_N^T]^T \in \mathbb{R}^{\widetilde{M} \times M}$ is the global gather matrix.

The FEM problem is a constrained quadratic program:

$$\begin{aligned} &\underset{q_1 \dots q_N}{\arg\min} \sum_{i=1}^{N} (q_i^T K_i q_i - 2 g_i^T q_i) \\ &\text{subject to } Q\widetilde{q} = QBq, \end{aligned} \tag{27}$$

where $Q \in \mathbb{R}^{\widetilde{M} \times \widetilde{M}}$ is an invertible diagonal pre-conditioning matrix used to improve convergence.

Although the cost function of Equation (27) is decoupled across robots, the constraint forces every robot's local copy

of a nodal value to match its correct global counterpart. This correspondence can be expressed as a collection of index sets

$$\mathcal{N}_\ell = \{j : [\widetilde{q}]_\ell \sim [q]_k \text{ implies } [\widetilde{q}]_j \sim [q]_k\} \text{ for } \ell = 1 \text{ to } \widetilde{M},$$

where $[\widetilde{q}]_\ell \sim [q]_k$ indicates that the two nodal values correspond to the same node. Let $\mathcal{M}_i$ be the set of local indices for robot i's nodes. Because $\widetilde{q}$ is partitioned across robots, each robot can (conceptually) permute $\widetilde{q}$, making its own local node vector q_i first (i.e., in implementation each robot uses $\mathcal{M}_i = \{1 \text{ to } m_i\}$). The robots find $\mathcal{N}_\ell$ for each of their own nodes by communicating with their Voronoi neighbors. Each robot transmits its local node locations and indices. Nodes at the same location are then grouped together.

To solve Equation (27) we use the pre-conditioned ADMM method of [14]. ADMM solves Equation (27) by repeating three steps: a primal update (with local primal vector $\hat{q}_i \in \mathbb{R}^{m_i}$), a dual update (with local dual vector $\nu_i \in \mathbb{R}^{m_i}$), and a Lagrange multiplier update (with local Lagrange multiplier vector $u_i \in \mathbb{R}^{m_i}$). Only the dual update requires communication.

The ADMM iterations for robot i are below, where superscripts with k indicate iteration number:

$$\hat{q}_i^{k+1} = (K_i + \rho Q_i^T Q_i)^{-1}(g_i + \rho Q_i^T (Q_i \nu_i^k - u_i^k)), \quad (28)$$

$$[\nu]_\ell^{k+1} = \frac{1}{\sum_{j \in \mathcal{N}_\ell} [Q]_{jj}^2} \sum_{j \in \mathcal{N}_\ell} [\bar{\nu}]_j^k, \text{ for } \ell \in \mathcal{M}_i, \quad (29)$$

$$u_i^{k+1} = u_i^k + \alpha Q_i \hat{q}_i^{k+1} + (1-\alpha) Q_i \nu_i^k - Q_i \nu_i^{k+1}. \quad (30)$$

Here, $\rho > 0$ and $0 < \alpha < 2$ are optimization parameters, the global dual variable is $\nu \in \mathbb{R}^M$, and

$$[\bar{\nu}]_j^k = [Q]_{jj}(\alpha [Q]_{jj} [\hat{q}]_j^{k+1} + (1-\alpha)[Q]_{jj}[\nu]_j^k + [u]_j^k). \quad (31)$$

When $Q = I$ and $\alpha = 1$, the pre-conditioned overrelaxed ADMM becomes regular ADMM [13].

Each robot computes Equations (28), (30) and (31) locally and then transmits its local vector $\bar{\nu}_i \in \mathbb{R}^{m_i}$ to its neighbors. Using the index set $\mathcal{N}_\ell$ and the received $\bar{\nu}$ vectors, each robot computes Equation (29). The ADMM steps are repeated for T_a time steps (see Algorithm 3).

C. Communication Analysis

If all robots communicate with their Voronoi neighbors, they can implement Equation (29) because two robots that are not Voronoi neighbors do not contribute to each others' relevant dual variable ν entries. After their Voronoi neighbors are established, the robots ignore packets from non-neighbors. Communication between the robots occurs in three stages.

The first stage happens only once per meshing operation, and lets the robots determine $\mathcal{N}_\ell$, the correspondence between their nodes and those of its neighbors. Each robot transmits a vector whose length is proportional to m_i, the number of nodes on robot i's subdomain boundary.

The second communication step occurs once per ADMM run: the robots transmit the diagonal pre-conditioning matrix Q_i transmitting a vector whose length is proportional to m_i.

Algorithm 3 Distributed ADMM

Input:
The number of ADMM iterations T_a
Set $\mathcal{N}_\ell$ of nodes robot i shares with its Voronoi neighbors
Local measurements z_i and their locations X_i.
ADMM parameters ρ and α

Output: Estimate of local node states q_i

```
Compute measurement weights μ_i
Local FEM assembly to find K_i and g_i
Transmit pre-conditioning entries, the diagonal of Q_i
Receive pre-conditioning entries from Voronoi neighbors
repeat
    Update the primal variables q̂_i using Equation (28)
    Compute [ν̄]_j for each local node j with Equation (31)
    Transmit [ν̄]_j for each local node j
    Receive ν̄ from each neighbor ℓ ∈ V_i
    Compute the dual update using Equation (29)
    Update the Lagrange multipliers using Equation (30)
until has executed T_a iterations
return The primal (i.e., local node) values q̂_i
```

Finally, every ADMM iteration requires every robot to transmit $\bar{\nu}_i \in \mathbb{R}^{m_i}$. This step dominates the communication. The size of $\bar{\nu}_i$ depends on the number of nodes on the boundary of each robot's FEM mesh, so on each iteration each robot sends a packet containing on the order of m_i numbers.

D. Convergence and Parameter Selection

Convergence of the ADMM algorithm is guaranteed; however, determining when convergence occurs requires computing residuals that depend on global information [13]. To avoid this computation, the robots run ADMM for a fixed number of iterations. To track time-varying environments, ADMM is restarted with new measurements and previous measurements, weighted according to a time covariance function (see [17]). Measurements with weights below a threshold are discarded.

Remark 7. *To track time-varying fields, we assume that the timescale of environmental variation is slower than that of algorithm convergence. This restriction also applies to consensus-based estimation methods such as [22], [29].*

Remark 8. *We assume each robot knows its absolute position (obtained, for example, via GPS). Inaccuracy in this knowledge manifests itself as noise in the estimate. The effect of such noise on DVIM is the same as that of VIM because the distributed estimate converges to the centralized estimate.*

The convergence rate of ADMM depends on its parameters and (indirectly) on the condition number of K. In our simulations, improving the conditioning of K by adjusting the FEM mesh (using the edge expansion algorithm proposed in Section III-B) has ensured consistent performance of the algorithm across a range of scenarios (see Section VI).

Convergence analysis and the selection of ρ, α, and Q for a system where each robot estimates the same vector are discussed in [14]. Although we use this algorithm, the analysis techniques of [14] do not directly apply because, with

our communication structure and distribution of the solution vector across the robots, we cannot satisfy a key assumption. In particular, [14] assumes that $Q_i^T Q_i = K_i$ for $i = 1, \ldots N$; however, choosing Q_i to satisfy this assumption is equivalent to solving the global FEM problem. This equivalence occurs because distributing Equation (29) requires the distributed computation of $(B^T Q^T Q B)^{-1}$ which, when $Q_i^T Q_i = K_i$, becomes the same as inverting the global stiffness matrix.

Instead of enforcing $Q_i^T Q_i = K_i$, we set the diagonal elements of Q_i to be the square root of the diagonal elements of K_i (i.e., $[Q]_{jj} = \sqrt{[K]_{jj}}$). Thus $Q_i^T Q_i$ approximates the pre-conditioning assumption from [14]. In our experience, this pre-conditioning has greatly improved the performance of the ADMM algorithm. The parameters ρ and α are manually tuned via simulation; however, as discussed in Section VI, parameters tuned for one small estimation problem have worked well across a range of estimation scenarios.

E. Timing

The deployment task occurs for a fixed number of timesteps. The meshing task requires only one timestep. Then the estimation task occurs for a fixed number of timesteps and can be repeated with new measurements periodically.

Conceptually, the timing and coordination of these tasks depends on a synchronized global clock. The transitions between the various tasks depends on running each task and switching between tasks simultaneously. Implementations can avoid needing a global clock by using the following scheme.

Every algorithm requires a single communication step per iteration. The robots first transmit local information to their neighbors and then wait to receive information from their neighbors. When a robot reaches the fixed number of timesteps for a task (according to its internal count), it starts the next task. The first communication step in the next task is always a transmission and the information required to compute the packet is available locally. Receiving robots detect when the received packet matches the packet type for the next task. Upon receiving such a packet, those robots immediately transition to the next task as well. This task preemption spreads across the network until all the robots perform the same task.

V. Performance Evaluation

We analyze three aspects of our algorithm's performance: communication topology, memory usage, and total communication. All three of these factors directly contribute to the cost and energy usage of the robots. These performance measures depend on the correlation length scale of the environment and the number of robots used to measure it.

We also compare DVIM to consensus-based approaches (e.g., [22]), where all robots agree on a full environment estimate, and approaches with a distributed model, (e.g., [26], [28], [29]), where each robot estimates a subset of the environment. Of the methods with a distributed environmental representation, we focus on the Kriged Kalman filter of [29] because it is the most similar to DVIM.[5] In particular, one step in the distributed Kriged Kalman filter of [29] uses Jacobi over-relaxation to produce a BLUE estimate; thus our methods incorporate spatial uncertainty similarly, due to the equivalence of VIM and BLUE (see Appendix A). We refer to this step of the distributed Kriged Kalman filter as DBLUE and use it for our comparison.

To ease comparison, we adopt a radius-limited communication model, like the model used in [29]: every robot communicates with talks to every other robot within a radius R of itself and no other robots. We also assume that the environment is a spatial random field with correlation length L and correlation function given by Equation (8). Our analysis considers the performance as the correlation length and the density of the robots varies.

All distance and area measurements are normalized by the longest possible distance between any two points in the domain. For example, a communication radius of 1 means that every robot communicates with every other robot. In all the examples, the variance of the sensor noise is $\epsilon_i = .01$ and the robots' initial positions are uniformly randomly distributed (so the robots do not undergo the deployment phase).

A. Communication Topology

Here we examine the communication radius required by DVIM and DBLUE to generate an estimate, at different correlation lengths and robot densities. We omit consensus-based methods from the comparison because they typically apply to any connected network, a weaker requirement than DVIM or DBLUE. A small communication radius results in lower transmission power and reduces interference between robots transmitting simultaneously.

DBLUE Communication Radius: In DBLUE, the robots' subdomains are discs of radius $\widetilde{R}$, centered at each robot's location. Unlike in DVIM, in DBLUE the subdomains might overlap and might not cover the whole domain. To compare DVIM and DBLUE, we assume that robots cover equally sized subdomains: area covered per robot is $\frac{1}{D}$, where D is the robot density (the number of robots divided by the domain's area).

For a robot running DBLUE to estimate all points within a radius $\widetilde{R}$, it must know all measurements within a radius of $\widetilde{R} + \bar{\rho}L$, where $\bar{\rho} > 0$ sets a distance beyond which the correlation of the measurement and estimate location becomes negligible. Obtaining these measurements (with one-hop communication) requires a communication radius of $\widetilde{R} + \bar{\rho}L$. Since each robot covers an area of $\frac{1}{D} = \pi \widetilde{R}^2$, for a given robot density the communication radius required by DBLUE is

$$R = \sqrt{\frac{1}{D\pi}} + \bar{\rho}L. \tag{32}$$

The required radius decreases slightly with increasing robot density because robots estimate smaller regions; however, for high density the effect of the correlation length dominates.

DVIM Communication Radius: In DVIM, the communication radius must be large enough such that every robot communicates with its bounded Voronoi neighbors. This radius is independent of the environmental correlation length.

[5] Technically, [29] also has a consensus-based component using global basis functions; however, the BLUE portion of the estimate is computed separately from the consensus-based part.

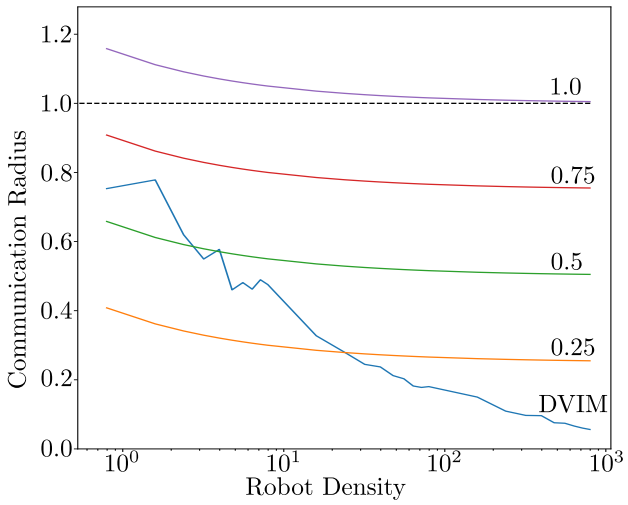

Fig. 5. Minimum required communication radius versus robot density, with distances normalized by the diameter of the circular domain. Curves are labeled with the value of L (for DBLUE) or with DVIM (there is only one DVIM curve because the DVIM communication radius is independent of L). Communication radii above the dashed line result in all-to-all communication.

We determine the communication radius for a given robot density empirically. We randomly place robots in the unit circle and compute the longest distance between any robot and its Voronoi neighbor: this distance is the minimal communication required for every robot to see its Voronoi neighbors in the given example. For each fixed robot density we repeat the test for 100 trials and use the longest radius encountered as the minimal required radius for that robot density.

DVIM and DBLUE Comparison: Figure 5 plots the minimum required communication radius versus density for DVIM and DBLUE at different correlation lengths, using $\bar{\rho} = 1$. The results indicate that, although the required communication radius decreases with density for both DVIM and DBLUE, in DBLUE this effect is overwhelmed by the dependence on the correlation length.

For DBLUE, the decrease in communication radius with density is from each robot estimating a region of decreasing size. The communication radius eventually reaches a minimum related to the correlation length. For DVIM, the required radius approaches zero as density increases—regardless of correlation length—because the bounded Voronoi neighbors of each robot become closer together.

For environments with longer correlation lengths, DVIM requires a significantly smaller communication radius than DBLUE. Our analysis is conservative and actually understates the required communication radius for DBLUE because with $\bar{\rho} = 1$ the correlation between the estimate and measurement considered negligible is anything less than $\mathbb{K}_1(1) = 0.60$, which is not negligible in most applications.

B. Memory

To measure memory usage, we estimate the number of numbers each robot must store to produce an estimate anywhere within its subdomain. Consensus-based methods always require more memory than DVIM or DBLUE because every robot stores a global representation of the environment which, for an equivalent estimate, is always a superset of the local environment representations used in DVIM or DBLUE. Although it uses more memory, such a global representation means that every robot can estimate the environment anywhere, whereas methods like DVIM or DBLUE require a query system (see e.g., [42]) if an operator wants to obtain an environmental estimate at any location from any robot.

DBLUE Memory Usage: In DBLUE, every robot stores every measurement and measurement location within R to compute the estimate; therefore, the memory usage increases with robot density and the number of measurements per robot. Assuming one measurement per robot, there are $D\pi R^2$ measurements and measurement locations that every DBLUE robot must store. Substituting Equation (32) for R reveals that the number of measurements each DBLUE robot stores is approximately

$$D\pi \left(\sqrt{\frac{1}{D\pi}} + \rho L \right)^2 . \tag{33}$$

DVIM Memory Usage: For DVIM, the amount of memory used per robot depends on the number of nodes in each robot's mesh. Each robot's mesh generates triangles with side lengths less than $\ell_{\max} = \frac{L}{3}$ and 12 nodes (See Appendix B). An equilateral triangle with side length $\frac{L}{3}$ has area $\frac{\sqrt{3}}{4}L^2$ so, not accounting for overlapping nodes (which reduces memory requirements), there are $12\frac{1}{D}\frac{\sqrt{3}}{4}L^2$ nodes per robot.

With high robot density, each robot's subdomain may have smaller area than a single $\frac{L}{3}$ equilateral triangle. In this case, we assume that the robot's subdomain is hexagonal and divided into 4 triangles.[6] Accounting for overlapping nodes in this case the minimum number of nodes per robot is 27.

DVIM and DBLUE Comparison: Figure 6 shows the memory usage per robot for DVIM and DBLUE. As robot density increases, the amount of memory required for DBLUE increases without bound because (aside from practical limitations) there can always be more robots added within the relevant radius. Memory usage for DVIM decreases with density until each robot's subdomain contains the minimum number of nodes due to it being smaller than an equilateral triangle with side length $\frac{L}{3}$.

C. Fixed Communication Radius

At a given communication radius, the DVIM curve in Figure 5 shows the approximate minimum density needed by DVIM to meet the Voronoi neighbor communication assumption. At this same density value (and depending on the correlation length), DBLUE may require a larger communication radius than DVIM to incorporate all relevant measurements and form a proper estimate. As the density increases above this minimum, the errors in the DBLUE and DVIM estimate decrease because there are more measurements; however, DBLUE's per robot memory usage grows unbounded whereas DVIM's per robot memory usage decreases to a fixed value.

[6]The average number of Voronoi neighbors is less than or equal to 6 [45].

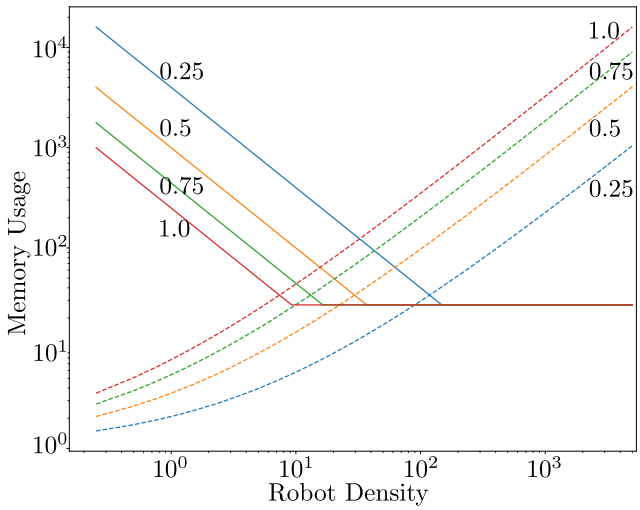

Fig. 6. Memory usage per robot versus robot density for DVIM and DBLUE estimators. Solid lines are for DVIM, dashed are for DBLUE; curve labels provide the L value. As density increases the area covered per robot decreases, reducing the memory requirements for DVIM. In DBLUE, memory usage is dominated by the correlation length; as density increases there are more measurements within the relevant distance of each robot, leading to increased memory usage.

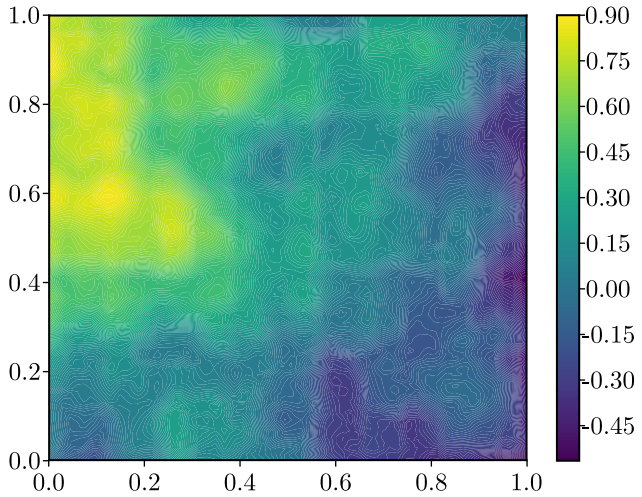

Fig. 7. The actual environment for a trial with 60 robots and normalized correlation length of 0.7.

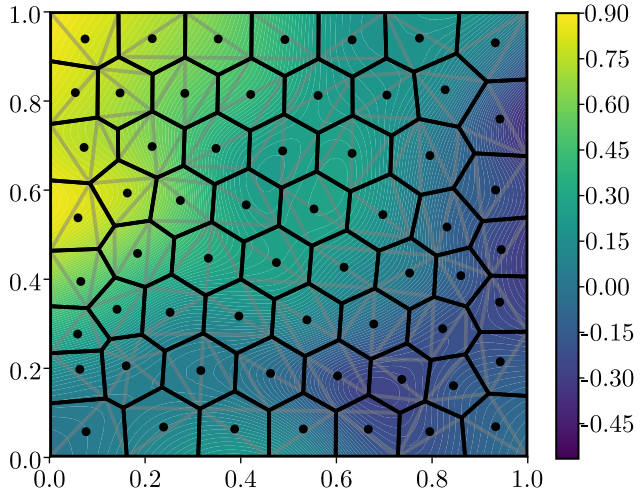

Fig. 8. The local estimates and Voronoi cells for a trial with 60 robots and normalized correlation length of 0.7.

VI. Simulations

In this section we validate our approach using agent-based simulations over a wide variety of conditions. We also analyze the total communication required by DVIM and compare it to the consensus-based method of [22].

We performed trials over a uniform grid of parameters: the number of robots varied between 20 and 120 in increments of one and the normalized correlation length was between 0.4 and 1.0 in increments of 0.05, for a total of 1313 trials. The normalized communication radius of 0.6 was sufficient for every robot to talk to its Voronoi neighbors at every trial. The domain Ω was the unit square centered at $(0.5, 0.5)$. The robots' initial positions were uniformly randomly distributed in a subset of Ω (a square with a vertex at the origin and side length of 0.0625).

The environment was a zero-mean Gaussian process with unit background variance ($\gamma = 1$) and correlation given by Equation (8). We generated a new random environment for each trial by approximating the spatial random process over a 200×200 grid with the circular embedding algorithm of [54]. We used bilinear interpolation to get the field value between grid points. Each robot had a sensor variance of 0.01, and the maximum triangle length was $L/3$.

The robots ran Lloyd's algorithm for $T_\mathrm{v} = 1000$ iterations and then the expansion algorithm for $T_\mathrm{s} = 100$ iterations, with a nominal step size of $\beta = 0.5$. There were $T_\mathrm{a} = 300$ ADMM iterations, with parameters $\rho = 0.1$ and $\alpha = 1.8$. We hand tuned the ADMM parameters using a separate simulation involving only four robots, so they were not specifically tuned for the situations tested to generate our results.

Figure 7 shows the actual field and Figure 8 shows the estimate for a trial with 60 robots and a normalized correlation length of 0.7. Let q_L be the merged vector of local solutions after T_a ADMM iterations and q_G be the global solution vector, solved on a mesh composed of the individual robot's meshes. The relative convergence error is $\frac{\|q_\mathrm{L} - q_\mathrm{G}\|}{\|q_\mathrm{G}\|}$.

Across all the trials, after 300 ADMM iterations, the best relative convergence error between the global and local solutions was on the order of 10^{-12}, while the mean relative error was 10^{-9} and the worst error was 10^{-6} (see Figure 9).

The number of robots n and the correlation length L relate to convergence through their effect on the FEM mesh geometry and therefore the conditioning of the problem. As n increases, the area each robot covers is smaller (for fixed Ω). As L increases, the maximum triangle side length used by the robots increases. The general trend is that convergence slows slightly as more robots are added and with higher correlation lengths (see Figure 10). Lower error can be achieved by running the ADMM estimator for longer or adjusting the ADMM parameters for a narrower range of scenarios.

We also compare DVIM to the consensus-based method of [22] using the equivalent timestep value, which measures how many average consensus iterations the swarm can conduct

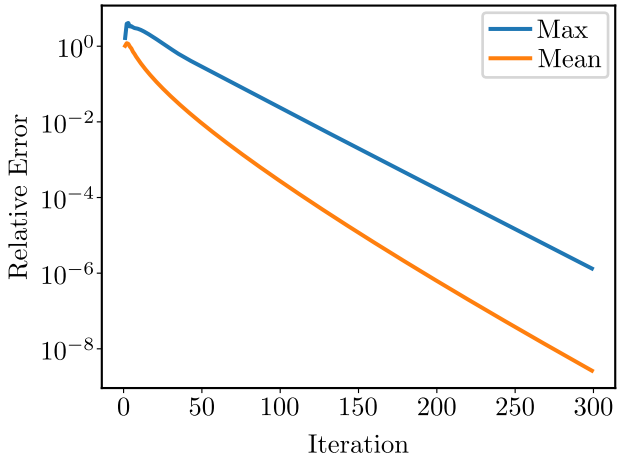

Fig. 9. Mean and maximum relative error versus ADMM iteration, across all trials.

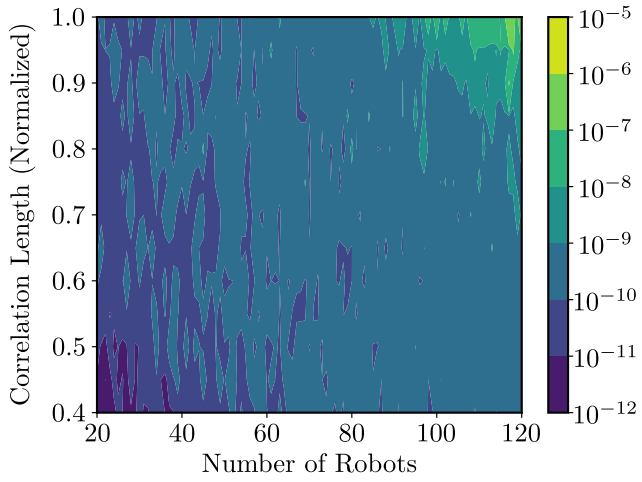

Fig. 10. Relative error (after 300 iterations) for each trial versus number of robots and correlation length, on a logarithmic scale. Robot density and the number of robots are the same here because the domain has unit area.

before the total number of numbers transmitted by the swarm equals the number of numbers transmitted by DVIM: $\frac{V_t T_a}{Mn}$, where V_t is the total number of numbers the DVIM robots transmit per timestep, T_a is the number of ADMM timesteps, and M is the total number of states. The equivalent timestep value assumes that the consensus-based method uses the same number of states as DVIM and that each consensus estimator transmits only one value per state.

In our trials, the total number of values transmitted was, at worst, equivalent to what would be transmitted after 24 consensus iterations and, at best, equivalent to five consensus iterations. For comparison, in [19], it takes an average consensus estimator approximately 50 timesteps to converge for a similar network. This performance suggests that DVIM can significantly reduce the total communication cost of environmental estimation versus consensus-based methods.

VII. Conclusion

We have introduced distributed VIM (DVIM), which uses distributed FEM and ADMM to estimate an environment. Although the estimates match a centralized solution, no single agent has a full environment representation, allowing our method to work even when the environment does not fit in a single computer's memory. Unlike existing methods, adding more robots to a given area decreases the memory used per robot and the required communication radius, and reduces the total communication needed. Future work includes removing the need for computing an FEM mesh by using mesh-free FEM methods and incorporating a model of the field's time variation rather than repeatedly estimating static time-slices.

Appendix A

We compare two approaches to environmental estimation: the best linear unbiased estimator (BLUE) and the variational inverse method (VIM). BLUE minimizes the expected error of the estimate, while VIM finds a smooth field close to the measurements. As established in [15], [16], these two seemingly different methods are closely related and even equivalent under some conditions. The analysis in this section relies heavily on the work of [2], [15], [16].

A. Best Linear Unbiased Estimator

To derive BLUE, we first write the estimate $\hat{\phi}(x)$ of the field $\phi(x)$ as a linear combination of the measurements:

$$\hat{\phi}(x) = u^T(x)z, \tag{34}$$

where $u(x) \in \mathbb{R}^N$ is a weight vector whose weights depend on the desired estimate position. The goal of BLUE is to determine the weights $u(x)$ that minimize the mean-square error given by Equation (6). Proposition 1 describes the BLUE in terms of the functional $\mathcal{H}$ that evaluates the field at the measurement locations.

Proposition 1. *For a zero-mean spatial random field with covariance $C(x_1, x_2)$, the BLUE is*

$$\widehat{\phi}(x) = F^T(x)G^{-1}z, \tag{35}$$

where

$$F(x) = \mathcal{H}_{(x_1)}C(x, x_1) \tag{36}$$

is the covariance vector $\mathrm{E}(\phi(x)z)$ between each measurement and the field at position x and where

$$G = \mathrm{E}(zz^T) = (\mathcal{H}_{(x_1)}\mathcal{H}^T_{(x_2)})C(x_1, x_2) + \epsilon \tag{37}$$

is the measurement covariance matrix.

The minimal expected error of the estimate is

$$e(x) = C(x, x) - F(x)^T G^{-1} F(x). \tag{38}$$

Proof. See [2]. □

B. Variational Inverse Method

We now write the solution to VIM in terms of functionals and inner products. For VIM to be well posed, the norm in Equation (4) must be the norm for a reproducing kernel Hilbert space (RKHS) $\mathbb{W}$. Therefore, we implicitly assume that the field $\phi \in \mathbb{W}$ and the estimate $\hat{\phi} \in \mathbb{W}$.

Since $\mathbb{W}$ is an RKHS, there exists, by the Riesz representation theorem, a representer function $r_x \in \mathbb{W}$ such that

$\langle r_x, \phi \rangle = \mathcal{L}_x[\phi] = \phi(x)$, where $\langle \cdot, \cdot \rangle$ is the inner product and $\mathcal{L}_x$ is the evaluation functional associated with $\mathbb{W}$ [15].

Every RKHS has an associated positive-definite function called the reproducing kernel, given by $\langle r_{x_1}, r_{x_2} \rangle = K(x_1, x_2)$. The value of $K(x_1, x_2)$ depends on the norm.

Using representers, the cost functional Equation (4) is:[7]

$$J[\phi] = \langle \phi, \phi \rangle + (\langle \phi, r \rangle - z)^T U (\langle \phi, r \rangle - z), \tag{39}$$

where r is a vector of representer functions for each measurement position (i.e., $[r]_i = r_{X_i}$), and $U \in \mathbb{R}^{n \times n}$ is a diagonal weight matrix such that the i-th diagonal element

$$U_{ii} = \mu_i. \tag{40}$$

Using the representers, any function $\phi \in \mathbb{W}$ can be written

$$\phi = r^T b + g, \tag{41}$$

where $b \in \mathbb{R}^n$ is a vector of coefficients, and $g \in \mathbb{W}$ is orthogonal to each representer $[r]_i$ (i.e., $\langle [r]_i, g \rangle = 0$ for $i = 1, \ldots, n$). Minimizing $J[\phi]$ now requires finding the appropriate coefficients b and function g.

Proposition 2. *The minimum $\widehat{\phi}_k = \arg\min_\phi J[\phi]$ is achieved when $g = 0$ and*

$$b = (\langle r, r^T \rangle + U^{-1})^{-1} z. \tag{42}$$

The resulting estimate is

$$\widehat{\phi}_k = r^T (\langle r, r^T \rangle + U^{-1})^{-1} z. \tag{43}$$

Proof. See [2]. □

Although Proposition 2 provides the solution to the minimization problem, it is of limited practical usefulness because we do not know r. The finite element solution explained in Section II-E allows us to approximate the space spanned by r with systematically chosen basis functions and solve for the appropriate coefficients.

C. Relationship Between VIM and BLUE

Both VIM and BLUE are closely related and parameters can be chosen such that they are equivalent (see [2], [15], [16]). We now establish this relationship.

Proposition 3. *VIM and BLUE are equivalent when the reproducing kernel $K(x_1, x_2)$ and the correlation function $R(x_1, x_2)$ are the same and $U = \gamma \epsilon^{-1}$.*

Proof. This proof is based on [2], [15]. Using properties of reproducing kernels and recalling that r is the vector of representers for each measurement location yields

$$r = \mathcal{H}_{(x_1)} K(x, x_1) \tag{44}$$

and

$$\langle r, r^T \rangle = \mathcal{H}_{(x_1)} \mathcal{H}^T_{(x_2)} K(x_1, x_2). \tag{45}$$

Substituting Equations (44) and (45) into Equation (43) yields

$$\mathcal{H}_{(x_1)} K(x, x_1) \left(\mathcal{H}_{(x_1)} \mathcal{H}^T_{(x_2)} K(x_1, x_2) + U^{-1} \right)^{-1} z. \tag{46}$$

[7]When the inner product $\langle \cdot, \cdot \rangle$ acts on vectors of functions, the rules of vector multiplication apply but the multiplication operation is replaced with $\langle \cdot, \cdot \rangle$. For example, $\langle \phi, r \rangle$ is a vector whose i-ith entry is $\langle \phi, [r]_i \rangle$.

Substituting $U = \gamma \epsilon^{-1}$ and $K(x_1, x_2) = R(x_1, x_2)$ yields the BLUE estimate from Equation (35). □

D. Estimation Uncertainty with VIM

Using the analogy between VIM and BLUE allows us to use VIM (and DVIM) to determine the uncertainty in the estimate, as discussed in [7], [11], [12]. Let the estimate of the field, as a function of the measurements and their locations be given by the analysis operator $\mathcal{A}$:

$$\hat{\phi}(x) = \mathcal{A}(z, X). \tag{47}$$

The expected error in Equation (38) can be written as

$$e(x) = C(x, x) - \mathcal{A}(\bar{F}, X), \tag{48}$$

where for each measurement i, $[\bar{F}]_i = C([X]_i, x)$ is the covariance between each measurement location and the estimation location [9].

The analysis operator is evaluated by performing VIM; essentially, to compute the error, measurements are substituted with covariances [9], [12]. Technically, computing the error field requires solving an FEM problem for each estimated error location. However, an approximation using the "poor man error" method of [9], [12] replaces every element in F with the background variance γ. This approximation sacrifices some accuracy for the ability to determine an error field with only one additional FEM solution.

APPENDIX B

This appendix provides a practical guide to the Veubeke triangular elements used in our FEM implementation. More details and theory can be found in [43], [44].

The Veubeke triangular element consists of three sub-triangles, combined into a single triangular element. Each sub-triangle has a complete cubic polynomial interpolating function that is constrained by the interpolating function of its neighboring subtriangle. See Figure 12 for an overall picture.

A. Sub-triangle

1) The Degrees Of Freedom: The interpolating function over the sub-triangle is the complete cubic polynomial:

$$\begin{aligned} &a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy \\ &\quad + a_5 y^2 + a_6 x^3 + a_7 x^2 y + a_8 x y^2 + a_9 y^3, \end{aligned} \tag{49}$$

where a_i is one of the ten coefficients needed to describe the polynomial. There are therefore ten degree-of-freedom functionals that evaluate $\phi_e(x)$ and its derivatives at a given

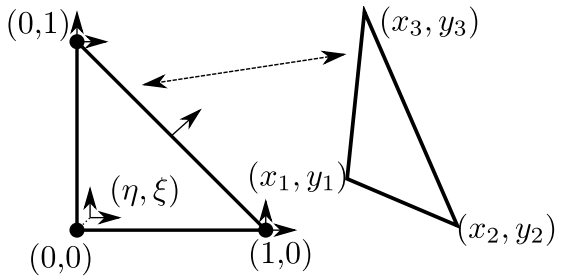

Fig. 11. Veubeke subtriangle in the parent coordinate system (η, ξ) and in the real coordinates (x, y). The dots represent nodes that are values, and the arrows represent nodes that are derivatives (the arrow points in the direction of the derivative). There are three value nodes (at each corner), three x derivative nodes (at each corner), three y derivative nodes (at each corner), and one normal derivative node (at the midpoint of one side).

nodal location. The degree-of-freedom functionals for the triangle with vertices (x_1, y_1), (x_2, y_2), (x_3, y_3) are

$$\sigma_0(p) = p(x_1, y_1) \tag{50}$$

$$\sigma_1(p) = \frac{\partial p(x,y)}{\partial x}\Big|_{\substack{x=x_1\\y=y_1}} \tag{51}$$

$$\sigma_2(p) = \frac{\partial p(x,y)}{\partial y}\Big|_{\substack{x=x_1\\y=y_1}} \tag{52}$$

$$\sigma_3(p) = p(x_2, y_2) \tag{53}$$

$$\sigma_4(p) = \frac{\partial p(x,y)}{\partial x}\Big|_{\substack{x=x_2\\y=y_2}} \tag{54}$$

$$\sigma_5(p) = \frac{\partial p(x,y)}{\partial y}\Big|_{\substack{x=x_2\\y=y_2}} \tag{55}$$

$$\sigma_6(p) = p(x_3, y_3) \tag{56}$$

$$\sigma_7(p) = \frac{\partial p(x,y)}{\partial x}\Big|_{\substack{x=x_3\\y=y_3}} \tag{57}$$

$$\sigma_8(p) = \frac{\partial p(x,y)}{\partial y}\Big|_{\substack{x=x_3\\y=y_3}} \tag{58}$$

$$\sigma_9(p) = \frac{1}{\sqrt{(x_2-x_3)^2+(y_2-y_3)^2}} \times \left(\frac{\partial p(x,y)}{\partial x}, \frac{\partial p(x,y)}{\partial y}\right)\Big|_{\substack{x=\frac{x_1+x_2}{2}\\y=\frac{y_1+y_2}{2}}} \cdot [y_3-y_2, x_2-x_3]. \tag{59}$$

Each of these functionals corresponds to a shape function (which we derive next), and are chosen such that $\sigma_i w_j(x) = 1$ if $i = j$ and zero otherwise.

To simplify the algebra, we define a reference triangle with vertices at $(0, 0)$, $(1, 0)$, and $(0, 1)$ and then transform from this domain to the vertices of the actual triangle (x_1, y_1), (x_2, y_2), (x_3, y_3) (see Figure 11). The correspondence between reference and real triangle vertices is $(x_1, y_1) = (0, 0)$, $(x_2, y_2) = (1, 0)$, and $(x_3, y_3) = (0, 1)$.

The affine transformation between the reference triangle coordinates (η, ξ) and the real triangle coordinates (x, y) is

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} \begin{bmatrix} \eta \\ \xi \end{bmatrix} + \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}. \tag{60}$$

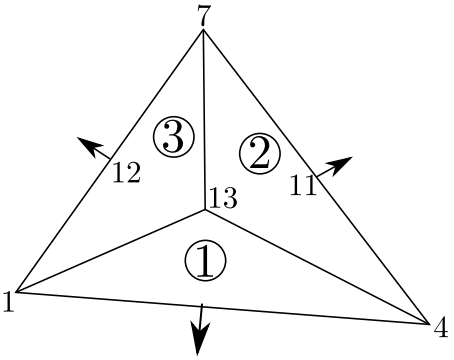

Fig. 12. The full triangle consists of three sub-triangles placed together to form a larger triangle. The circled numbers indicate the sub-triangle index. The smaller numbers are node indices. At each sub-triangle vertices there are three DOF: the value and the x and y derivatives.

To simplify notation we sometimes express coordinates as

$$x = X(\eta, \xi) \tag{61}$$

$$y = Y(\eta, \xi) \tag{62}$$

$$\eta = \hat{\eta}(x, y) \tag{63}$$

$$\xi = \hat{\xi}(x, y). \tag{64}$$

The Jacobian of the transformation is

$$J = \begin{bmatrix} \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \\ \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \xi} \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y2 - y_1 & y_3 - y_1 \end{bmatrix}. \tag{65}$$

The FEM cost functional expressed over an element (i.e., a real triangle) Ω_e is

$$\int_{\Omega_e} f\left(\phi(x,y), \frac{\partial\phi(x,y)}{\partial x}, \frac{\partial\phi(x,y)}{\partial y}, \frac{\partial^2\phi(x,y)}{\partial x^2}, \frac{\partial^2\phi(x,y)}{\partial x\partial y}, \frac{\partial^2\phi(x,y)}{\partial y^2}\right) d\Omega_e, \tag{66}$$

where f depends on the specific variational principle and is a quadratic form of its arguments.

We change coordinates in Equation (66) so that the integration is performed over the reference triangle $\hat{\Omega}_e$:

$$\int_{\hat{\Omega}_e} f\left(\phi(\bar{X}(\eta,\xi)), \frac{\partial\phi(\bar{X}(\eta,\xi))}{\partial x}, \frac{\partial\phi(\bar{X}(\eta,\xi))}{\partial y}, \frac{\partial^2\phi(\bar{X}(\eta,\xi))}{\partial x^2}, \frac{\partial^2\phi(\bar{X}(\eta,\xi))}{\partial x\partial y}, \frac{\partial^2\phi(\bar{X}(\eta,\xi))}{\partial y^2}\right) \times |J| d\hat{\Omega}_e,$$

where $\bar{X}(\eta, \xi) = (X(\eta, \xi), Y(\eta, \xi))$.

Letting $\hat{\phi}(\eta, \xi) = \phi(X(\eta, \xi), Y(\eta, \xi))$ be the interpolating function over the reference triangle, the integral becomes

$$\int_{\hat{\Omega}_e} f\left(\hat{\phi}(\eta,\xi), \frac{\partial\hat{\phi}(\eta,\xi)}{\partial x}, \frac{\partial\hat{\phi}(\eta,\xi)}{\partial y}, \frac{\partial^2\hat{\phi}(\eta,\xi)}{\partial x^2}, \frac{\partial^2\hat{\phi}(\eta,\xi)}{\partial x\partial y}, \frac{\partial^2\hat{\phi}(\eta,\xi)}{\partial y^2}\right) |J| d\hat{\Omega}_e, \tag{67}$$

B. Change Basis from Standard to Reference Triangle

We can write $\hat{\phi}(\eta, \xi)$ in terms of two bases for the complete cubic polynomial space P_3: the standard basis

$$\bar{w}(\eta, \xi) = [1, \eta, \xi, \eta^2, \eta\xi, \xi^2, \eta^3, \eta^2\xi, \eta\xi^2, \xi^3] \tag{68}$$

and a basis $\hat{w}(\eta, \xi)$ defined such that

$$\hat{\sigma}_i(\hat{w}_j) = \begin{cases} 1 & i = j \\ 0 & \text{otherwise,} \end{cases} \tag{69}$$

where $\hat{\sigma}_i$ is the DOF operation for the reference triangle (the derivatives are in reference triangle coordinates as well).

Each basis function in $\hat{w}$ is a linear combination of the components of $\bar{w}$, since both are bases for the same space:

$$\hat{w}(\eta, \xi) = A\bar{w}(\eta, \xi). \tag{70}$$

Applying each $\hat{\sigma}_i$ to Equation (70), and using the linearity of $\hat{\sigma}_i$ and the orthogonality property of Equation (69) yields

$$I = AV, \tag{71}$$

where V^T is a Vandermonde matrix such that $V = [\hat{\sigma}_0(\bar{w}), \ldots, \hat{\sigma}_9(\bar{w})]$. Thus, $A = V^{-1}$ and we can now compute $\hat{w}$ in terms of $\bar{w}$ simply by applying $\hat{\sigma}_i$ to $\bar{w}$.

Noting that

$$\frac{\partial \bar{w}(\eta, \xi)}{\partial \eta} = [0, 1, 0, 2\eta, 0, 3\eta^2, 2\eta\xi, \xi^2, 0]^T \tag{72}$$

$$\frac{\partial \bar{w}(\eta, \xi)}{\partial \xi} = [0, 0, 1, 0, \eta, 2\xi, 0, \eta^2, 2\xi\eta, 3\xi^2]^T \tag{73}$$

$$\left[\frac{\partial \bar{w}(\eta, \xi)}{\partial \xi}\right] \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = [0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \eta\sqrt{2}, \frac{\eta + \xi}{\sqrt{2}}, \xi\sqrt{2}, \frac{3\eta^2}{\sqrt{2}}, \frac{\eta^2}{\sqrt{2}} + \eta\xi\sqrt{2}, \frac{\xi^2}{\sqrt{2}} + \eta\xi\sqrt{2}, \frac{3\xi^2}{\sqrt{2}}]^T, \tag{74}$$

the matrix V is

$$V = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & \frac{3}{4\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \frac{3}{4\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{3}{4\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \frac{3}{4\sqrt{2}} \end{bmatrix}. \tag{75}$$

We now have

$$\hat{\phi}(\eta, \xi) = (V^{-1}\bar{w}(\eta, \xi))^T \hat{q}, \tag{76}$$

where $\hat{q} = \hat{\sigma}(\hat{\phi})$ is the vector of unknown nodal values.

Taking derivatives of Equation (76) and noting that $\hat{q}$ is constant yields:

$$\frac{\partial \hat{\phi}(\eta, \xi)}{\partial x} = \frac{\partial (V^{-1}\bar{w}(\eta, \xi))^T}{\partial x}\hat{q}, \tag{77}$$

$$\frac{\partial \hat{\phi}(\eta, \xi)}{\partial y} = \frac{\partial (V^{-1}\bar{w}(\eta, \xi))^T}{\partial y}\hat{q}. \tag{78}$$

Since $\hat{w}(\eta, \xi) = V^{-1}\bar{w}(\eta, \xi)$ and f is a quadratic form, Equation (67) is equivalent to

$$\hat{q}^T \int_{\hat{\Omega}_e} f\left(\hat{w}(\eta, \xi), \frac{\partial \hat{w}(\eta, \xi)}{\partial x}, \frac{\partial \hat{w}(\eta, \xi)}{\partial y}, \frac{\partial^2 \hat{w}(\eta, \xi)}{\partial x^2}, \frac{\partial^2 \hat{w}(\eta, \xi)}{\partial x \partial y}, \frac{\partial^2 \hat{w}(\eta, \xi)}{\partial y^2}\right) |J| d\hat{\Omega}_e \hat{q}^T. \tag{79}$$

C. Change Basis from Reference to Real Triangle

We need to express $\hat{q}$, the nodal vector in the basis of the reference triangle, in the coordinates of the real triangle to equate neighboring FEM nodes during assembly.

Given the shape functions in the real triangle's basis $w(x, y)$ and the nodal vector q, we have

$$w^T(x, y)q = \hat{w}^T(\hat{\eta}(x, y), \hat{\xi}(x, y))\hat{q}. \tag{80}$$

To change the basis we follow a process analogous to the one used to convert from the standard basis to the reference triangle. For change of basis matrix B,

$$w(x, y) = B\hat{w}(\hat{\eta}(x, y), \hat{\xi}(x, y)). \tag{81}$$

Applying σ_i to both sides of Equation (81) yields

$$\begin{bmatrix} \sigma_0(w(x, y)) & \ldots & \sigma_9(w(x, y)) \end{bmatrix} = BW, \tag{82}$$

where

$$W = \begin{bmatrix} \sigma_0(\hat{w}(\hat{\eta}(x, y), \hat{\xi}(x, y))) & \ldots & \sigma_9(\hat{w}(\hat{\eta}(x, y), \hat{\xi}(x, y))) \end{bmatrix}. \tag{83}$$

The left-hand side of Equation (82), by construction, is I. The matrix W is the transpose of a Vandermonde matrix. We can evaluate W and find B via $B = W^{-1}$.

From Equations (80) and (81) we see that

$$\hat{w}^T(\hat{\eta}(x, y), \hat{\xi}(x, y))B^T q = \hat{w}^T(\hat{\eta}(x, y), \hat{\xi}(x, y))\hat{q}. \tag{84}$$

Therefore,

$$\hat{q} = B^T q. \tag{85}$$

Substituting Equation (85) into Equation (79) allows us to express the integral over a sub-triangle in the proper basis.

D. Merging the Triangles

Three sub-triangles, placed with their normal derivatives outwards to form a full triangle (as depicted in Figure 12) compose the Veubeke triangle. Such triangles are formed from a triangular mesh by splitting each triangle at its centroid.

The combined triangle has 15 nodes: value and derivatives at the triangle vertices, value and derivatives at the centroid, and the normal derivatives at the midpoint of each side. Let $q \in \mathbb{R}^{15}$ be the nodal vector for the triangle, and let $q_1 \in \mathbb{R}^{12}$,

$q_2 \in \mathbb{R}^{12}$, and $q_3 \in \mathbb{R}^{12}$ be the nodal vectors for each sub-triangle. Each sub-triangle's nodal vector is written in terms of the global nodal vector according to

$$q_i = L_i q. \tag{86}$$

This process is analogous to generic finite element assembly.

Each L_i matrix is sparse. Table IV lists the indices of L_i that are equal to one. We partition each L_i into two parts:

$$L_i = \begin{bmatrix} \bar{L}_i & L_0 \end{bmatrix}, \tag{87}$$

where L_0 contains the columns that map the global nodes to the nodes at the centroid of the full triangle and $\bar{L}_i$ maps global nodes to the edges of the triangle. The value of the element inside each sub-triangle is

$$\begin{bmatrix} \phi^{(1)}(x,y) \\ \phi^{(2)}(x,y) \\ \phi^{(3)}(x,y) \end{bmatrix} = \begin{bmatrix} \bar{L}_1 & L_0 \\ \bar{L}_2 & L_0 \\ \bar{L}_3 & L_0 \end{bmatrix} \begin{bmatrix} \bar{q} \\ q_0 \end{bmatrix}, \tag{88}$$

where q_0 contains the nodal values at the centroid of the triangle and $\bar{q}$ contains all other nodal values. Equation (88) provides the value anywhere within the whole triangle, given the global node vector q. To remove the internal degree of freedom we introduce a constraint and perform static condensation.

E. Static Condensation

We first introduce constraints at the interfaces between the sub-elements. Let n_{ij} be the normal vector between sub-element i and j, with $n_{ij} = -n_{ji}$. Let $\frac{\partial \phi}{n}$ be the directional derivative in the direction of normal vector n.

At the midpoint on the internal edge between sub-elements i and j we introduce the constraint

$$\frac{\partial \phi^{(i)}}{n_{ij}} + \frac{\partial \phi^{(j)}}{n_{ji}} = 0. \tag{89}$$

In terms of the basis functions and the global node vector, Equation (89) becomes

$$\left(\frac{\partial w^{(i)}}{\partial n_{ij}} R_i + \frac{\partial w^{(j)}}{\partial n_{ji}} R_j \right) q = 0. \tag{90}$$

Stacking the equations at each interface yields

$$\begin{bmatrix} \bar{W} & W_0 \end{bmatrix} \begin{bmatrix} \bar{q} \\ q_0 \end{bmatrix} = 0, \tag{91}$$

with

$$\bar{W} = \begin{bmatrix} \frac{\partial w^{(1)}}{\partial n_{13}} \bar{R}_1 + \frac{\partial w^{(3)}}{\partial n_{31}} \bar{R}_3 \\ \frac{\partial w^{(2)}}{\partial n_{21}} \bar{R}_2 + \frac{\partial w^{(1)}}{\partial n_{12}} \bar{R}_1 \\ \frac{\partial w^{(3)}}{\partial n_{32}} \bar{R}_3 + \frac{\partial w^{(2)}}{\partial n_{23}} R_2 \end{bmatrix}, \tag{92}$$

and

$$W_0 = \begin{bmatrix} \frac{\partial w^{(1)}}{\partial n_{13}} R_0 + \frac{\partial w^{(3)}}{\partial n_{31}} R_0 \\ \frac{\partial w^{(2)}}{\partial n_{21}} R_0 + \frac{\partial w^{(1)}}{\partial n_{12}} R_0 \\ \frac{\partial w^{(3)}}{\partial n_{32}} R_0 + \frac{\partial w^{(2)}}{\partial n_{23}} R_0 \end{bmatrix}. \tag{93}$$

Solving the constraint in Equation (91) for q_0 yields

$$q_0 = -W_0^{-1} \bar{W} \bar{q}. \tag{94}$$

L_1	(1,13)	(2,14)	(3,15)	(4,1)	(5,2)
L_1	(6,3)	(7,4)	(8,5)	(9,6)	(10,10)
L_2	(1,13)	(2,14)	(3,15)	(4,4)	(5,5)
L_2	(6,6)	(7,7)	(8,8)	(9,9)	(10,11)
L_3	(1,13)	(2,14)	(3,15)	(4,7)	(5, 8)
L_3	(6,9)	(7,1)	(8,2)	(9,3)	(10,12)

TABLE IV
THE NONZERO INDEXES OF EACH L_i MATRIX.

From the FEM for an element, we must minimize

$$\frac{1}{2} q^T K q - g^T q, \tag{95}$$

where K is the stiffness matrix and g is the load vector.

Partitioning K and g to conform with q yields

$$\frac{1}{2} \begin{bmatrix} q^T & q_0^T \end{bmatrix} \begin{bmatrix} K_1 & K_2 \\ K_3 & K_4 \end{bmatrix} \begin{bmatrix} q \\ q_0 \end{bmatrix} - \begin{bmatrix} \bar{g}^T & g_0^T \end{bmatrix} \begin{bmatrix} \bar{q} \\ q_0 \end{bmatrix}. \tag{96}$$

Expanding Equation (96) yields

$$\frac{1}{2} (\bar{q}^T K_1 \bar{q} + \bar{q}^T K_2 q_0 + q_0^T K_3 \bar{q} + q_0^T K_4 q_0) - \bar{g}^T \bar{q} - g_0^T q_0. \tag{97}$$

Substituting Equation (94) into Equation (97) yields

$$\frac{1}{2} \bar{q}^T K_r \bar{q} - g_r^T \bar{q}, \tag{98}$$

where the reduced stiffness matrix is

$$K_r = K_1 - K_2 W_0^{-1} \bar{W} - \bar{W}^T W_0^{-T} K_3 + \bar{W}^T W_0^{-T} K_4 W_0^{-1} \bar{W} \tag{99}$$

and the reduced load vector is

$$g_r = \bar{g}^T + g_0^T W_0^{-1} \bar{W}. \tag{100}$$

Now, in the assembly process, each element uses the reduced stiffness and load vectors K_r and g_r. Additionally, when assembling, the normal derivatives of adjacent elements point in opposite directions. To account for this, one element considers positive nodal values to represent positive normal derivatives and the other element considers positive normal values to represent negative normal derivatives. The element that views positive nodal values as the negative normal derivatives performs the normal assembly process, but then must negate some elements of K_r and g_r. If the element views its local node j as being negated, it negates row j of K_r and g_r and also column j of K_r to account for the difference (note that $[K_r]_{jj}$ retains its original sign).

In the centralized case the global node indexes are used to determine which element negates its normal derivatives. In the distributed case, the robot with the lower identifier negates its adjacent normal derivatives.

F. Polygonal Elements

Another static condensation process is performed to combine the Veubeke triangular elements into polygonal elements corresponding to each Voronoi cell. To perform the static condensation, the robots partition q_i into two parts: $\bar{q}_i$, which contains the nodal values on the boundary of the subdomain $\bar{\Omega}_i$, and ξ_i, which contains the nodal values in the interior of

$\bar{\Omega}_i$. Partitioning K_i and g_i to conform with the partitioning of q_i yields the local FEM equation

$$\begin{bmatrix} \bar{K}_i & \hat{K}_i \\ \hat{K}_i & \tilde{K}_i \end{bmatrix} \begin{bmatrix} \bar{q}_i \\ \xi_i \end{bmatrix} = \begin{bmatrix} \bar{g}_i \\ \tilde{g}_i \end{bmatrix}. \tag{101}$$

Expanding Equation (101) yields

$$\xi_i = \tilde{K}_i^{-1}(\tilde{g}_i - \hat{K}_i \bar{q}_i), \tag{102}$$

and

$$K_i' \bar{q}_i = g_i', \tag{103}$$

where

$$K_i' = \bar{K}_i - \hat{K}_i \tilde{K}_i^{-1} \hat{K}_i, \tag{104}$$

and

$$g_i' = \bar{g}_i - \hat{K}_i \tilde{K}_i^{-1} \tilde{g}_i. \tag{105}$$

Matrix $\tilde{K}_i$ is invertible by construction of the FEM problem. Now we only must find $\bar{q}_i$ in a decentralized way; once a robot knows $\bar{q}_i$ it can recover ξ_i using Equation (102).

References

[1] D. M. Glover, W. J. Jenkins, and S. C. Doney, *Modeling Methods for Marine Science*. Cambridge: Cambridge University Press, 2011.

[2] A. F. Bennett, *Inverse Methods in Physical Oceanography*. Cambridge University Press, 1992.

[3] J. Fish and T. Belytschko, *A First Course in Finite Elements*. John Wiley & Sons, 2007.

[4] Z. Chen, *Finite Element Methods and Their Applications*. Heidelberg: Springer-Verlag, 2005.

[5] P. P. Brasseur and J. A. Haus, "Application of a 3-D variational inverse model to the analysis of ecohydrodynamic data in the Northern Bering and Southern Chukchi Seas," *Journal of Marine Systems*, vol. 1, no. 4, pp. 383 – 401, 1991.

[6] P. Brasseur, J. Beckers, J. Brankart, and R. Schoenauen, "Seasonal temperature and salinity fields in the Mediterranean Sea: Climatological analyses of a historical data set," *Deep Sea Research Part I: Oceanographic Research Papers*, vol. 43, no. 2, pp. 159 – 192, 1996.

[7] C. Troupin, M. Ouberdous, D. Sirjacobs, A. Alvera-Azcárate, A. Barth, M.-E. Toussaint, S. Watelet, and J.-M. Beckers, *DIVA User Guide*, GeoHydrodynamics and Environment Research, MARE (GHER), University of Liege, 2015. [Online]. Available: http://modb.oce.ulg.ac.be

[8] C. Troupin, F. Machín, M. Ouberdous, D. Sirjacobs, A. Barth, and J.-M. Beckers, "High-resolution climatology of the northeast Atlantic using data-interpolating variational analysis (DIVA)," *Journal of Geophysical Research: Oceans*, vol. 115, no. C8, 2010.

[9] C. Troupin, A. Barth, D. Sirjacobs, M. Ouberdous, J.-M. Brankart, P. Brasseur, M. Rixen, A. Alvera-Azcárate, M. Belounis, A. Capet, F. Lenartz, M.-E. Toussaint, and J.-M. Beckers, "Generation of analysis and consistent error fields using the data interpolating variational analysis (DIVA)," *Ocean Modelling*, vol. 52–53, pp. 90 – 101, 2012.

[10] A. Barth, A. A. Azcárate, P. Joassin, J.-M. Becers, and C. Troupin, *Introduction to Optimal Interpolation and Variational Analysis*, GeoHydrodynamics and Environment Research (GHER), 2008.

[11] J. Brankart and P. Brasseur, "The general circulation in the Mediterranean Sea: a climatological approach," *Journal of Marine Systems*, vol. 18, no. 1–3, pp. 41 – 70, 1998.

[12] J.-M. Beckers, A. Barth, C. Troupin, and A. Alvera-Azcárate, "Approximate and efficient methods to assess error fields in spatial gridding with data interpolating variational analysis (DIVA)," *Journal of Atmospheric and Oceanic Technology*, vol. 31, no. 2, pp. 515–530, 2014.

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[14] A. Teixeira, E. Ghadimi, I. Shames, H. Sandberg, and M. Johansson, "The ADMM algorithm for distributed quadratic problems: Parameter selection and constraint preconditioning," *IEEE Transactions on Signal Processing*, vol. 64, no. 2, pp. 290–305, Jan 2016.

[15] G. Wahba, *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

[16] P. C. McIntosh, "Oceanographic data interpolation: Objective analysis and splines," *Journal of Geophysical Research: Oceans*, vol. 95, no. C8, pp. 13 529–13 541, 1990.

[17] M. L. Elwin, "Distributed algorithms for multi-robot environmental monitoring," Ph.D. dissertation, Northwestern University, 2017.

[18] M. L. Elwin, R. A. Freeman, and K. M. Lynch, "A systematic design process for internal model average consensus estimators," in *52nd IEEE Conference on Decision and Control*, Dec 2013, pp. 5878–5883.

[19] ——, "Worst-case optimal average consensus estimators for robot swarms," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 3814–3819.

[20] B. V. Scoy, R. A. Freeman, and K. M. Lynch, "Feedforward estimators for the distributed average tracking of bandlimited signals in discrete time with switching graph topology," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 4284–4289.

[21] H. Bai, R. A. Freeman, and K. M. Lynch, "Robust dynamic average consensus of time-varying inputs," in *49th IEEE Conference on Decision and Control (CDC)*, Dec 2010, pp. 3104–3109.

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

[23] H. Bai, R. A. Freeman, and K. M. Lynch, "Distributed Kalman filtering using the internal model average consensus estimator," in *Proceedings of the 2011 American Control Conference*, June 2011, pp. 1500–1505.

[24] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proceedings of the 46th IEEE Conference on Decision and Control*, Dec 2007, pp. 5492–5498.

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

[26] G. Battistelli, L. Chisci, N. Forti, S. Selleri, and G. Pelosi, "Decentralized consensus finite-element Kalman filter for field estimation," *arXiv*, vol. abs/1604.02392, 2016.

[27] P. P. Brasseur, "A variational inverse method for the reconstruction of general circulation fields in the northern Bering Sea," *Journal of Geophysical Research: Oceans*, vol. 96, no. C3, pp. 4891–4907, 1991.

[28] S. Martínez, "Distributed interpolation schemes for field estimation by mobile sensor networks," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 2, pp. 491–500, March 2010.

[29] J. Cortes, "Distributed Kriged Kalman filter for spatial estimation," *Automatic Control, IEEE Transactions on*, vol. 54, no. 12, pp. 2816–2827, Dec 2009.

[30] P. Dames, "Distributed multi-target search and tracking using the phd filter," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, Dec 2017, pp. 1–8.

[31] X. Lan and M. Schwager, "Rapidly exploring random cycles: Persistent estimation of spatiotemporal fields with multiple sensing robots," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1230–1244, Oct 2016.

[32] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, Feb 2016.

[33] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration on the motion group SE(2)," in *52nd IEEE Conference on Decision and Control*, Dec 2013, pp. 4517–4522.

[34] R. O'Flaherty and M. Egerstedt, "Optimal exploration in unknown environments," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2015, pp. 5796–5801.

[35] N. Leonard, D. Paley, F. Lekien, R. Sepulchre, D. Fratantoni, and R. Davis, "Collective motion, sensor networks, and ocean sampling," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 48–74, Jan 2007.

[36] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme, "Persistent ocean monitoring with underwater gliders: Adapting sampling resolution," *Journal of Field Robotics*, vol. 28, no. 5, pp. 714–741, 2011.

[37] A. Kozma, C. Conte, and M. Diehl, "Benchmarking large-scale distributed convex quadratic programming algorithms," *Optimization Methods and Software*, vol. 30, no. 1, pp. 191–214, 2015.

[38] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.

[39] A. Kozma, E. Klintberg, S. Gros, and M. Diehl, "An improved distributed dual Newton-CG method for convex quadratic programming problems," in *2014 American Control Conference*, June 2014, pp. 2324–2329.

[40] W. Bangerth, C. Burstedde, T. Heister, and M. Kronbichler, "Algorithms and data structures for massively parallel generic adaptive finite element codes," *ACM Trans. Math. Softw.*, vol. 38, 2011.
[41] C. Nielsen, W. Zhang, L. Alves, N. Bay, and P. Martins, *Modeling of Thermo-Electro-Mechanical Manufacturing Processes with Applications in Metal Forming and Resistance Welding*. Springer, 2012.
[42] M. L. Elwin, R. A. Freeman, and K. M. Lynch, "Environmental estimation with distributed finite element agents," in *55th IEEE Conference on Decision and Control*, Dec 2016, pp. 5918–5924.
[43] B. F. D. Veubeke, G. Sander, and P. Beckers, "Dual analysis by finite elements: linear and non linear applications," University of Liege, Tech. Rep. AFFDL-TR-72-93, 1971. [Online]. Available: orbi.ulg.ac.be/bitstream/2268/205883/1/ST_Veubeke_023.pdf
[44] B. F. De Veubeke, "Variational principles and the patch test," *International Journal for Numerical Methods in Engineering*, vol. 8, no. 4, pp. 783–801, 1974.
[45] A. Okabe, B. Boots, K. Sugihara, S. N. Chiu, and D. G. Kendall, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed. John Wiley and Sons, 2000.
[46] M. L. Elwin, R. A. Freeman, and K. M. Lynch, "Distributed Voronoi neighbor identification from inter-robot distances," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1320–1327, July 2017.
[47] M. Cao and C. Hadjicostis, "Distributed algorithms for Voronoi diagrams and application in ad-hoc networks," UIUC Coordinated Science Laboratory, Tech. Rep. UILU-ENG-03-2222,DC-210, 2003.
[48] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *Robotics and Automation, Proceedings of the IEEE International Conference on*, vol. 2, 2002, pp. 1327–1332.
[49] A. Breitenmoser, M. Schwager, J. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 4982–4989.
[50] S. Bhattacharya, R. Ghrist, and V. Kumar, "Multi-robot coverage and exploration on Riemannian manifolds with boundaries," *Int. J. Rob. Res.*, vol. 33, no. 1, pp. 113–137, Jan. 2014.
[51] D. Sieger, P. Alliez, and M. Botsch, *Optimizing Voronoi Diagrams for Polygonal Finite Element Computations*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 335–350.
[52] J. R. Shewchuk, "What is a good linear finite element? Interpolation, conditioning, anisotropy, and quality measures," In Proc. of the 11th International Meshing Roundtable, Tech. Rep., 2002.
[53] L. Rineau, "2D conforming triangulations and meshes," in *CGAL User and Reference Manual*, 4.7 ed. CGAL Editorial Board, 2015.
[54] D. P. Kroese and Z. I. Botev, "Spation process simulation," in *Stochastic Geometry, Spatial Statistics and Random Fields: Models and Algorithms*, V. Schmidt, Ed. Berlin: Springer-Verlag, 2015, pp. 369–404.

Matthew L. Elwin (S'11–M'17) received a B.E. in engineering sciences from Dartmouth College, Hanover, NH, USA, in 2009 and a Ph.D. degree in mechanical engineering from Northwestern University, Evanston, IL, USA, in 2017. He is an Assistant Professor of Instruction in the Mechanical Engineering Department, Northwestern University, Evanston, IL, USA, where he is affiliated with the Master of Science in Robotics program and the Northwestern Center for Robotics and Biosystems. His research interests include multi-robot coordination; distributed estimation and control; and mechatronics. He is a coauthor of the textbook *Embedded Computing and Mechatronics* (Elsevier, 2015).

Randy A. Freeman (S'90–M'95) received a Ph.D. in Electrical Engineering from the University of California at Santa Barbara in 1995. Since then he has been a faculty member at Northwestern University (Evanston, Illinois), where he is currently Professor of Electrical and Computer Engineering. He received the National Science Foundation CAREER Award in 1997. He has been a member of the IEEE Control System Society Conference Editorial Board since 1997, and has served on Program and Operating Committees for the American Control Conference, the IEEE Conference on Decision and Control, and the IFAC Workshop on Distributed Estimation and Control in Networked Systems. He has served as an associate editor for the *IEEE Transactions on Automatic Control* and the *IEEE Transactions on Control of Network Systems*. His research interests include complex systems, nonlinear systems, distributed control, multi-agent systems, robust control, optimal control, and oscillator synchronization.

Kevin M. Lynch (S'90–M'96–SM'05–F'10) received the B.S.E. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 1989, and the Ph.D. degree in robotics from Carnegie Mellon University, Pittsburgh, PA, USA, in 1996. He is a Professor and the Chair of the Mechanical Engineering Department, Northwestern University, Evanston, IL, USA. He is director of the Northwestern Center for Robotics and Biosystems and a member of the Northwestern Institute on Complex Systems. His research interests include dynamics, motion planning, and control for robot manipulation and locomotion; self-organizing multiagent systems; and human-robot systems. He is a coauthor of the textbooks *Principles of Robot Motion* (MIT Press, 2005), *Embedded Computing and Mechatronics* (Elsevier, 2015), and *Modern Robotics: Mechanics, Planning, and Control* (Cambridge University Press, 2017). He is editor-in-chief of the *IEEE Transactions on Robotics*