Environmental Estimation With Distributed Finite Element Agents

Matthew L. Elwin

Randy A. Freeman

Kevin M. Lynch

Abstract—We develop an environmental estimation method that allows large groups of agents to infer the value of an environmental field using measurements. Agents maintain estimates for subregions of the domain and communicate with local neighbors, so the method's communication and memory requirements do not increase with the number of agents, the size of the environment representation, or the agents' density in the environment. Despite the distributed representation, the union of individual estimates matches an estimate generated by a central computer with access to all measurements employing the variational inverse method, a finite element-based interpolation procedure. We also introduce a distributed query system, allowing users to determine an estimate anywhere in the domain without accessing all measurements or the full environment representation.

I. INTRODUCTION

A common problem in sciences such as oceanography, earth science, and atmospheric science is to infer the value of a scalar field (e.g., temperature, chemical concentration, radiation intensity) anywhere in an environment based on irregularly spaced measurements. Typically, the relatively sparse data are processed by a central computer; such methods have been studied extensively (see e.g., [1]). The advent of cheap sensors, however, presents an opportunity to collect more data than ever before. As the number of sensors used to measure an environment increases, so do computation and communication burdens. These problems become especially acute in remote environments such as the ocean, the polar icecaps, outer space, and underground. In such environments, communication with a distant base station may be significantly harder than communication between agents, making it impractical to send all measurements to a central computer.

To address the problem of handling many measurements with limited memory and bandwidth, we develop a distributed version of the variational inverse method (VIM) for data interpolation [2]–[6]. This technique, used in oceanography, generates an estimate by minimizing a cost functional using the Finite Element Method (FEM) [7]. Rather than having a central computer collect data from sensors, we deploy the sensors on agents that communicate with each other and perform computations. Each agent determines an estimate for a small portion of the environment. No single agent, therefore, stores a full environmental representation

This work is supported by the Office of Naval Research, Grant N00014-13-1-0331. The authors are with 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@u.northwestern.edu, freeman@eecs.northwestern.edu, kmlynch@northwestern.edu.

or all of the measurements. By communicating with local neighbors, however, the collection of all agents' estimates matches the result obtained by a central computer using VIM.

We use the alternating direction method of multipliers (ADMM) to distribute computations and solve the FEM problem [8]. Other methods, which generally require more communication than ADMM include [9] and [10].

We couple our distributed VIM (DVIM) method with a query system, allowing a user to determine the value of the field anywhere. The user can obtain a low-resolution view of the overall field and then focus on higher resolution estimates in areas of particular interest. Using this system, the full set of measurements and a complete environment representation are never communicated to or stored by a single computer. When communication with a remote base station (e.g., via satellite) costs significantly more than interagent communication (e.g., via XBee), minimizing such communication becomes especially important.

In Section II we compare DVIM to existing methods in the literature. In Section III we formulate the problem, introducing VIM and FEM. Section IV describes our distributed solution, DVIM, and the query system. Section V provides simulation results. We outline our conclusions and future work in Section VI.

II. METHOD COMPARISON

We compare DVIM to existing techniques whose computation, communication, and memory requirements do not increase with respect to the number of agents. Many criteria can be used to evaluate methods, but in this work we are particularly concerned with three desirable properties: scalability with respect to the environment representation, scalability with respect to agent density, and incorporation of all relevant measurements.

Scalability with respect to the environment representation means that every agent's memory and communication requirements do not increase with the resolution of the environment model. Methods that scale with the environment representation allow adding more agents with fixed capabilities to obtain higher resolution estimates.

Scalability with respect to agent density means that every agent's memory and communication requirements do not increase as the density of the agents in the environment (and therefore measurement density) increases. High measurement density contributes to more accurate estimates, so methods with this property provide opportunities for improved accuracy by adding more agents with fixed capabilities.

Methods that incorporate all relevant measurements generate an estimate at a given point using measurements from

| | VIM | KF [12] | KK [13] | NP [14] | NN [11] | ID [11] | FE [15] |
|---------|-----|---------|---------|---------|---------|---------|---------|
| SER | X | | | | Х | Х | Х |
| SAD | Х | Х | | Х | Х | | Х |
| IRM | Х | Х | Х | Х | | Х | Х |
| TABLE I | | | | | | | |

Comparison of the various methods. SER - scale with environment representation, SAD - scale with agent density, IRM - incorporate relevant measurements.

locations correlated with that point. Many methods assume that the correlation between points decreases with distance, and that points further than a characteristic correlation length apart are uncorrelated. Although VIM (and therefore DVIM) does not rely on this assumption (an advantage; it can handle cases when nearby points should be uncorrelated, see [3]), it incorporates all measurements when forming estimates and does have a notion of spatial correlation [4]. Methods that disregard measurements that should influence an estimate at a given point do not use information efficiently.

Two features of all the methods we examine that DVIM currently lacks are control laws for mobile agents and error estimation abilities (we leave these for future work). To ensure a fair comparison, we assume, for all papers, stationary agents and known error fields for all agents. Although estimating an error field provides benefits, it also imposes additional communication and memory requirements on the agents. Additionally, in [11], movement increases communication and memory requiremental representation is coupled to the agents' trajectories.

We now describe several methods and evaluate whether they scale with the environment representation, scale with agent density, and incorporate all relevant measurements. Table I shows this comparison, with more details provided below. Note that a central computer collecting measurements from all agents and applying VIM does not scale with the environment representation or with agent density.

A. Distributed Kalman Filter [12]

This method uses a distributed Kalman filter based on average consensus estimators to incorporate measurements and estimate basis function coefficients that represent the environment. The method could also work using the distributed data fusion techniques of [16]. Communication and memory is proportional to the number of basis functions, which is fixed beforehand; therefore, this method does not scale with the environment representation. Every agent, however, has a full environment estimate. This method scales with agent density because it works over any connected communication graph. The relationship between measurements at different points is incorporated in the basis function choice, so measurements contribute to the field estimate everywhere.

B. Kriged Kalman Filter [13]

This method represents the environment as the mean and covariance functions of a spatial Gaussian process. The representation consists of basis function coefficients (as in [12]) and as a combination of measurements within a given radius R of each agent. If the coefficients are unknown, communication and memory is proportional to the number of basis functions; however, by assuming known coefficients, communication is independent of the global environment description and this method can scale with the environment representation. Regardless, the communication increases with density: estimation requires the agents to incorporate all measurements within a radius of R. Since the agents account for measurements within a radius of R and points further than R apart are assumed to be uncorrelated, all relevant measurements contribute to the estimate.

C. Non-parametric Information [14]

This method represents the environment as a weighted set of discrete samples from a probability distribution. Accordingly, it can approximate any type of environment, not just scalar fields. Every agent, however, has a complete environment representation and must communicate information proportional to the size of this representation. This method, therefore, does not scale with the representation size. It does, however, scale with agent density because it works over connected graphs. Information is fused using a particle-filter like approach, so the estimate at a given location incorporates relevant measurements.

D. Local Interpolation: Nearest Neighbor [11]

In this method, the environment is a piecewise constant function. When used with stationary agents, an agent's estimate over its Voronoi region is its measurement. This method, therefore, scales with the environment representation (adding more agents adds more Voronoi cells) and with density. The nearest neighbor approach, however, uses a single measurement to estimate the field over each Voronoi cell, so it does not use all relevant measurements. With mobile agents, as originally intended, the initial Voronoi cells are subdivided based on current and past measurements, creating a more detailed estimate. The environment representation, however, is coupled to the agent's trajectories; lossy compression is used to maintain scalability with respect to the number of measurements when the agents move.

E. Local Interpolation: Inverse Distance [11]

Closely related to Nearest Neighbor Local Interpolation, this method uses piecewise continuous functions to represent the environment. Here, measurements have a diminishing effect on the estimate as distance from the measurement decreases. Thus, unlike the Nearest Neighbor version, estimates at a given point incorporate relevant measurements. However, now the estimate must incorporate all measurements from within a given radius R, so the method no longer scales with agent density.

F. Distributed Finite Element Kalman Filter [15], [17]

In this method, the environment is modeled as a partial differential equation (PDE) in space and time. Using finite elements over the spatial variables and discretizing over time, the PDE is converted into a discrete time state-space system and a Kalman Filter used to estimate the state. Each agent is

responsible for only a subset of the domain; thus the method scales with respect to the environmental model. At each time-step, the state is estimated using the parallel Schwarz method and measurements are incorporated using a Kalman filter update. The method requires only information shared by adjacent regions to be exchanged and converges to a centralized solution; therefore the method scales with agent density and incorporates all relevant information.

G. Distributed Variational Inverse Method

This method, as developed here, is scalable with respect to environment representation, agent density, and incorporates relevant measurements when forming an estimate. The environment representation is the total number of nodes in a global FEM mesh, but each agent handles only a subset of those nodes. As the density of the agents increases, communication costs do not increase because communication is proportional to the number of Voronoi neighbors and the perimeter of every agent's Voronoi region, which do not increase with density. Finally, in each agents' region, the estimate matches the centralized VIM method, which incorporates all measurements into the estimate.

Unlike [15], [17], our method does not require a model of the field; thus it is applicable in situations when partial differential equations for the environment are not known. Additionally, we present a method to form consistent finite element meshes in a distributed manner and a query system to extract the estimate, issues that are not addressed in [15], [17].

III. VARIATIONAL INVERSE METHOD

A. Sensor and Environment Model

The environment is a scalar field $\overline{\phi}(x, y)$ over a region $\Omega \subset \mathbb{R}^2$. Stationary agents dispersed over Ω , with agent *i* located at $(x_i, y_i) \in \Omega$, inaccurately measure the field:

$$z_i = \phi(x_i, y_i) + v_i, \tag{1}$$

where z_i is agent *i*'s measurement and v_i is an error (e.g. Gaussian noise) that depends on the particular agent. There are *n* measurements, which we use to generate an estimate $\phi(x, y)$ of the field $\overline{\phi}(x, y)$ everywhere in Ω .

B. Variational Inverse Method

We use the Variational Inverse Method (VIM), an interpolation method closely related to spline interpolation to estimate the environment [2]–[6]. Overall, the method generates an estimate $\phi(x, y)$ of the field $\overline{\phi}(x, y)$ by minimizing a cost functional using the Finite Element Method (FEM) [7].

C. Variational Principle

The cost functional to minimize is

$$J[\phi] = \int_{\Omega} S[\phi] d\Omega + D[\phi]$$
⁽²⁾

where $\phi(x, y)$ is the field estimate, $S[\phi]$ penalizes nonsmoothness of the estimate and $D[\phi]$ penalizes the deviation



Fig. 1. Finite element mesh with seven quadratic elements. Measurements are denoted with x. The dots represent the nodes.

of the estimate from measurements. In this paper we use

$$S[\phi] = \left(\frac{\partial^2 \phi}{\partial x^2}\right)^2 + \left(\frac{\partial^2 \phi}{\partial y^2}\right)^2 + 2\left(\frac{\partial^2 \phi}{\partial x \partial y}\right)^2 + \alpha_1 \left(\left(\frac{\partial \phi}{\partial x}\right)^2 + \left(\frac{\partial \phi}{\partial y}\right)^2\right) + \alpha_0 (\phi - \phi_0)^2 \quad (3)$$

and

$$D[\phi] = \sum_{k=1}^{n} \mu_k \left(\phi(x_k, y_k) - z_k\right)^2$$
(4)

where $\phi_0(x, y)$ is a known background field, α_1 is a weight on the curvature, α_0 weights the error of the field, and μ_k weights each measurement.

The known background field $\phi_0(x, y)$ is the assumed value of the field in the absence of measurements; it allows us to incorporate prior assumptions about the field into the estimate. Alternatively, $\phi_0(x, y)$ can be taken to be the average of the data, a linear fit to the data, or zero. When $\phi_0(x, y) = 0$ it makes sense to let $\alpha_0 = 0$ to avoid penalizing the magnitude of the resulting estimated field.

D. Finite Element Method

We use FEM to minimize the cost functional (2); details can be found in [7]. FEM consists of the following five steps:

- 1) Divide the domain into sub-domains called elements.
- 2) Find interpolating equations for each element based on values at points within each element, called nodes.
- 3) Assemble elements into a global system of equations.
- 4) Solve the equations.
- 5) Post-process the result to obtain information from the solution, for example, producing a visualization.

1) Dividing the domain: The domain is partitioned into a mesh of m non-overlapping sub-domains called elements (see Figure 2). Each element's domain is Ω_e and the elements approximately cover Ω . Elements only intersect at vertices or along full edges. Many element geometries are available. For simplicity, we use a mesh of triangular elements, as generated by the Computational Geometry Algorithms Library (CGAL) [18].

2) Interpolating equations: The estimate within each element, $\phi_e(x, y)$, is approximated by a polynomial containing all terms up to order d, which can be fully described by r coefficients. This polynomial's coefficients are related to the value of $\phi_e(x, y)$ at points within the element called nodes:

$$\phi_e(x,y) = w^T(x,y)q_e,\tag{5}$$

where w(x, y) is the vector of r shape functions and $q_e \in \mathbb{R}^r$ is the vector of nodal values. Many shape functions are available. We use quadratic polynomials over triangular domains, so r = 2 (see Figure 2). The polynomial is $b_0+b_1x+b_2y+b_3x^2+b_4xy+b_5y^2$, where b_i is a coefficient. There are six coefficients and therefore six nodes (one at each corner and one bisecting each edge).

Each shape function (an element of w(x, y)) corresponds to a node: it is one at that node and zero at all other nodes. For example, with r = 2, evaluating $w^T(x, y)$ at element e's second node yields $\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}$, which selects the second element of q_e in equation (5).

3) Assembly: Each nodal vector q_e is local to element e; however, nodes on adjacent element edges overlap. Generally, therefore, there are p < rm nodes in the finite element mesh. The global node vector $q \in \mathbb{R}^p$ is related to the local node vectors by the Boolean stochastic gather matrices $L_e \in \mathbb{R}^{r \times p}$ such that:

$$q_e = L_e q. \tag{6}$$

By defining $\phi_e(x, y) = 0$ when $(x, y) \notin \Omega_e$, an approximation of the whole field is

$$\phi(x,y) \approx \sum_{i=1}^{m} \phi_e(x,y). \tag{7}$$

Note that by using the Dirac delta δ , all terms of the cost functional (2) can be written inside the integral as

$$J[\phi] = \int_{\Omega} \left(S[\phi] + D'[\phi] \right) d\Omega, \tag{8}$$

where

$$D'[\phi] = \sum_{k=1}^{n} \mu_k (\phi - z_k)^2 \delta(x - x_k) \delta(y - y_k).$$
(9)

Substituting equation (7) into the cost functional (9) and performing some manipulations yields

$$J[\phi] \approx \sum_{e=1}^{m} (q_e^T K_e q_e - 2q_e^T g_e) + \sum_{k=1}^{n} \mu_k (z_k - \phi_0(x_k, y_k))^2,$$
(10)

with

$$K_{e} = \int_{\Omega_{e}} \left[\left(\frac{\partial^{2} w}{\partial x^{2}} \right) \left(\frac{\partial^{2} w}{\partial x^{2}} \right)^{T} + \left(\frac{\partial^{2} w}{\partial y^{2}} \right) \left(\frac{\partial^{2} w}{\partial y^{2}} \right)^{T} \right. \\ \left. + 2 \left(\frac{\partial^{2} w}{\partial x \partial y} \right) \left(\frac{\partial^{2} w}{\partial x \partial y} \right)^{T} + \alpha_{1} \left(\frac{\partial w}{\partial x} \right) \left(\frac{\partial w}{\partial x} \right)^{T} \right. \\ \left. + \alpha_{1} \left(\frac{\partial w}{\partial y} \right) \left(\frac{\partial w}{\partial y} \right)^{T} + \alpha_{0} w w^{T} \right] d\Omega_{e} \\ \left. + \sum_{(x_{k}, y_{k}) \in \Omega_{e}} \mu_{k} w(x_{k}, y_{k}) w^{T}(x_{k}, y_{k})$$
(11)

and

$$g_e = \sum_{(x_k, y_k) \in \Omega_e} \mu_k w(x_k, y_k) z_k.$$
(12)

Substituting equation (6) into equation (10) and minimizing with respect to q yields the global FEM equation

m

$$Kq = g, \tag{13}$$

where

$$K = \sum_{e=1}^{m} L_e^T K_e L_e \tag{14}$$

and

$$g = \sum_{e=1}^{m} L_e^T g_e.$$
(15)

4) Solving: Solving equation (13) for q provides the field estimate. For brevity we have omitted boundary conditions; implicitly assuming that derivatives of the estimate across the boundary are zero. However, our technique extends to cases when either the derivatives on the boundary or the value of some boundary nodes are specified.

5) *Post-processing:* Given the global node vector q, the field can be estimated at any location using equations (7) and (6), allowing us to, for example, plot the estimate.

IV. DISTRIBUTED METHOD

Our method, distributed VIM (DVIM), enables agents to estimate fields using local communication without any agent needing all measurements or a full environment representation. The scaling properties of the algorithm, derived from its distributed nature, allow for more detailed environment representations as the group size increases without a concomitant increase in the memory or communication requirements of individuals. Memory and communication requirements also do not increase with increasing agent density. Individual agents' estimates are consistent with VIM solved on a central computer with full access to all measurements.

To retrieve estimates from the group requires a query system. When at least one agent can communicate with a base station, users can query field estimates at any location. Users may request estimates over a low-resolution grid and then focus on specific areas, allowing for less communication between the group and the base station.

Agents must perform the following four tasks:

- Determine a region of dominance (ROD), the area over which they generate estimates. In this paper an agent's ROD is the agent's the Voronoi region.
- Form a mesh within their (ROD) such that the union of all agents' RODs forms a valid FEM mesh. We modify an existing meshing algorithm for this step.
- 3) Solve the FEM problem. We use the alternating direction method of multipliers (ADMM) algorithm.
- Respond to requests directed at determining the estimate somewhere within the domain. We develop a direction-based depth first search query system.

These tasks can be repeated indefinitely to provide estimates of time varying fields.

Determining the ROD (task 1) and forming the mesh (task 2) roughly correspond to the global FEM steps of dividing the domain (step 1) and finding the interpolating function

(step 2). Task 3, where the agents solve the FEM problem, covers both the assembling (step 3) and solving (step 4) steps of global FEM. One advantage of using ADMM over other solving methods is that assembly happens locally on the agents; individual entries in K and g are never explicitly constructed by any agent. Task 4, querying, is the distributed version of post-processing (step 5).

We now discuss each step of distributed VIM individually. For clarity, we assume that every agent takes one measurement with one sensor, so there are n agents, n measurements, and n sensors; however, this method easily allows agents to take multiple measurements with multiple sensors.

1) Region of Dominance: Each agent establishes a region of dominance (ROD) $\Omega_i \subset \Omega$, partitioning Ω into *n* non-overlapping regions that approximately cover Ω . If two agents have intersecting RODs, they are *neighbors* and must be able to communicate.

We use bounded Voronoi regions for establishing RODs, defined as the set of all points (x, y) such that $||(x, y) - (x_i, y_i)|| < ||(x, y) - (x_j, y_j)||$ and $(x, y) \in \Omega$ for all $i \neq j$. Two agents are Voronoi neighbors if their Voronoi regions share an edge. If the agents know their absolute position and can communicate with their Voronoi neighbors, the algorithm of [19] allows distributed computation of the region. As argued in [19], this communication requirement results in scalable networks because, on average, each agent has, at most, six neighbors.

2) Consistent Mesh: Once the agents know their ROD, they must form a mesh consistent with their neighbors' meshes. Although agents compute their meshes independently, agents with shared edges must place any mesh vertices on those edges at the same location. Assuming a polygonal ROD, every agent first divides its edges into the minimum number of segments such that no segment exceeds some pre-determined maximum length ℓ_{max} . The vertices subdividing these edges will be vertices in the resulting mesh, and, due to symmetry, they will be at the same absolute location for adjacent agents.

Next, each agent runs the meshing algorithm of [18], which generates a mesh with no triangle edge lengths exceeding $\ell_{\rm max}$. If the algorithm adds vertices on the ROD edges, they must be removed. Vertex removal is well-defined on the structure returned by the meshing algorithm.

3) Distributed FEM: After determining its ROD and meshing it, every agent *i* has m_i elements and p_i nodes. Let E_i denote the set of elements contained in agent *i*'s ROD. Since elements are not shared between agents, E_i and E_j are disjoint for $i \neq j$. Thus, each agent has a regional node vector $\bar{q}_i \in \mathbb{R}^{p_i}$ (analogous to q), which is related to the node vectors for each element in its region by

$$q_f = \bar{L}_f \bar{q}_i, \text{ for all } f \in E_i, \tag{16}$$

where the stochastic Boolean matrix \bar{L}_f is analogous to L_e .

Likewise, we form the regional matrices \bar{K}_i and \bar{g}_i analogously to equations (14) and (15) according to

i

$$\bar{K}_i = \sum_{f \in E_i} \bar{L}_f^T K_f \bar{L}_f \tag{17}$$

and

$$\bar{g}_i = \sum_{f \in E_i} \bar{L}_f^T g_f. \tag{18}$$

The map between agent *i*'s node vector and the global node vector is given by g(i, j) such that $[\bar{q}_i]_j$ corresponds to $[q]_{q(i,j)}$, where $[\cdot]_j$ indicates the *j*-th entry in a vector.

Using equations (10), (6), (17), and (18), we derive the following constrained minimization problem:

$$\operatorname{argmin} \sum_{i=1}^{n} \left(\bar{q}_{i}^{T} \bar{K}_{i} \bar{q}_{i} - 2 \bar{q}_{i}^{T} \bar{g}_{i} \right)$$

with respect to \bar{q}_{i} , for $i = 1$ to n

subject to
$$[\bar{q}_i]_j = [q]_{g(i,j)}$$
, for $i = 1$ to n and $j = 1$ to p_i .
(19)

Note that the measurements enter the minimization problem (19) via the term \bar{g}_i . The distributed ADMM method of [8] allows every agent to solve this problem for its own \bar{q}_i . Due to the constraints on \bar{q}_i , the union of all agents solutions is equivalent to solving the global FEM problem on the union of all agents' meshes.

Let $N_{i,j}$ be the set of all pairs (k, l) such that $[\bar{q}_k]_l = [q]_{g(i,j)}$. In other words, $N_{i,j}$ is the set of all agentindex, local-node-index pairs corresponding to the global node index g(i, j). The cardinality of $N_{i,j}$, $|N_{i,j}|$ is then the number of local nodes corresponding to a given global node. To implement ADMM, every agent *i* iterates over three steps. We show a version specific to our problem derived from [8]:

$$[\xi_i(t)]_j = \frac{1}{|N_{i,j}|} \sum_{(k,l) \in N_{i,j}} [\hat{q}_k(t)]_l,$$
(20)

$$\hat{q}_i(t+1) = (\bar{K}_i + \rho I)^{-1} (\bar{g}_i - \nu_i(t) + \rho \xi_i(t)), \qquad (21)$$

$$\nu_i(t+1) = \nu_i(t) + \rho\left(\hat{q}_i(t) - \xi_i(t)\right), \qquad (22)$$

where $\rho > 0$ is an optimization parameter, t is the discretetime index, $\xi_i(t) \in \mathbb{R}^{p_i}$, $\hat{q}_i(t) \in \mathbb{R}^{p_i}$ and $\nu_i(k) \in \mathbb{R}^{p_i}$ are the state variables of the ADMM algorithm. The agents execute this algorithm repeatedly until a stopping condition is met; we use a fixed number of iterations, but adopting the stopping criteria of [8] is left for future work. When the algorithm completes, the agents take new measurements and the process begins again, allowing estimation of changing fields.

The first step, equation (20), averages all of the local node values that correspond to a given entry in q, and is the only step that requires communication. Only nodal values on the boundary of an ROD must be transmitted; nodes on the interior of the ROD directly correspond to a unique entry in q and need not be communicated.

Equation (21) is the primal update step, and $\hat{q}_i(t)$ is agent *i*'s local estimate of \bar{q}_i . The third step, equation (22) updates the dual variable $\nu_i(k)$. Neither the primal nor the dual update steps require communication. When the algorithm converges, the entries of $\xi_i(t)$ and $\hat{q}_i(t)$ converge to the corresponding entries in q, so every agent has a piece of the global FEM solution.

The agents cannot compute equation (20) directly because they do not know g(i, j). However, if agents *i* and *j* are not neighbors they do not share nodes and do not contribute to each others' $\xi_i(t)$ update; thus agents only must communicate with their ROD neighbors. Additionally, as long as neighboring agents know the correspondence between the nodes shared with their neighbors, they can compute (20) by averaging all values corresponding to the same node without knowing the global index g(i, j). Therefore, agents transmit nodal values on their ROD boundary and some book-keeping information that allows the agents to determine which shared nodes correspond to the same global node.

For example, in the worst case, agents can transmit nodal positions along with nodal values. They can then determine which of their neighbors' nodes correspond to the same global node. In a more efficient method, agents group their nodal values by ROD edge, labeling each group with the neighboring agent's identifier and sorting the nodes with in each group in counter-clockwise order relative to the transmitting agent. This technique allows agents to associate the entries in their nodal vector with the nodal values of their neighbors.

Overall, an agent's memory use increases with the number of nodes in its local mesh. For a fixed number of global nodes, adding more agents decreases the number of nodes per agent (and hence its memory usage); alternatively, adding more agents with fixed capabilities allows the number of global nodes to increase. Thus the agents' memory usage scales with respect to environment representation.

Communication per agent does not directly depend on the environmental representation size, but rather it increases with the number of nodes on ROD boundaries. For Voronoi regions, as the number of agents increases, the number of Voronoi edges (and hence communication neighbors) per agent remains (on average) constant [20]. As agent density increases, Voronoi region perimeter decreases, and, for fixed $\ell_{\rm max}$, the number of shared nodes per agent approaches the number of Voronoi vertices for that agent's region. Hence, the communication requirements per agent do not increase with increasing agent density. The solutions obtained by individual agents, however, always match the global solution determined over an equivalent global mesh.

A. Queries

Although agents estimate the field within their own RODs, they do not know the field outside their ROD. A query system allows a user at a base station to retrieve the estimate at any location by asking any agent. A user can make sparse queries over the whole field to obtain a broad estimate and then focus on areas of particular interest. If there are many agents and measurements, making these queries requires significantly less data than sending all measurements back to a central computer, a savings that is compounded if communication with a base station is more costly than agentto-agent communication (e.g., satellite vs. XBee).

Queries consist of the location of the desired estimate. Upon receiving a query, a naive approach employs depth first search: if the point is in the agent's region it returns the estimate, otherwise it forwards the request to one of its neighbors. That neighbor repeats the process, but will only return the request to the initiator after it has exhausted all other neighbors. Since the graph is connected and the agents completely cover the region, an agent with the desired information will eventually be found. This search can usually be made more efficient by always picking the next neighbor based on which is closest to the vector from the current agent to the destination. By alternating between running ADMM iterations until convergence and several rounds of querying, time-varying fields can be monitored if they change slowly. Algorithm 1 displays the query process from a single agent's perspective.

| Algorithm I Query Implementation | | | | | | | |
|----------------------------------|--|--|--|--|--|--|--|
| 1: | neighbors $\leftarrow \{(id, ROD edge),\}$ | | | | | | |
| 2: | $(from, location) \leftarrow receive-query-for-location()$ | | | | | | |
| 3: | if location in ROD then | | | | | | |
| 4: | send-estimate-to(from, my-estimate-at(location)) | | | | | | |
| 5: | else | | | | | | |
| 6: | repeat | | | | | | |
| 7: | to \leftarrow edge-closest-to(location, neighbors) | | | | | | |
| 8: | neighbors \leftarrow remove(neighbors, to) | | | | | | |
| 9: | response \leftarrow send-query(to) | | | | | | |
| 10: | until has-estimate(response) or is-empty(neighbors) | | | | | | |
| 11: | if is-empty(neighbors) then | | | | | | |
| 12: | send-not-found-to(from) | | | | | | |
| 13: | else | | | | | | |
| 14: | send-estimate-to(from, response) | | | | | | |
| 15: | end if | | | | | | |
| 16: | end if | | | | | | |

V. SIMULATIONS

We simulate 18 agents in an environment given by $\bar{\phi}(x,y) = x^4y + x^3y + y^2 + x^2 + 2$ over a unit square centered at (0.5, 0.5). The agents have zero mean Gaussian sensor noise with variance of 0.1. Agents use $\ell_{\max} = 0.25$, $\alpha_0 = 0$, $\alpha_1 = 0.3$, $\rho = 1$, $\mu_i = 1$ and $\phi_0(x, y) = 0$. Figure 2 depicts the agents' Voronoi regions, their meshes, their estimates, and a location that the base station queried after running the ADMM algorithm for 8000 time-steps. Figure 3 depicts the actual field and Figure 4 shows the error, averaged over 100 Monte-Carlo runs. The distributed solution and standard VIM solution over the same mesh yield the same results. The query was initially directed at agent zero, and the arrows show the chain of queries needed to determine the value at (0.7, 0.6). Due to noise and because no interpolation method is perfect, the actual and estimated values differ.

VI. CONCLUSION

We have introduced distributed VIM (DVIM), which uses a distributed FEM solver based on ADMM to generate environment estimates. Although the estimates generated match a centralized solution, no single agent has a full environment representation. The scaling properties of the algorithm mean



Fig. 2. The agents estimates, the mesh, their Voronoi region, and the path of a query. Agent (and measurement) locations are marked with blue dots, with the numbers providing the agent's identifier.



Fig. 3. The actual field.

that adding more agents more densely in the environment does not increase the communication or memory burdens of individual agents. More densely distributed agents, however, provide more detailed and accurate environment estimates. A query system allows a user to determine the environment estimate at any given location. Future work includes developing control laws for mobile DVIM agents, estimating an error field from VIM (using the techniques of [4], [6], [21]), and explicitly incorporating time-varying field models into the estimation.

REFERENCES

- D. M. Glover, W. J. Jenkins, and S. C. Doney, *Modeling Methods for* Marine Science. Cambridge: Cambridge University Press, 2011.
- [2] 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.
- [3] 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
- [4] C. Troupin, A. Barth, D. Sirjacobs, M. Ouberdous, J.-M. Brankart, P. Brasseur, M. Rixen, A. Alvera-Azcrate, 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. 5253, no. 0, pp. 90 – 101, 2012.



Fig. 4. Absolute value of the error between the estimate and the actual field averaged over 100 Monte-Carlo runs.

- [5] A. Barth, A. A. Azcárate, P. Joassin, J.-M. Becers, and C. Troupin, *Introduction to Optimal Interpolation and Variational Analysis*, Geo-Hydrodynamics and Environment Research (GHER), 2008.
- [6] J. Brankart and P. Brasseur, "The general circulation in the mediterranean sea: a climatological approach," *Journal of Marine Systems*, vol. 18, no. 13, pp. 41 – 70, 1998.
- [7] J. Fish and T. Belytschko, A First Course in Finite Elements. John Wiley & Sons, 2007.
- [8] 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.
- [9] 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, pp. 14/1–28, 2011.
- [10] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989.
- [11] S. Martinez, "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.
- [12] 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.
- [13] J. Cortes, "Distributed kriged kalman filter for spatial estimation," *Automatic Control, IEEE Transactions on*, vol. 54, no. 12, pp. 2816– 2827, Dec 2009.
- [14] 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.
- [15] G. Battistelli, L. Chisci, N. Forti, S. Selleri, and G. Pelosi, "Decentralized consensus finite-element kalman filter for field estimation," *CoRR*, vol. abs/1604.02392, 2016.
- [16] S. Grime and H. Durrant-Whyte, "Data fusion in decentralized sensor networks," *Control Engineering Practice*, vol. 2, no. 5, pp. 849 – 863, 1994.
- [17] G. Battistelli, L. Chisci, N. Forti, G. Pelosi, and S. Selleri, "Distributed finite element kalman filter," in *Control Conference (ECC)*, 2015 *European*, July 2015, pp. 3695–3700.
- [18] L. Rineau, "2D conforming triangulations and meshes," in CGAL User and Reference Manual, 4.7 ed. CGAL Editorial Board, 2015.
- [19] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," in *Robotics and Automation*, 2002. *Proceedings. ICRA '02. IEEE International Conference on*, vol. 2, 2002, pp. 1327–1332.
- [20] 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.
- [21] P. C. McIntosh, "Oceanographic data interpolation: Objective analysis and splines," *Journal of Geophysical Research: Oceans*, vol. 95, no. C8, pp. 13529–13541, 1990.