INTERNATIONAL JOURNAL OF ROBUST AND NONLINEAR CONTROL
*Int. J. Robust Nonlinear Control* 2009; **1**:1–2 *Prepared using* rncauth.cls *[Version: 2002/11/11 v1.00]*

# Cooperative adaptive sampling of random fields with partially known covariance

Rishi Graham[1,*] and Jorge Cortés[2]

[1] *Department of Applied Mathematics and Statistics, University of California, Santa Cruz*
[2] *Department of Mechanical and Aerospace Engineering, University of California, San Diego*

SUMMARY

This paper considers autonomous robotic sensor networks taking measurements of a physical process for predictive purposes. The physical process is modeled as a spatiotemporal random field. The network objective is to take samples at locations that maximize the information content of the data. The combination of information-based optimization and distributed control presents difficult technical challenges as standard measures of information are not distributed in nature. Moreover, the lack of prior knowledge on the statistical structure of the field can make the problem arbitrarily difficult. Assuming the mean of the field is an unknown linear combination of known functions and its covariance structure is determined by a function known up to an unknown parameter, we provide a novel distributed method for performing sequential optimal design by a network comprised of static and mobile devices. We characterize the correctness of the proposed algorithm and examine in detail the time, communication, and space complexities required for its implementation. Copyright © 2009 John Wiley & Sons, Ltd.

KEY WORDS: robotic network, spatial estimation, stochastic process, Bayesian learning

## 1. Introduction

Networks of environmental sensors are playing an increasingly important role in scientific studies, with applications to a variety of scenarios, including detection of chemical pollutants, animal monitoring, and mapping of ocean currents. Among their many advantages, robotic sensor networks can improve

*Correspondence to: Rishi Graham at `rishig@ams.ucsc.edu`

Copyright © 2009 John Wiley & Sons, Ltd.

the efficiency of data collection, adapt to changes in the environment, and provide a robust response to individual failures. The design of coordination algorithms for networks performing these spatially-distributed sensing tasks faces the major challenge of incorporating the complex statistical techniques that come into play in the analysis of environmental processes. Traditional statistical modeling and inference assume full availability of all measurements and central computation. While the availability of data at a central location is certainly a desirable property, the paradigm for motion coordination builds on partial, fragmented information. Coordination algorithms need to be distributed and scalable to make robotic networks capable of operating in an autonomous and robust fashion. At the same time, these algorithms must be statistically driven to steer the network towards locations that provide the most informative samples about the physical processes. This work is a step forward in bridging the gap between sophisticated statistical modeling and distributed motion coordination.

In this paper, we are interested in a predictive scenario, in which the goal is to estimate the value of a physical process at unmeasured locations based on the samples collected. We assume that the model of the underlying process is stochastic. This can be a useful approach if the dynamics of the field are not well known, require a high dimensional parameter space to model, or require extremely accurate specification of initial conditions. Treating the field as a Gaussian Process, we use a Bayesian modeling approach, in which the prediction at any given location takes the form of a *distribution*, with a full accounting of the predictive uncertainty. When choosing the locations to take samples, this uncertainty should inform any measure of their utility. This presents a difficult challenge in a distributed setting, because the predictive uncertainty depends on all samples in a nontrivial way.

*Literature review* Complex statistical techniques allow a detailed account of uncertainty in modeling physical phenomena. Of particular relevance to this work are [1], regarding statistical models, and [2, 3], regarding the application of optimal design techniques to Bayesian models. Under certain conditions on the covariance structure, data taken far from the prediction site have very little impact on the predictor [4]. When the random field does not have a covariance structure with finite spatial correlation, an approximation which does may be generated via covariance tapering [5]. Optimal design [6, 7] addresses the problem of choosing sample locations which optimize estimation.

In cooperative control, various works consider mobile sensor networks performing spatial estimation tasks. [8] introduces performance metrics for oceanographic surveys by autonomous underwater vehicles. [9] considers a robotic sensor network with centralized control estimating a static field from measurements with both sensing and localization error. [10] chooses optimal sampling trajectories from a parameterized set of paths. [11] discusses the tracking of level curves in a noisy scalar field. [12] develops distributed estimation techniques for predictive inference of a spatiotemporal random field and its gradient. In [13, 14, 15, 16] the focus is on estimating deterministic fields with random measurement noise. Of the above references, those which consider random field models do so under an assumption of known covariance. To our knowledge this is the first work in the cooperative control arena which allows for uncertainty in the covariance as well as the mean. We make use of a model derived in [17], which is the only spatial model we are aware of that makes a direct analytical connection between uncertainty in the covariance and the resulting predictive uncertainty. Aside from this model or derivatives, the common practice when confronted with unknown covariance is to either run a separate estimation procedure and then treat the covariance as known, or to use simulation methods such as Markov chain Montecarlo to estimate the posterior distribution. The work [18] addresses a method of choosing sample locations from a discrete space which are robust to misspecification of the covariance. Another method for handling unknown covariance has recently grown out of the exploration-exploitation approach of reinforcement learning (see, e.g. [19]). The work [20] applies this approach to the spatial estimation scenario by breaking up the objective into an exploration component which focuses on learning about the model in a discretized space and an exploitation component in which that knowledge is put to use in optimizing for prediction. Here, we provide a result in which no discretization is necessary and we take full advantage of the mobile capabilities of networks of autonomous sensors. Our work is based in part on previous material presented in [21] and [22].

*Statement of contributions* We begin with a widely accepted Bayesian model for the prediction of a spatiotemporal random field, designed to handle various degrees of knowledge about the mean and covariance. The predictive variance of this model can be written as a scaled product of two components, one corresponding to uncertainty about the covariance of the field, the other corresponding to uncertainty of the prediction conditional on the covariance. Our first contribution is the development of an approximate predictive variance which may be calculated efficiently in a sequential and distributed manner. This includes introducing a scheduled update of the estimated covariance parameter based on uncorrelated clusters of samples. Our second contribution is the characterization of the smoothness properties of the objective function and the computation of its gradient. Using consensus and distributed

Jacobi overrelaxation algorithms, we show how the objective function and its gradient can be computed in a distributed way across a network composed of robotic agents and static nodes. Our third contribution is the design of a coordination algorithm based on projected gradient descent which guarantees one-step-ahead locally optimal data collection. Due to the nature of the solution, optimality here takes into account both the unknown parameter in the covariance and the (conditional) uncertainty in the prediction. Finally, our fourth contribution is the characterization of the communication, time, and space complexities of the proposed algorithm. For reference, we compare these complexities against the ones of a centralized algorithm in which all sample information is broadcast throughout the network at each step of the optimization.

*Organization* Section 2 introduces basic notation and describes the statistical model. Section 3 states the robotic network model and the overall network objective. The following two sections present the main results of the paper. Section 4 introduces the objective function, with attention to its smoothness properties, and discusses how the network can make the required calculations in a distributed way. Section 5 presents the cooperative strategy for optimal data collection along with correctness and complexity results. Section 6 contains our conclusions.

## 2. Preliminary notions

Let $\mathbb{R}$, $\mathbb{R}_{>0}$, and $\mathbb{R}_{\geq 0}$ denote the set of reals, positive reals and nonnegative reals, respectively. We consider a convex polytope $\mathcal{D} \subset \mathbb{R}^d$, $d \in \mathbb{N}$. Let $\mathcal{D}_e = \mathcal{D} \times \mathbb{R}$ denote the space of points over $\mathcal{D}$ and time. For $p \in \mathbb{R}^d$ and $r \in \mathbb{R}_{>0}$, let $\overline{B}(p, r)$ be the *closed ball* of radius $r$ centered at $p$. We denote by $\lfloor x \rfloor$, respectively $\lceil x \rceil$ the floor, respectively ceiling of $x \in \mathbb{R}$. Given $\underline{u} = (u_1, \dots, u_a)^T$, $a \in \mathbb{Z}_{>0}$, and $\underline{v} = (v_1, \dots, v_b)^T$, $b \in \mathbb{Z}_{>0}$, we denote by $(\underline{u}, \underline{v})$ the concatenation $(\underline{u}, \underline{v}) = (u_1, \dots, u_a, v_1, \dots, v_b)^T$. We denote by $\partial S$ the boundary of a set $S$. Let $i_{\mathbb{F}} : (\mathbb{R}^d)^n \to \mathbb{F}(\mathbb{R}^d)$ be the natural immersion, i.e., $i_{\mathbb{F}}(P)$ contains only the distinct points in $P = (p_1, \dots, p_n)$. Note that $i_{\mathbb{F}}$ is invariant under permutations of its arguments and that the cardinality of $i_{\mathbb{F}}(p_1, \dots, p_n)$ is in general less than or equal to $n$. The $\epsilon$-*contraction* of a set $S$, with $\epsilon > 0$, is the set $S_\epsilon = \{q \in S \mid \mathrm{d}(q, \partial S) \geq \epsilon\}$. A *convex polytope* is the convex hull of a finite point set. For a bounded set $S \subset \mathbb{R}^d$, we let $\mathrm{CR}(S)$ denote the *circumradius* of $S$, that is, the radius of the smallest-radius $d$-sphere enclosing $S$. We denote by $\mathbb{F}(S)$ the collection of finite subsets of $S$. The *Voronoi partition* $\mathcal{V}(\underline{s}) = (V_1(\underline{s}), \dots, V_n(\underline{s}))$ of $\mathcal{D}$ generated by the points

$\underline{s} = (s_1, \dots, s_n)$ is defined by $V_i(\underline{s}) = \{q \in \mathcal{D} \mid \|q - s_i\| \leq \|q - s_j\|, \ \forall j \neq i\}$. Each $V_i(\underline{s})$ is called a *Voronoi cell*. Two points $s_i$ and $s_j$ are *Voronoi neighbors* if their Voronoi cells share a boundary.

We use $\lambda_{\text{min}}(A)$, respectively $\lambda_{\text{max}}(A)$ to denote the smallest, respectively largest eigenvalue of the square matrix $A$, and $\text{sprad}(A)$ to denote the spectral radius of $A$. We denote by $[A]_{ij}$ the $(i,j)$th element of the matrix $A$, and by $\text{col}_i(A)$ its $i$th column. Let $\mathbf{0}_{i \times j}$ denote the $i \times j$ zero matrix. If the dimensions are clear from the context we may omit the subscripts and use $\mathbf{0}$. Given a partitioned matrix, $A = \begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}$, we denote by $(A_{11} \,|A)$, respectively $(A_{22} \,|A)$ the Schur complement of $A_{11}$, respectively $A_{22}$ in $A$, i.e.,

$$(A_{11} \,|A) = A_{22} - A_{21}A_{11}^{-1}A_{12} \qquad \text{and} \qquad (A_{22} \,|A) = A_{11} - A_{12}A_{22}^{-1}A_{21}.$$

## 2.1. *Projected gradient descent*

Next, we describe the constrained optimization technique known as projected gradient descent [23] to iteratively find the minima of an objective function $F : \mathbb{R}^m \to \mathbb{R}_{\geq 0}$. Let $\Omega$ denote a nonempty, closed, and convex subset of $\mathbb{R}^m$, $m \in \mathbb{N}$. Assume that $\nabla F$ is globally Lipschitz on $\Omega$. Let $\text{proj}_\Omega : \mathbb{R}^m \to \Omega$ denote the orthogonal projection onto the set $\Omega$,

$$\text{proj}_\Omega(x) = \underset{y \in \Omega}{\text{argmin}} \, \|x - y\|.$$

Consider a sequence $\{x_k\} \in \Omega$, $k \in \mathbb{N}$, which satisfies

$$x_{k+1} = \text{proj}_\Omega \left( x_k - a_k \nabla F(x_k) \right), \, x_1 \in \Omega, \tag{1}$$

where the step size, $a_k$, is chosen according to the LINE SEARCH ALGORITHM described in Table I, evaluated at $x = x_k$.

Here the grid size $\tau$ determines the granularity of the line search. The tolerance $\theta$ may be adjusted for a more (larger $\theta$) or less (smaller $\theta$) strict gradient descent. With $\theta > 0$, the LINE SEARCH ALGORITHM must terminate in finite time. The Armijo condition (step 7) ensures that the decrease in $F$ is commensurate with the magnitude of its gradient. A sequence $\{x_k\}_{k=1}^{\infty}$ satisfying these requirements converges in the limit [23] as $k \to \infty$ to stationary points of $F$.

## 2.2. *Bayesian modeling of space-time processes*

Let $Z$ denote a random space-time process taking values on $\mathcal{D}_e$. Let $\underline{y} = (y_1, \dots, y_n)^T \in \mathbb{R}^n$ be $n \in \mathbb{N}$ measurements taken from $Z$ at corresponding locations $\underline{x} = (x_1, \dots, x_n)^T \in \mathcal{D}_e^m$, with

```
Name:     LINE SEARCH ALGORITHM
Goal:     Determine step size for algorithm (1)
Input:    $x \in \Omega$
Assumes:  $\tau, \theta \in (0,1)$, max step $\alpha_{\text{max}} \in \mathbb{R}_{>0}$
Output:   $\alpha \in \mathbb{R}_{\geq 0}$
1: $\alpha = \alpha_{\text{max}}$
2: repeat
3:     $x_{\text{new}} = \text{proj}_{\Omega}\left(x - \alpha \nabla F(x)\right)$
4:     $\nu = \frac{\theta}{\alpha}\|x - x_{\text{new}}\|^2 + F(x_{\text{new}}) - F(x)$
5:     if $\nu > 0$ then
6:         $\alpha = \alpha\tau$
7: until $\nu \leq 0$
```

Table I. LINE SEARCH ALGORITHM.

$x_i = (s_i, t_i),\ i \in \{1, \dots, n\}$. Given these data, various models allow for prediction of $Z$ at any point in $\mathcal{D}_e$, with associated uncertainty. Optimal design is the process of choosing locations to take measurements in order to reduce the uncertainty of the resulting statistical prediction. Since prediction uncertainty drives the problem, it should be modeled as accurately as possible.

In a Bayesian setting, the prediction takes the form of a distribution, called the posterior predictive [24]. If the field is modeled as a Gaussian Process with known covariance, the posterior predictive mean corresponds to the *Best Linear Unbiased Predictor*, and its variance corresponds to the mean-squared prediction error. Predictive modeling in this context is often referred to in geostatistics as *simple kriging* if the mean is also known, or *universal kriging* if the mean is treated as an unknown linear combination of known basis functions. If the covariance of the field is not known, however, few analytical results exist which take the full uncertainty into account. We present here a model [1] which allows for uncertainty in the covariance process and still produces an analytical posterior predictive distribution. We assume that the measurements are distributed as the $m$-variate normal distribution,

$$\underline{y} \sim \text{N}_n\left(\mathbf{F}^T\beta, \sigma^2\mathbf{K}\right). \tag{2}$$

Here $\beta \in \mathbb{R}^p$ is a vector of unknown regression parameters, $\sigma^2 \in \mathbb{R}_{>0}$ is the unknown variance parameter, and $\mathbf{K}$ is a correlation matrix whose $i,j$th element is $\mathbf{K}_{ij} = \text{Cor}[y_i, y_j]$. Note that $\mathbf{K}$ is symmetric, positive definite, with 1's on the diagonal. We assume a finite correlation range in space, $r_\text{s} \in \mathbb{R}_{>0}$, and in time, $r_\text{t} \in \mathbb{R}_{>0}$, such that if $\|s_i - s_j\| \geq r_\text{s}$ or $|t_i - t_j| \geq r_\text{t}$, then $\mathbf{K}_{ij} = \mathbf{K}_{ji} = 0$.

The matrix $\mathbf{F}$ is determined by a set of $p \in \mathbb{N}$ known basis functions $f_i : \mathcal{D}_e \to \mathbb{R}$ evaluated at the locations $\underline{x}$, i.e.,

$$\mathbf{F} = \begin{bmatrix} f_1(x_1) & \dots & f_1(x_n) \\ \vdots & \ddots & \vdots \\ f_p(x_1) & \dots & f_p(x_n) \end{bmatrix}.$$

We will also use $\mathbf{f}(x) = (f_1(x), \dots, f_p(x))^T \in \mathbb{R}^p$ to denote the vector of basis functions evaluated at a single location. To ensure an analytical form for the posterior predictive distribution, we assume conjugate prior distributions for the parameters,

$$\beta|\sigma^2 \sim \mathrm{N}_p\left(\beta_0, \sigma^2 \mathbf{K}_0\right), \tag{3a}$$

$$\sigma^2 \sim \Gamma^{-1}\left(\frac{\nu}{2}, \frac{q\nu}{2}\right). \tag{3b}$$

Here $\beta_0 \in \mathbb{R}^p$, $\mathbf{K}_0 \in \mathbb{R}^{p \times p}$, and $q, \nu \in \mathbb{R}_{>0}$ are constants, known as *tuning parameters* for the model, and $\Gamma^{-1}(a, b)$ denotes the inverse gamma distribution with shape parameter $a$ and scale parameter $b$ (see, e.g. [25]). Since it is a correlation matrix, it should be noted that $\mathbf{K}_0$ must be positive definite. A common practice in statistics is to use $\mathbf{K}_0 \propto \boldsymbol{I}$.

**Proposition 2.1 (Posterior predictive distribution)** *Under the Bayesian model* (2)*, the posterior predictive at location $x_0 \in \mathcal{D}_e$ is a shifted Students t distribution (see, e.g. [25]) with $\nu + n$ degrees of freedom, which takes the form, for $Z = Z(x_0)$,*

$$Z|\underline{y}, \underline{x} \propto \mathrm{Var}[Z|\underline{y}, \underline{x}]^{-\frac{1}{2}} \left(1 + \frac{\left(Z - \mathrm{E}[Z|\underline{y}, \underline{x}]\right)^2}{(\nu + n - 2)\,\mathrm{Var}[Z|\underline{y}, \underline{x}]}\right)^{-\frac{\nu+n+1}{2}}.$$

*Here, the expectation is given by*

$$\mathrm{E}[Z|\underline{y}, \underline{x}] = \left(\mathbf{f}(x_0) - \mathbf{F}\mathbf{K}^{-1}\mathbf{k}\right)^T \beta^\dagger + \mathbf{k}^T \mathbf{K}^{-1} \underline{y},$$

$$\beta^\dagger = (\mathbf{E} + \mathbf{K}_0^{-1})^{-1} \left(\hat{\beta} + \mathbf{K}_0^{-1}\beta_0\right),$$

*where $\hat{\beta} = \mathbf{E}^{-1}\mathbf{F}\mathbf{K}^{-1}\underline{y}$, $\mathbf{E} = \mathbf{F}\mathbf{K}^{-1}\mathbf{F}^T$, and $\mathbf{k} = \mathrm{Cor}[\underline{y}, Z] \in \mathbb{R}^n$. The variance is given by*

$$\begin{aligned} \mathrm{Var}[Z|\underline{y}, \underline{x}] &= \varphi(\underline{y}, \underline{x})\phi(x_0; \underline{x}), \\ \phi(x_0; \underline{x}) &= \mathrm{Cor}[Z, Z] - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k} + \xi_0^T \left(\mathbf{K}_0^{-1} + \mathbf{E}\right)^{-1} \xi_0, \\ \xi_0 &= \mathbf{f}(x_0) - \mathbf{F}\mathbf{K}^{-1}\mathbf{k}, \\ \varphi(\underline{y}, \underline{x}) &= \frac{1}{\nu + n - 2} \left(q\nu + (\underline{y} - \mathbf{F}^T\beta_0)^T \left(\mathbf{K} + \mathbf{F}^T\mathbf{K}_0\mathbf{F}\right)^{-1} (\underline{y} - \mathbf{F}^T\beta_0)\right). \end{aligned}$$

**Proof.** This result can be obtained from application of Bayes Theorem to the model outlined by Equations (3), or from results in [1] and [17] using a technique similar to the one used in the proof of Proposition I.4 in Appendix I. ■

Note that since $\mathbf{K}_0$ and $\mathbf{K}$ are positive definite, the quantities $\phi(x_0; \underline{x})$ and $\varphi(\underline{y}, \underline{x})$ are well posed.

**Remark 2.2 (Terms in the posterior predictive variance)** Note the form for the posterior predictive variance in Proposition 2.1 as a product of two terms. The first term, $\varphi(\underline{y}, \underline{x})$, is the posterior mean of the parameter $\sigma^2$, given the sampled data. We will call it the *sigma mean*. The second term, $\phi(x_0; \underline{x})$, can be thought of as the scaled posterior predictive variance conditioned on $\sigma^2$. We will call it the *conditional variance*. •

The conditional variance is very close to what the predictive variance would look like if $\sigma^2$ were known, as we show in the next. The following results may be derived by applying Bayes Theorem to the prior model.

**Proposition 2.3 (Kriging variance)** *If the variance parameter $\sigma^2$ is known, the result is the* universal kriging predictor, *and the posterior predictive variance takes the form,*

$$\mathrm{Var}_{UK}[Z|\underline{y}, \underline{x}] = \sigma^2 \left( \mathrm{Cor}[Z, Z] - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k} + \xi_0^T \mathbf{E}^{-1} \xi_0 \right).$$

*If, in addition, the mean of the field is known, the result is the* simple kriging predictor, *and the posterior predictive variance is given by,*

$$\mathrm{Var}_{SK}[Z|\underline{y}, \underline{x}] = \sigma^2 \left( \mathrm{Cor}[Z, Z] - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k} \right).$$

**Remark 2.4 (Extension of subsequent results to Kriging)** The simple and universal kriging results are simplified versions of our overall model, and results from the rest of this paper may be applied to those models with minimal modifications. An exception is that when approximating $\mathrm{Var}_{\mathrm{UK}}$ using subsets of measurements, care must be taken to ensure well-posedness. Specifically, an assumption that $n > p$ is required to ensure that the matrix $\mathbf{E}$ is nonsingular. •

## 2.3. Distributed computational tools

Here we briefly describe some tools for distributed computations. Consider a network of $n$ agents with limited communication capabilities. Assume that the undirected communication graph, $G$, is connected (i.e., there is a two way communication path connecting any two agents). Let $a_{ij} \in \{0, 1\}, \ i, j \in$

$\{1,\dots,n\}$ be $1$ if agents $i$ and $j$ are neighbors in $G$, and $0$ otherwise. Let $b = (b_1,\dots,b_n)^T \in \mathbb{R}^n$, $C = [c_{ij}] \in \mathbb{R}^{n\times n}$, and assume agent $i$ knows $b_i$ and the $i$th row of $C$. Additionally assume that $c_{ii} \neq 0$ and for $i \neq j$, $c_{ij} \neq 0$ iff $i$ and $j$ are communication neighbors. Then the following results hold.

**JOR:** The network can compute the vector $y = C^{-1}b$ via a *distributed Jacobi overrelaxation* algorithm [26], formulated as the discrete-time dynamical system,

$$y_i(l+1) = (1-h)y_i(l) - \frac{h}{c_{ii}}\Big(\sum_{j\neq i} c_{ij}y_j(l) - b_i\Big), \tag{4}$$

for $l \in \mathbb{Z}_{\geq 0}$ and $i \in \{1,\dots,n\}$, where $y(0) \in \mathbb{R}^n$ and $h \in \left(0, \frac{2}{\lambda_{\max}(C)}\right)$. At the end of the algorithm, agent $i$ knows the $i$th element of $C^{-1}b$.

**Discrete-time average consensus:** The network can compute the arithmetic mean of elements of $b$ via the discrete dynamical system [27],

$$x_i(l+1) = x_i(l) + \epsilon\sum_{j\neq i} a_{ij}(x_i(l) - x_j(l)),\ x(0) = b,$$

where $\epsilon \in (0, \frac{1}{\Delta})$ and $\Delta = \max_{i\in\{1,\dots,n\}}\left\{\sum_{j\neq i} a_{ij}\right\}$ is the maximum degree of the network. At the end of the algorithm, all agents know $\frac{\sum_{i=1}^n b_i}{n}$.

**Maximum consensus:** The network can calculate the maximum value of elements of $b$ via a leader election algorithm [28]. Each agent sends the current estimate of the maximum to all neighbors, then updates its estimate. If the process is repeated a number of times equal to the diameter of the network, then every agent will know the maximum.

The first two results above are only exact asymptotically, but convergence is exponential with time.

## 3. Problem statement

Here we introduce the model for the group of robotic agents and static nodes, and detail the overall objective.

### 3.1. *Robotic sensor network model*

Consider a group $\{S_1,\dots,S_m\}$ of $m \in \mathbb{N}$ static nodes at locations $Q = (q_1,\dots,q_m)^T \in \mathcal{D}^m$. Assume that each node has a limited communication radius, $R \in \mathbb{R}_{>0}$, and that they are positioned so that each one can communicate with its Voronoi neighbors. In addition to the static nodes, consider a group $\{R_1, \dots, R_n\}$ of $n$ mobile robotic sensor agents. The position of robot $i \in \{1, \dots, n\}$ at time $t \in \mathbb{R}$ is denoted by $p_i(t) \in \mathcal{D}$. The robots take point samples of the spatial field at discrete instants of time in $\mathbb{Z}_{\geq 0}$. Between sample instants, each robot moves according to the discrete dynamics

$$p_i(k+1) = p_i(k) + u_i(k),$$

where $\|u_i\| \leq u_{\text{max}}$ for some $u_{\text{max}} \in \mathbb{R}_{>0}$. The communication radius of the robotic agents is also $R$. Each node will need to be able to communicate with any robot which may be within covariance range of the points in its Voronoi region at the following timestep. To that end, we assume that

$$R \geq \max_{i \in \{1,\dots,m\}} \{\mathrm{CR}(V_i(Q))\} + r_{\text{s}} + u_{\text{max}}. \tag{5}$$

The robots can sense the positions of other robots within a distance of $2u_{\text{max}}$. At discrete timesteps, each robot communicates the sample and location to static nodes within communication range, along with the locations of any other sensed robots. The nodes then compute control vectors, and relay them back to robots within communication range. The implementation does not require direct communication between robots. We refer to this network model as $\mathcal{N}$.

*3.1.1. Voronoi contraction for collision avoidance* We begin by specifying the region of allowed movement for the robotic agents. In addition to the maximum velocity and the restriction to $\mathcal{D}$, we impose a minimum distance requirement between robots. Beyond the benefit of collision avoidance, this restriction ensures that even under the assumption of zero sensor error, the posterior predictive variance is well-defined over the space of possible configurations.

Let $\omega \in \mathbb{R}_{>0}$ be a desired buffer width, assumed to be small compared to the size of $\mathcal{D}$. To ensure that the distance between two robots is never smaller than $\omega$, we introduce a contraction of the Voronoi diagram. Consider the locations $P = (p_1, \dots, p_n)$ of the $n$ robotic agents at the $k$th timestep. Define $\Omega_i^{(k)} = (V_i(P))_{\omega/2} \cap \overline{B}(p_i, u_{\text{max}})$, where $(V_i(P))_{\omega/2}$ denotes the $\frac{\omega}{2}$-contraction of $V_i(P)$. For each $j \neq i \in \{1, \dots, n\}$, we have $\mathrm{d}(\Omega_i^{(k)}, \Omega_j^{(k)}) \geq \omega$. Between timesteps $k$ and $k+1$, we restrict $R_i$ to the region $\Omega_i^{(k)}$. Figure 1 shows an example in $\mathbb{R}^2$ of this set. We denote by $\Omega^{(k)} = \prod_{i=1}^n \Omega_i^{(k)} \subset (\mathbb{R}^d)^n$ the region of allowed movement of all robotic agents at timestep $k \in \mathbb{Z}_{\geq 0}$. Note that $\Omega^{(k)}$ is closed, bounded, and convex.

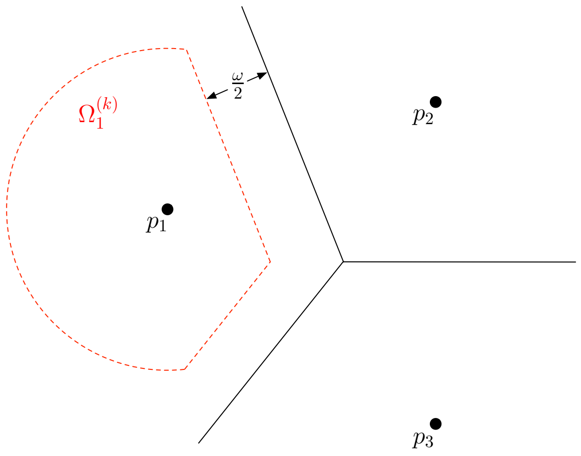


Figure 1. Example of region $\Omega_1^{(k)}$ (red) with Voronoi partition boundaries (black) for comparison.

## *3.2. The average variance as objective function*

For predictions over a region in space and time, the average variance is a natural measure of uncertainty. Using Proposition 2.1, we define the average over the spatiotemporal region of the posterior predictive variance,

$$\mathcal{A} = \varphi(\underline{y}, \underline{x}) \int_{\mathcal{D}} \int_{T} \phi((s_0, t_0); \underline{x})\, dt_0\, ds_0. \tag{6}$$

Here, $\underline{y} \in (\mathbb{R}^n)^{k_{\max}}$ is a sequence of samples taken at discrete times $\{1, \ldots, k_{\text{max}}\}$, $k_{\text{max}} \in \mathbb{Z}_{>0}$, at space-time locations $\underline{x} \in (\mathcal{D}_e^n)^{k_{\max}}$. $T = [1, k_{\text{max}}]$ is the time interval of interest.

One would like to choose the sample locations that minimize $\mathcal{A}$. Since samples are taken sequentially, with each new set restricted to a region nearby the previous, and since the sigma mean depends on the actual values of the samples, one cannot simply optimize over $(\mathcal{D}_e^n)^{k_{\max}}$ a priori.

Consider, instead, a greedy approach in which we use past samples to choose the positions for the next ones. At each timestep we choose the next locations to minimize the average posterior variance of the predictor given the data known so far. In Section 4, we develop a sequential formulation of the average posterior predictive variance and discuss its amenability to distributed implementation over $\mathcal{N}$.

## 4. Distributed criterion for adaptive design

In this section we develop an optimality criterion to maximally reduce the average predictive variance at each timestep. We begin by introducing some notation that will help us make the discussion precise.

Let $\underline{y}^{(k)} \in \mathbb{R}^n$, $k \in \{1, \dots, k_{\text{max}}\}$, denote the samples taken at time step $k$, at locations $\underline{x}^{(k)} \in \mathcal{D}_e^n$. Let $\underline{y}^{(k_1:k_2)} \in \mathbb{R}^{n(k_2-k_1+1)}$, $k_1 < k_2$ denote the vector of samples taken over a range of timesteps, at locations $\underline{x}^{(k_1:k_2)} \in \mathcal{D}_e^{n(k_2-k_1+1)}$, i.e., $\underline{y}^{(k_1:k_2)} = \big(\big(\underline{y}^{(k_1)}\big)^T, \dots, \big(\underline{y}^{(k_2)}\big)^T\big)^T$ and $\underline{x}^{(k_1:k_2)} = \big(\big(\underline{x}^{(k_1)}\big)^T, \dots, \big(\underline{x}^{(k_2)}\big)^T\big)^T$. At step $k$, the samples $\underline{y}^{(1:k)}$ have already been taken. We are interested in choosing locations, $P \in \Omega^{(k)}$, at which to take the next samples. To that end, let $\underline{x}^{(1:k+1)} : \Omega^{(k)} \to \mathcal{D}_e^{n(k+1)}$ map a new set of spatial locations to the vector of spatiotemporal locations which will result if the $(k+1)$st samples are taken there, i.e., $\underline{x}^{(1:k+1)}(P) = \big((\underline{x}^{(1:k)})^T, (P, k+1)^T\big)^T$. The adaptive design approach is then to use the samples that minimize the average prediction variance *so far*,

$$\mathcal{A}^{(k)}(P) = \varphi\big(\underline{y}^{(1:k+1)}, \underline{x}^{(1:k+1)}(P)\big) \int_{\mathcal{D}} \int_{T} \phi\big((s_0, t_0); \underline{x}^{(1:k+1)}(P)\big)\, dt_0\, ds_0. \tag{7}$$

This sequential formulation of the problem allows us to use past measurements without worrying about the ones at steps after $k+1$. However, efficient distributed implementation still suffers from three major obstacles. First, both terms in $\mathcal{A}^{(k)}(P)$ require inversion of the $n(k+1) \times n(k+1)$ covariance matrix of all sample locations up to step $k+1$. The complexity of this inversion grows with $k^2$, which quickly becomes an unreasonable burden. Second, the sigma mean also depends on the actual values of the samples at step $k+1$, which we do not know until the measurements are taken. In Section 4.1 we discuss an approximate sigma mean which addresses these issues. The third obstacle to our distributed implementation is the integration of the conditional variance over the entire spatial region. This integral involves a complex interaction between every sample and the entire predictive region, which does not lend itself to distributed calculation. In Section 4.2, we introduce a distributed approximation based on regional integrals, each of which may be calculated locally. Finally, Section 4.3 combines these results to propose an approximate average prediction variance that can be optimized via gradient descent.

### *4.1. Approximate sigma mean*

In this section, we describe our approach to deal with the term $\varphi\left(\underline{y}^{(1:k+1)}, \underline{x}^{(1:k+1)}(P)\right)$ in (7). We would like an approximation which may be calculated before the samples are taken and that scales with increasing $k$. This involves two steps. First, we develop a running estimate of $\varphi$ which uses scheduled updates based on a subset of the overall samples. Second, we address the issue of unrealized sample values by using a generalized least squares estimate.

*4.1.1. Incorporating new data.* From Lemma I.6 in Appendix I, it can be seen that minimizing $\varphi$ results in minimal values of both the posterior mean and variance of $\sigma^2$. As the number of samples increases, the posterior mean of $\sigma^2$ should approach some finite constant. Under the assumption of a Gaussian Process prior, any finite vector of realizations has a consistent joint distribution. This implies that if we evaluate the posterior mean with any subset of the samples, the result should asymptotically approach the same constant, regardless of which samples are used for inference. Using samples from all timesteps results in quick convergence, but requires a large computational burden later on, with little information gain. Instead, we consider a regular, scheduled update, using select blocks of measurements to perform the estimation, as described next.

Let $t_{\text{skip}} \in \mathbb{N}$ be a fixed number of timesteps to skip between updates, and let $t_{\text{blk}} \in \mathbb{N}$ denote a fixed number of steps to include in our estimation block. Since data sampled at different time lags provide different information about $\sigma^2$, a natural choice is blocks of samples covering $t_{\text{blk}} = \lceil r_{\text{t}} \rceil$ timesteps. To simplify notation, we will denote the time to complete an update cycle as $t_{\text{cycle}} = t_{\text{blk}} + t_{\text{skip}}$. We will use the first $t_{\text{blk}}$ timesteps out of every $t_{\text{cycle}}$ to update our estimate. For reasons which will become apparent later, we will break up the sample blocks into *previous* ones which have been completed and the *current* one in progress (if there is one in progress). Lemma 4.1 formally defines the vectors of previous and current samples used in the update at timestep $k$.

**Lemma 4.1.** *Assume that $t_{skip} \geq \lfloor r_t \rfloor$. Let $k \in \{1, \ldots, k_{max}\}$ and let $B_p^{(k)} = 1 + \left\lfloor \frac{k - t_{blk}}{t_{cycle}} \right\rfloor$ denote the number of previously completed sample blocks. If $B_p^{(k)} = 0$, let $\underline{y}_p^{(k)} = \underline{x}_p^{(k)} = \emptyset$. Otherwise, let the previous blocks be defined by*

$$\underline{y}_p^{(k)} = \left(\underline{y}_1^T, \ldots, \underline{y}_{B_p^{(k)}}^T\right)^T \text{ where } \underline{y}_i = \underline{y}^{((i-1)t_{cycle}+1:i\,t_{cycle}-t_{skip})},$$
$$\underline{x}_p^{(k)} = \left(\underline{x}_1^T, \ldots, \underline{x}_{B_p^{(k)}}^T\right)^T \text{ where } \underline{x}_i = \underline{x}^{((i-1)t_{cycle}+1:i\,t_{cycle}-t_{skip})}.$$

*Let $n_c^{(k)} = \max\{0, k - B_p^{(k)} t_{cycle}\}$ denote the number of samples taken so far in the current block. If $n_c^{(k)} = 0$, let $\underline{y}_c^{(k)} = \underline{x}_c^{(k)} = \emptyset$, otherwise let the current block be defined by,*

$$\underline{y}_c^{(k)} = \underline{y}^{(B_p^{(k)} t_{cycle}+1:k)}, \qquad \underline{x}_c^{(k)} = \underline{x}^{(B_p^{(k)} t_{cycle}+1:k)}.$$

*Then the previous sample blocks $\underline{y}_i$ are uncorrelated to each other, and uncorrelated to the current block $\underline{y}_c^{(k)}$.*

**Proof.** By construction, there are $t_{\text{skip}}$ timesteps between each of the successive previous blocks and between the last of them and the current block. This means that the time lag between blocks is $t_{\text{skip}}+1$, and by the assumption that $t_{\text{skip}} \geq \lfloor r_{\text{t}} \rfloor$, the blocks are uncorrelated. ■

Table II shows an example of the vectors used to calculate the estimated sigma mean at a given timestep.

| Value of $k$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Value of $\underline{y}_{\text{c}}^{(k)}$ | $\underline{y}^{(1)}$ | $\emptyset$ | $\emptyset$ | $\underline{y}^{(4)}$ | $\emptyset$ | $\emptyset$ |
| Value of $\underline{y}_{\text{p}}^{(k)}$ | $\emptyset$ | $\underline{y}_1$ | $\underline{y}_1$ | $\underline{y}_1$ | $(\underline{y}_1^T, \underline{y}_2^T)^T$ | $(\underline{y}_1^T, \underline{y}_2^T)^T$ |
| Block Number | 1 | 1 | skip | 2 | 2 | skip |

Table II. Example of update schedule for estimating sigma mean with $r_{\text{t}} = 1.5$, $t_{\text{skip}} = 1$, and $t_{\text{blk}} = 2$ (thus $t_{\text{cycle}} = 3$). Here $\underline{y}_{\text{c}}$ denotes the vector of samples in the *current* block, and $\underline{y}_{\text{p}}$ denotes the vector of samples in *previous* blocks, both give the values *after* the $k$th samples have been incorporated. The completed block vectors are given by $\underline{y}_1 = \underline{y}^{(1:2)}$ and $\underline{y}_2 = \underline{y}^{(4:5)}$.

Let $\mathbf{K}_i$, $\mathbf{F}_i$, and $\mathbf{E}_i$, respectively $\mathbf{K}_{\text{c}}^{(k)}$, $\mathbf{F}_{\text{c}}^{(k)}$, and $\mathbf{E}_{\text{c}}^{(k)}$ denote the values of the matrices $\mathbf{K}$, $\mathbf{F}$, and $\mathbf{E}$ as calculated from the locations $\underline{x}_i$, respectively $\underline{x}_{\text{c}}^{(k)}$. For $k$ such that $n_{\text{c}}^{(k)} = 0$, let $\mathbf{K}_{\text{c}}^{(k)} = \mathbf{F}_{\text{c}}^{(k)} = \emptyset$, and let $\mathbf{E}_{\text{c}}^{(k)} = \mathbf{0}_{p\times p}$. The following lemma illustrates how a running estimate of the sigma mean may be calculated from $\underline{y}_{\text{c}}^{(k)}$ and $\underline{x}_{\text{c}}^{(k)}$.

**Lemma 4.2 (Sequential formulation of the sigma mean)** *Let $\varphi^{(k)}$ denote the posterior predictive mean of $\sigma^2$ under the sample vector $\underline{y}_{\varphi}^{(k)} = \big((\underline{y}_p^{(k)})^T, (\underline{y}_c^{(k)})^T\big)^T$. Under the assumptions of Lemma 4.1, we may write,*

$$\varphi^{(k)} = \frac{1}{\nu + nt_{blk}B_p^{(k)} + n_c^{(k)} - 2}\Big[q\nu + \beta_0^T\mathbf{K}_0^{-1}\beta_0 + \sum_{i=1}^{B_p^{(k)}} \Upsilon_i + \Upsilon_c^{(k)} -$$
$$- \big(\mathbf{K}_0^{-1}\beta_0 + \sum_{i=1}^{B_p^{(k)}} \Gamma_i + \Gamma_c^{(k)}\big)^T\big(\mathbf{K}_0^{-1} + \sum_{i=1}^{B_p^{(k)}} \mathbf{E}_i + \mathbf{E}_c^{(k)}\big)^{-1}\big(\mathbf{K}_0^{-1}\beta_0 + \sum_{i=1}^{B_p^{(k)}} \Gamma_i + \Gamma_c^{(k)}\big)\Big], \textit{ where}$$
$$\Upsilon_i = \underline{y}_i^T\mathbf{K}_i^{-1}\underline{y}_i \qquad \Upsilon_c^{(k)} = (\underline{y}_c^{(k)})^T(\mathbf{K}_c^{(k)})^{-1}\underline{y}_c^{(k)}$$
$$\Gamma_i = \mathbf{F}_i\mathbf{K}_i^{-1}\underline{y}_i \qquad \Gamma_c^{(k)} = \mathbf{F}_c^{(k)}\big(\mathbf{K}_c^{(k)}\big)^{-1}\underline{y}_c^{(k)}.$$

*Here, $\Upsilon_c^{(k)}$, and $\Gamma_c^{(k)}$ are taken to be $0$ if $n_c^{(k)} = 0$.*

**Proof.** Under the assumption that $t_{\text{skip}} \geq \lfloor r_{\text{t}} \rfloor$, the sample blocks $\underline{y}_i$ are uncorrelated to each other, and uncorrelated to $\underline{y}_{\text{c}}^{(k)}$. Using Lemma I.4 in Appendix I, the posterior mean of $\sigma^2$ from data $\underline{y}_{\varphi}^{(k)}$ may be restated as in Equation (11). Successive applications of Equation (10) from Lemma I.3 (also in Appendix I) yield the result. ■

Lemma 4.2 demonstrates how an estimate of the sigma mean at step $k$ can be built from previous calculations. If the values of $\Upsilon_i$, $\Gamma_i$, and $\mathbf{E}_i$ have been calculated for all previous update blocks and their sums stored, the network need only calculate $\Upsilon_{\text{c}}^{(k)}$, $\Gamma_{\text{c}}^{(k)}$, and $\mathbf{E}_{\text{c}}^{(k)}$ from recent information and assemble the parts of $\varphi^{(k)}$.

*4.1.2. Approximating unrealized sample values.* While seeking optimal locations to take the $(k+1)$st set of measurements, we would like to incorporate the effect of their *locations* on the posterior variance, but the actual values have not yet been sampled. Our approach is to use a generalized least squares approximation of $\underline{y}^{(k+1)}$ given only the samples used in $\varphi^{(k)}$. We describe this in detail in the following result.

**Proposition 4.3 (Generalized least squares estimate of sigma mean)** *Under the assumptions of Lemma 4.2, let* $\hat{\underline{y}}_{LS}^{(k)} : \Omega^{(k)} \to \mathbb{R}^n$ *map a vector of locations,* $P \in \Omega^{(k)}$ *to the generalized least squares estimate, based on the sample vector* $\underline{y}_{\varphi}^{(k)}$*, of a vector of samples to be taken at locations* $(P, k+1)$*. Then we may write,*

$$\hat{\underline{y}}_{LS}^{(k)}(P) = \begin{cases} \mathbf{K}_{nc}^{(k)}(P)(\mathbf{K}_c^{(k)})^{-1}\underline{y}_c^{(k)} & \textit{if } n_c^{(k)} > 0 \\ \mathbf{0} & \textit{otherwise}, \end{cases} \tag{8}$$

*where* $\mathbf{K}_{nc}^{(k)}(P)$ *is the correlation matrix between the new set of locations and* $\underline{x}_c^{(k)}$*. Now, let* $\hat{\varphi}^{(k)} : \Omega^{(k)} \to \mathbb{R}$ *be defined as,*

$$\hat{\varphi}^{(k)}(P) = \varphi^{(k)} \qquad \textit{if} \quad (n_c^{(k+1)} = 0), \quad \textit{otherwise,}$$

$$\hat{\varphi}^{(k)}(P) = \frac{1}{\nu + nt_{blk}B_p^{(k+1)} + n_c^{(k+1)} - 2}\Big[q\nu + \beta_0^T\mathbf{K}_0^{-1}\beta_0 + \sum_{i=1}^{B_p^{(k)}}\Upsilon_i + \Upsilon_c^{(k)} -$$
$$- \big(\mathbf{K}_0^{-1}\beta_0 + \sum_{i=1}^{B_p^{(k)}}\Gamma_i + \Gamma_c^{(k)}\big)^T\big(\mathbf{K}_0^{-1} + \sum_{i=1}^{B_p^{(k)}}\mathbf{E}_i + \mathbf{E}_c^{(k+1)}(P)\big)^{-1}\big(\mathbf{K}_0^{-1}\beta_0 + \sum_{i=1}^{B_p^{(k)}}\Gamma_i + \Gamma_c^{(k)}\big)\Big],$$

*where* $\mathbf{E}_c^{(k+1)}(P)$ *denotes the matrix* $\mathbf{E}$ *as calculated with space-time location vector* $\underline{x}_c^{(k+1)}(P) = \big((\underline{x}_c^{(k)})^T, (P, k+1)^T\big)^T$*. After the new samples,* $\underline{y}^{(k+1)}$ *have been taken at locations* $(P, k+1)$*, let*

*$\overline{y}_{LS}^{(k)} : \Omega^{(k)} \to \mathbb{R}^n$ denote the estimation error, i.e., $\overline{y}_{LS}^{(k)}(P) = \underline{y}^{(k+1)} - \hat{\underline{y}}_{LS}^{(k)}(P)$. Then we may write,*

$$\varphi^{(k+1)} = \hat{\varphi}^{(k)}(P) + G(\underline{y}_{\varphi}^{(k+1)}, \underline{x}_{\varphi}^{(k+1)})\overline{y}_{LS}^{(k)}(P),$$

*for some function $G : \mathbb{R}^{nt_{\text{blk}}B_c^{(k+1)}} \times \mathcal{D}_e^{nt_{\text{blk}}B_c^{(k+1)}} \to \mathbb{R}$. In other words, $\varphi^{(k+1)}$ may be estimated by $\hat{\varphi}^{(k)}(P)$, and the estimation is exact if $\overline{y}_{LS} = \mathbf{0}$.*

**Proof.** Under the assumption that $\underline{y}_{\text{p}}^{(k)}$ is not correlated to $\underline{y}_{\text{c}}^{(k)}$, Equation (8) can be derived using Lemma I.2 in Appendix I. The estimation result takes two forms. If $n_{\text{c}}^{(k+1)} = 0$, then $\hat{\varphi}^{(k)}(P) = \varphi^{(k)}$. By Lemma 4.2, we also have $\varphi^{(k+1)} = \varphi^{(k)}$, so $G(\cdot) = \mathbf{0}$ satisfies the result. For $n_{\text{c}}^{(k+1)} > 0$, the result can be obtained by applying Equation (9) in Lemma I.3 to the quantities $\Upsilon_{\text{c}}^{(k+1)}$, $\Gamma_{\text{c}}^{(k+1)}$, and $\mathbf{E}_{\text{c}}^{(k+1)}(P)$. ■

In the next section, we examine the other part of $\mathcal{A}^{(k)}$, the conditional variance, and develop an efficient approximation which may be calculated locally by each node.

### *4.2. Upper bound on conditional variance*

Next, we seek an efficient approximation of the conditional variance term $\phi\big((s_0, t_0); \underline{x}^{(1:k+1)}(P)\big)$ in (7). As noted in Remark 2.2, $\phi$ represents the direct effect of the sample locations on the predictive uncertainty (i.e., conditional on $\sigma^2$). The following proposition gives an approximation of the integrated conditional variance which may be calculated by the network of nodes.

**Proposition 4.4 (Approximate integrated conditional variance)** *Let $\underline{x}_j^{(k+1-\lfloor r_t \rfloor:k+1)}(P)$ denote an ordering of the set of past or current locations correlated in space to $V_j(Q)$ and in time to $k+1$, such that,*

$$i_{\mathbb{F}}\left(\underline{x}_j^{(k+1-\lfloor r_t \rfloor:k+1)}(P)\right) = \left\{(s,t) \in i_{\mathbb{F}}\left(\underline{x}^{(1:k+1)}(P)\right) \mid \mathrm{d}(s, V_j(Q)) < r_s \text{ and } k+1-t < r_t\right\}.$$

*Let $\phi_j^{(k)} : \mathcal{D}_e \times \Omega^{(k)} \to \mathbb{R}$ map a prediction location, $x_0 \in \mathcal{D}_e$ and a vector of sample locations, $P \in \Omega^{(k)}$, to the conditional variance of a prediction made at $x_0$ using only the samples at space-time locations $\underline{x}_j^{(k+1-\lfloor r_t \rfloor:k+1)}(P)$. Then the following holds,*

$$\int_{\mathcal{D}} \int_{T} \phi((s_0, t_0); \underline{x}^{(1:k+1)}(P))\, dt_0\, ds_0 \leq \sum_{j=1}^{m} \int_{V_j(Q)} \int_{T} \phi_j^{(k)}((s_0, t_0); P)\, dt_0\, ds_0.$$

**Proof.** Note that although $\underline{x}_j^{(k+1-\lfloor r_t \rfloor:k+1)}(P)$ is not unique, the invariance of the conditional variance to permutations of the sample locations ensures uniqueness of $\phi_j\big(x_0; P\big)$. The result follows from Proposition I.5 in Appendix I. ■

## *4.3. The aggregate average prediction variance and its smoothness properties*

Building on the results from Sections 4.1 and 4.2, we define here the *aggregate average prediction variance* $\tilde{\mathcal{A}}^{(k)}$. Unlike $\mathcal{A}^{(k)}$, the function $\tilde{\mathcal{A}}^{(k)}$ may be computed efficiently in a distributed manner over $\mathcal{N}$. The proof of the following result is a direct consequence of the results above.

**Proposition 4.5. (Spatiotemporal approximation for distributed implementation)** *Let* $\tilde{\mathcal{A}}_j^{(k)} : \Omega^{(k)} \to \mathbb{R}$ *be defined by*

$$\tilde{\mathcal{A}}_j^{(k)}(P) = \hat{\varphi}^{(k)}(P) \int_{V_j(Q)} \int_T \phi_j^{(k)}\left((s,t), P\right)\, dt\, ds.$$

*Then* $\tilde{\mathcal{A}}^{(k)}(P) = \sum_{j=1}^{m} \tilde{\mathcal{A}}_j^{(k)}(P)$ *satisfies the inequality,* $\lim_{k\to\infty} \tilde{\mathcal{A}}^{(k)}(P) \geq \lim_{k\to\infty} \mathcal{A}^{(k)}(P)$.

**Remark 4.6.** Note that the comparison between $\tilde{\mathcal{A}}^{(k)}(P)$ and $\mathcal{A}^{(k)}(P)$ is somewhat stronger than the limiting result shown in Proposition 4.5. The quantity $\tilde{\mathcal{A}}^{(k)}(P)$ is comprised of the product of two terms. The approximate sigma mean is equal to the sigma mean in the limit, while the approximate conditional variance is an upper bound to the actual conditional variance for all $k$. •

Next, we characterize the smoothness properties of $\tilde{\mathcal{A}}^{(k)}$. Let $\nabla_{il}$ denote the partial derivative with respect to $p_{il}$, the $l$th spatial component of the location of $R_i$. We denote by $\nabla_i$ the partial derivative with respect to $p_i$, i.e., $\nabla_i = (\nabla_{i1}, \dots, \nabla_{id})^T$. Thus the gradient of $\tilde{\mathcal{A}}^{(k)}$ at location $P$ may be represented as the $n * d$-dimensional vector $\left(\nabla_1^T \tilde{\mathcal{A}}^{(k)}(P), \dots, \nabla_n^T \tilde{\mathcal{A}}^{(k)}(P)\right)^T$. Given matrix, $A$, we denote by $\nabla_{il} A$ the component-wise partial derivative of $A$. The proofs of the following results amount to a careful bookkeeping of the smoothness properties of the various ingredients involved in the expressions.

**Lemma 4.7 (Gradient of conditional variance)** *Assume that* $f_1, \dots, f_p$ *and the covariance of* $Z$ *are* $C^1$ *with respect to the spatial position of their arguments. Then the map* $P \mapsto \phi_j^{(k)}(x_0, P)$ *is* $C^1$ *on* $\Omega^{(k)}$ *and admits the partial derivative,*

$$\begin{aligned}\nabla_{il}\phi_j^{(k)}(x_0,P) = {} & -2\mathbf{k}^T\mathbf{K}^{-1}\nabla_{il}\mathbf{k} + \mathbf{k}^T\mathbf{K}^{-1}\nabla_{il}\mathbf{K}\mathbf{K}^{-1}\mathbf{k} - \\ & - \xi_0^T\left(\mathbf{K}_0^{-1}+\mathbf{E}\right)^{-1}\nabla_{il}\mathbf{E}\left(\mathbf{K}_0^{-1}+\mathbf{E}\right)^{-1}\xi_0 + 2\xi_0^T\left(\mathbf{K}_0^{-1}+\mathbf{E}\right)^{-1}\nabla_{il}\xi_0, \text{ with}\end{aligned}$$

$$\nabla_{il}\xi_0 = -\nabla_{il}\mathbf{F}\mathbf{K}^{-1}\mathbf{k} - \mathbf{F}\mathbf{K}^{-1}\nabla_{il}\mathbf{k} + \mathbf{F}\mathbf{K}^{-1}\nabla_{il}\mathbf{K}\mathbf{K}^{-1}\mathbf{k},$$

$$\nabla_{il}\mathbf{E} = \nabla_{il}\mathbf{F}\mathbf{K}^{-1}\mathbf{F}^T + \mathbf{F}\mathbf{K}^{-1}\nabla_{il}\mathbf{F}^T - \mathbf{F}\mathbf{K}^{-1}\nabla_{il}\mathbf{K}\mathbf{K}^{-1}\mathbf{F},$$

*where the matrices $\mathbf{K}$, $\mathbf{E}$, and $\mathbf{F}$ and the vectors $\mathbf{k}$, and $\xi_0$ are calculated from the location subvector, $\underline{x}_j^{(k+1-\lfloor r_t \rfloor:k+1)}(P)$.*

It is worth noting that the matrix $\nabla_{il}\mathbf{F}$ is nonzero only in column $i$. The matrix $\nabla_{il}\mathbf{K}$ is nonzero only in row and column $i$. Additionally, due to the finite correlation range in space and time, only those elements corresponding to correlation with other measurement locations $x = (s,t)$ which satisfy $\|p_i - s\| \leq r_{\mathrm{s}}$ and $t > k+1-r_{\mathrm{t}}$ are nonzero.

**Lemma 4.8 (Gradient of conditional variance is Lipschitz)** *Under the assumptions of Lemma 4.7, assume, in addition, that the partial derivatives of $f_1, \dots, f_p$ and the covariance of $Z$ are $C^1$ with respect to the spatial position of their arguments. Then the map $P \mapsto \nabla_i \phi_j^{(k)}(x_0, P)$ is globally Lipschitz on $\Omega^{(k)}$.*

Note that the value of $\hat{\varphi}^{(k)}(P)$ depends on $P$ only through the matrix $\mathbf{E}_{\mathrm{c}}^{(k+1)}$, whose partial derivative is analogous to that of $\mathbf{E}$ in Lemma 4.7. This leads us to the following continuity results.

**Lemma 4.9 (Gradient of sigma mean)** *Under the assumptions of Lemma 4.7, $\hat{\varphi}^{(k)}$ is $C^1$ on $\Omega^{(k)}$ and admits the partial derivative,*

$$\nabla_{il}\hat{\varphi}^{(k)}(P) = \begin{cases} \mathbf{0} & \textit{if} \quad n_c^{(k+1)} = 0 \\ -\frac{\Psi(P)^T \, \nabla_{il}\mathbf{E}_c^{(k+1)}(P)\, \Psi(P)}{\nu + nt_{\mathit{blk}} B_p^{(k+1)} + n_c^{(k+1)} - 2} & \textit{otherwise, where,} \end{cases}$$

$$\Psi(P) = \big(\mathbf{K}_0^{-1} + \sum_{i=1}^{B_p^{(k)}} \mathbf{E}_i + \mathbf{E}_c^{(k+1)}(P)\big)^{-1}\big(\mathbf{K}_0^{-1}\beta_0 + \sum_{i=1}^{B_p^{(k)}} \Gamma_i + \Gamma_c^{(k)}\big).$$

*Additionally, under the assumptions of Lemma 4.8, the gradient $\nabla\hat{\varphi}^{(k)}$ is globally Lipschitz on $\Omega^{(k)}$.*

**Proof.** This result stems from the fact that $\hat{\varphi}^{(k)}(P)$ depends on $P$ only through $\mathbf{E}_{\mathrm{c}}^{(k+1)}(P)$. ■

We are now ready to state the smoothness properties of $\tilde{\mathcal{A}}^{(k)}$ and provide an explicit expression for its gradient. This is a direct result of the Lemmas above.

**Proposition 4.10 (Gradient of sigma mean is Lipschitz)** *Under the assumptions of Lemma 4.7, $\tilde{\mathcal{A}}^{(k)}$ is $C^1$ on $\Omega^{(k)}$ and admits the partial derivative,*

$$\nabla_i\tilde{\mathcal{A}}^{(k)}(P) = \hat{\varphi}^{(k)}(P)\int_{V_j(Q)}\int_T \nabla_i\phi_j^{(k)}\left((s,t),P\right)\, dt\, ds \\ + \nabla_i\hat{\varphi}^{(k)}(P)\int_{V_j(Q)}\int_T \phi_j^{(k)}\left((s,t),P\right)\, dt\, ds.$$

*Additionally, under the assumptions of Lemma 4.8, the gradient $\nabla\tilde{\mathcal{A}}^{(k)}$ is globally Lipschitz on $\Omega^{(k)}$.*

## *4.4. Distributed computation of aggregate average prediction variance and its gradient*

Here, we substantiate our assertion that the aggregate average prediction variance and its gradient introduced in Section 4.3 are distributed over the network $\mathcal{N}$. Since $\mathcal{V}(Q)$ is a partition of the physical space, we may partition all sample locations by region. Thus for each $(s,t) \in i_{\mathbb{F}}(\underline{x})$, there is exactly one $j \in \{1,\dots,m\}$ such that $s \in V_j(Q)$. In order for the network to calculate $\tilde{\mathcal{A}}^{(k)}$ and its gradient at $P$, it is sufficient for $S_j$ to compute $\tilde{\mathcal{A}}_j^{(k)}$ and $\nabla_i \tilde{\mathcal{A}}_j^{(k)}$ for each robot in $V_j(Q)$. Then $\tilde{\mathcal{A}}^{(k)}$ may be calculated via discrete-time average consensus (cf. Section 2.3), while $\nabla_i \tilde{\mathcal{A}}^{(k)}$ may be calculated from information local to $R_i$. From Propositions 4.5 and 4.10, it can be seen that calculation of $\tilde{\mathcal{A}}_j^{(k)}$ and $\nabla_i \tilde{\mathcal{A}}_j^{(k)}$ requires only local information in addition to the (global) values of $\hat{\varphi}^{(k)}(P)$ and $\nabla_i \hat{\varphi}^{(k)}(P)$. Let us explain how these two quantities can be calculated.

In this section we are concerned with elements of the vectors and matrices associated with the *current* update block of $\hat{\varphi}^{(k)}(P)$. Let $x_{\mathrm{c}:i}^{(k)}$, respectively $y_{\mathrm{c}:i}^{(k)}$ denote the $i$th element of the vector $\underline{x}_{\mathrm{c}}^{(k)}$, respectively $\underline{y}_{\mathrm{c}}^{(k)}$ for $i \in \{1,\dots,n\}_{\mathrm{c}}^{(k)}$. Let $\mathrm{I}_{\mathrm{Local}}^{(k)} : \mathbb{N} \to \mathbb{F}(\mathbb{N})$ map the index of the node to the set of indices of samples in the current update block whose spatial position lies inside its Voronoi cell, and whose time element is correlated to time $k+1$,

$$\mathrm{I}_{\mathrm{Local}}^{(k)}(j) = \begin{cases} \emptyset & \text{if } n_{\mathrm{c}}^{(k)} = 0, \\ \Big\{ i \in \{1,\dots,n_{\mathrm{c}}^{(k)}\} \mid x_{\mathrm{c}:i}^{(k)} = (s,t) \text{ and } s \in V_j(Q) \Big\} & \text{otherwise.} \end{cases}$$

With a slight abuse of notation, define $\mathrm{I}_{\mathrm{Local}}^{(k+1)}(j,P)$ to be the equivalent set of indices into the full vector of measurement locations, $\underline{x}_{\mathrm{c}}^{(k+1)}(P)$, with the caveat that $\mathrm{I}_{\mathrm{Local}}^{(k+1)}(j,P) = \emptyset$ if $n_{\mathrm{c}}^{(k+1)} = 0$.

Our first result illustrates the parts of $\hat{\varphi}^{(k)}(P)$ which do not include the locations $P$.

**Proposition 4.11 (Distributed calculations without $P$)** *Assume that $S_j$, for $j \in \{1,\dots,m\}$, knows $x_{c:i}^{(k)}, y_{c:i}^{(k)}$ for each $i \in I_{Local}^{(k)}(j)$. After $p+1$ executions of the JOR algorithm and $2$ subsequent consensus algorithms, $S_j$ has access to,*

`#1:` *element $i$ of $(\mathbf{K}_c^{(k)})^{-1}\underline{y}_c^{(k)} \in \mathbb{R},\ i \in I_{Local}^{(k)}(j)$ via JOR;*

`#2:` *$col_i\left(\mathbf{F}_c^{(k)}(\mathbf{K}_c^{(k)})^{-1}\right) \in \mathbb{R}^p,\ i \in I_{Local}^{(k)}(j)$ via JOR;*

`#3:` *$\Gamma_c^{(k)} \in \mathbb{R}^p$ via consensus;*

`#4:` *$\Upsilon \in \mathbb{R}^p$ via consensus.*

**Proof.** Under the assumptions on $\mathcal{N}$, the matrix $\mathbf{K}_{\mathrm{c}}^{(k)}$, satisfies the requirements of the distributed JOR algorithm. The results here build on this fact and the connectedness of the network of nodes (which allows for consensus). ■

Next, we describe calculations that the network can execute when robotic agents are at locations $P$.

**Proposition 4.12 (Distributed calculations with $P$)** *Given $P \in \Omega^{(k)}$, assume that $S_j$, for $j \in \{1,\dots,m\}$, knows $x_{c:i}^{(k)}$ for each $i \in I_{Local}^{(k+1)}(j,P)$ and the results of Proposition 4.11. Let $\mathbf{F}_c^{(k+1)}(P)$ denote the matrix of basis functions evaluated at locations $\underline{x}_c^{(k+1)}$. After $p$ executions of JOR and $p^2$ executions of consensus algorithms, $S_j$ has access to,*

#5: *col*$_i\left(\mathbf{F}_c^{(k+1)}(P)(\mathbf{K}_c^{(k+1)}(P))^{-1}\right) \in \mathbb{R}^p,\ i \in I_{Local}^{(k+1)}(j,P)$ *via JOR;*

#6: $\mathbf{E}_c^{(k+1)}(P) \in \mathbb{R}^{p\times p}$ *via consensus.*

*After these computations, $S_j$ can calculate $\nabla_{il}\mathbf{E}_c^{(k+1)}$ for $l \in \{1,\dots,d\}$. Under the assumption that $S_j$ knows the quantities $\sum_{i=1}^{B_c^{(k)}} \mathbf{E}_i$, $\sum_{i=1}^{B_c^{(k)}} \Upsilon_i$, and $\sum_{i=1}^{B_c^{(k)}} \Gamma_i$, then $S_j$ can calculate $\hat{\varphi}^{(k)}(P)$ and $\nabla_i\hat{\varphi}^{(k)}(P)$ for each robot in $\{i \in \{1,\dots,n\} \mid p_i \in V_j(Q)\}$.*

**Proof.** The matrix $\mathbf{K}_{\mathrm{c}}^{(k+1)}(P)$ satisfies the requirements of the distributed JOR algorithm by the assumptions on $\mathcal{N}$. The itemized results follow. The calculation of $\hat{\varphi}^{(k)}(P)$ and its partial derivatives follow from Lemmas 4.2 and 4.9. ■

## 5. Distributed optimization of the aggregate average predictive variance

Here we present a distributed projected gradient descent algorithm which is guaranteed to converge to a stationary point of $\tilde{\mathcal{A}}^{(k)}$ on $\Omega^{(k)}$. Table III outlines the DISTRIBUTED LINE SEARCH ALGORITHM, which is a distributed version of the LINE SEARCH ALGORITHM from Table I. The maximum stepsize, $\alpha_{\max} \in \mathbb{R}_{>0}$, is designed to ensure that all robots with nonzero partial derivatives can move the maximum distance.

The DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM in Table IV allows the network of static nodes and mobile agents to find local minima of $\tilde{\mathcal{A}}^{(k)}$ on $\Omega^{(k)}$. At timestep $k$, the nodes follow a gradient descent algorithm, defining a sequence of configurations, $\{P_l^\dagger\}$, $l \in \mathbb{N}$, such that $P_1^\dagger$ is $P^{(k)} \in \Omega^{(k)}$, the vector of current spatial locations of the robotic agents and

$$P_{l+1}^\dagger = \mathrm{proj}_\Omega\left(P_l^\dagger - \alpha\nabla\tilde{\mathcal{A}}|_{P_l^\dagger}\right),\ \alpha \in \mathbb{R}_{\geq 0},$$

| | |
|---|---|
| **Name:** | DISTRIBUTED LINE SEARCH ALGORITHM |
| **Goal:** | Compute step size for projected gradient descent of $\tilde{\mathcal{A}}^{(k)}$ |
| **Input:** | Configuration, $P = (p_1, \dots, p_n) \in \Omega^{(k)}$ |
| **Assumes:** | (i) Connected network of static nodes |
| | (ii) $S_j$ knows $p_i$, $\tilde{\mathcal{A}}_j^{(k)}(P)$, $\nabla_i \tilde{\mathcal{A}}^{(k)}(P)$ and $\Omega_i$ for each robot within communication range |
| | (iii) $\Vert\nabla_i \tilde{\mathcal{A}}^{(k)}(P)\Vert \neq 0$ for at least one $i \in \{1, \dots, n\}$ |
| | (iv) $S_j$ knows items #3 and #4 from Proposition 4.11 |
| | (v) Shrinkage factor $\tau$ and tolerance $\theta \in (0,1)$ known a priori by all static nodes |
| **Uses:** | (i) Projection of next set of locations on $\Omega_i$, $P'_j(\alpha, P) = \Big\{\mathrm{proj}_{\Omega_i}(p_i + \alpha \nabla_i \tilde{\mathcal{A}}(P)), \text{for each } i \text{ such that } \mathrm{d}(p_i, V_j(Q)) \leq r_{\mathrm{s}} + u_{\max} + \omega\Big\}$. |
| | (ii) Total distance traveled by robots entering $V_j(Q)$, $\mathrm{d}_j(\alpha, P) = \sum_{\substack{i \in \{1,\dots,n\} \text{ such that} \\ \mathrm{proj}_{\Omega_i}\left(p_i + \alpha \nabla_i \tilde{\mathcal{A}}(P)\right) \in V_j(Q)}} \Vert \mathrm{proj}_{\Omega_i}\Big(p_i + \alpha \nabla_i \tilde{\mathcal{A}}(P)\Big) - p_i \Vert^2$. |
| **Output:** | Step size $\tau \in \mathbb{R}$ |

Initialization

```
1: $S_1, \dots, S_m$ calculate $\alpha_{\max} = \dfrac{u_{\max}}{\min\{\Vert\nabla_i \tilde{\mathcal{A}}(P)\Vert \mid \Vert\nabla_i \tilde{\mathcal{A}}(P)\Vert \neq 0\}}$ via a consensus algorithm
```

For $j \in \{1, \dots, m\}$, node $S_j$ executes concurrently

```
1: $\alpha = \alpha_{\max}$
2: repeat
3:   calculates $\mathrm{d}_j(\alpha, P)^2$
4:   calculates $\hat{\varphi}^{(k)}\big(P'_j(\alpha, P)\big)$ according to Proposition 4.12
5:   calculates $\tilde{\mathcal{A}}_j^{(k)}\big(P'_j(\alpha, P)\big)$
6:   execute consensus algorithm to calculate the following:
        $$\tilde{\mathcal{A}}^{(k)}\big(P'(\alpha, P)\big) = \sum_{j=1}^{m} \tilde{\mathcal{A}}_j^{(k)}\big(P'_j(\alpha, P)\big)$$
        $$\big\Vert P - P'(\alpha, P)\big\Vert^2 = \sum_{j=1}^{m} \mathrm{d}_j(\alpha, P)^2$$
7:   $\nu = \frac{\theta}{\alpha}\Vert P - P'(\alpha, P)\Vert^2 + \tilde{\mathcal{A}}^{(k)}(P'(\alpha, P)) - \tilde{\mathcal{A}}^{(k)}(P)$
8:   if $\nu > 0$ then
9:     $\alpha = \alpha\tau$
10: until $\nu \leq 0$
```

Table III. DISTRIBUTED LINE SEARCH ALGORITHM.

where $\alpha$ is chosen via DISTRIBUTED LINE SEARCH ALGORITHM. When $|\tilde{\mathcal{A}}^{(k)}(P^{\dagger}_{l+1}) - \tilde{\mathcal{A}}^{(k)}(P^{\dagger}_{l})| = 0$, the algorithm terminates, and the nodes set $P^{(k+1)} = P^{\dagger}_{l+1}$. By the end of this calculation, each node knows the identity of robotic agents in its Voronoi cell at timestep $k+1$. Node $S_j$ transmits $p_i(k+1)$ to robot $R_i$, which then moves to the location between timesteps.

The following proposition describes some nice properties of the DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM. Its proof is a direct result of the construction of the algorithm and the fact that it is equivalent to a centralized projected gradient descent.

**Proposition 5.1. (Properties of the DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM)** *The* DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM *is distributed over the network $\mathcal{N}$. Moreover, under the assumptions of Lemma 4.8, any execution is such that the robots do not collide and, at each timestep after the first, measurements are taken at stationary configurations of $P \mapsto \tilde{\mathcal{A}}^{(k)}(P)$ over $\Omega^{(k)}$.*

The proposed algorithm is robust to agent failures. If an agent stops sending position updates, it ceases to receive new control vectors. The rest of the network continues operating with the available resources and will eventually sample the areas previously covered by the failing agents.

**Remark 5.2 (Extension to relative positioning)** It is interesting to observe that, due to the fact that the actual positions of samples are only required in a local context, our algorithm can also be implemented in a robotic network with relative positioning. The only requirements are: that each node can calculate the mean basis function for all local samples; that each node can calculate the correlations between pairs of local samples; and that neighboring nodes can agree on the ordering of those samples within the global matrix. These modifications would not impact the convergence properties of the algorithm. •

## *5.1. Complexity analysis*

Here we examine the complexity of the DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM in detail. We are interested in characterizing the complexity of the algorithm implementation in terms of the number of robotic agents and the number of static nodes. For reference, we compare our proposed algorithm against a centralized algorithm that uses all-to-all broadcast and global information, and does not take advantage of the distributed nature of the problem.

Given that the DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM is sequential, and

| | |
|---|---|
| **Name:** | DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM |
| **Goal:** | Find a local minimum of $\tilde{\mathcal{A}}^{(k)}$ within $\Omega^{(k)}$. |
| **Assumes:** | (i) Connected network of static computing nodes and mobile robotic sensing agents<br>(ii) Static nodes deployed over $\mathcal{D}$ such that $R \geq \max_{i\in\{1,\dots,m\}} \{\mathrm{CR}(V_i(Q))\} + r_{\mathrm{s}} + u_{\max}$, robotic agents in initial configuration $P^{(1)} \in \Omega^{(k)}$<br>(iii) Line search shrinkage factor $\tau$ and tolerance value $\theta \in (0,1)$ known a priori by all nodes<br>(iv) A termination marker known to all nodes and robots which may be sent to mark the end of a gradient descent loop. |
| **Uses:** | (i) Each node uses the temporary vectors $P_{\text{cur}}$, respectively $P_{\text{next}}$ to hold the configuration at the current, respectively next step of the gradient projection algorithm. For ease of exposition, we use global notation although $S_j$ only calculates and uses the parts of these vectors which correspond to agents currently within communication range. |

At time $k \in \mathbb{Z}_{\geq 0}$, node $S_j$ executes:

```
1: sets $R_{\text{cov}}(j) = \{R_i \mid \mathrm{d}(p_i(k), V_j(Q)) \leq r_{\mathrm{s}}\}$
2: collects initial samples and locations from $R_i$ for each $i \in R_{\text{cov}}(j)$.
3: computes first $\tilde{\mathcal{A}}_j^{(k)}(P^{(k)})$ and then $\tilde{\mathcal{A}}^{(k)}(P^{(k)})$ via consensus
4: sets $P_{\text{next}} = P^{(k)}$
5: repeat
6:    sets $P_{\text{cur}} = P_{\text{next}}(j)$ and calculates $-\nabla\tilde{\mathcal{A}}_j^{(k)}|_{P_{\text{cur}}}$
7:    transmits vector $\nabla_i\tilde{\mathcal{A}}_j^{(k)}(P_{\text{cur}})$ to all robots in $R_{\text{cov}}(j)$
8:    collects sum $\nabla_i\tilde{\mathcal{A}}^{(k)}(P_{\text{cur}})$ from all robots in $R_{\text{cov}}(j)$
9:    runs DISTRIBUTED LINE SEARCH ALGORITHM at $P_{\text{cur}}$ to get $\alpha$
10:   sets $P_{\text{next}} = P_{\text{cur}} + \alpha\nabla\tilde{\mathcal{A}}^{(k)}|_{P_{\text{cur}}}$
11:   calculates $|\tilde{\mathcal{A}}^{(k)}(P_{\text{next}}) - \tilde{\mathcal{A}}^{(k)}(P_{\text{cur}})|$ from known quantities
12: until $|\tilde{\mathcal{A}}^{(k)}(P_{\text{next}}) - \tilde{\mathcal{A}}^{(k)}(P_{\text{cur}})| = 0$
13: sets $P^{(k+1)} = P_{\text{next}}$ and sends next position to robots in $V_j(Q)$
```

Meanwhile, robot $R_i$ executes:

```
1: takes measurement at $p_i(k)$
2: sets $S_{\text{cov}}(i) = \{S_j \mid \mathrm{d}(p_i(k), V_j(Q)) \leq r_{\mathrm{s}}\}$
3: sends measurement and position to all nodes in $S_{\text{cov}}(i)$
4: repeat
5:    receives $\nabla_i\tilde{\mathcal{A}}_j^{(k)}(P^{(k)})$ from nodes in $S_{\text{cov}}(i)$
6:    calculates sum $\nabla_i\tilde{\mathcal{A}}^{(k)}(P^{(k)})$
7:    sends $\nabla_i\tilde{\mathcal{A}}^{(k)}(P^{(k)})$ to all nodes in $S_{\text{cov}}(i)$
8: until receives termination marker from any node
9: receives next location $p_i(k+1)$
10: moves to $p_i(k+1)$.
```

Table IV. DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM.

designed to run for a fixed number of timesteps, we are concerned here with complexities involved in performing a single step. Below, where we refer to complexity notions over multiple iterations of an algorithm, we are considering the nested algorithms such as JOR, or consensus, which run during a single step of the DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM.

We examine the algorithm performance against the following notions of complexity, see [29, 30, 28]

**Communication complexity:** the total number of bits transmitted over all (directed) communication channels between nodes in the network over the course of the algorithm;

**Time complexity:** the number of iterations of the algorithm times the maximum number of bits sent over any channel during one iteration;

**Space complexity:** the total number of bits for which space may be required *by a single node at any given time*.

We are now ready to characterize the complexities of our algorithms, beginning with the inner iterations.

**Proposition 5.3. (Average consensus complexity)** *Assume that the group of static nodes is regular in the sense that its diameter, diam$_Q$ $\in \mathbb{Z}_{>0}$ satisfies diam$_Q$ $\in O(\sqrt[d]{m})$. Further assume that the maximum number of neighbors of a given node is bounded by a constant deg$_Q$ $\in \mathbb{Z}_{>0}$. Let $b = (b_1, \dots, b_m)^T \in \mathbb{R}^m$ denote a vector distributed across the network of nodes in the sense that $S_j$ knows $b_j$ for each $j \in \{1, \dots, m\}$. The discrete time consensus algorithm may be run on the network of nodes to calculate an arithmetic mean of the form $\frac{b^T b}{n}$ with communication complexity in $O\left(m^2 \sqrt[d]{m}\right)$, time complexity in $O\left(m \sqrt[d]{m}\right)$, and space complexity in $O(1)$.*

**Proof.** Each node sends a single message to each neighbor at each step, so the time complexity is bounded by the number of iterations to completion. The number of iterations may be bounded by the number required to reach a desired tolerance level. The error at iteration $t$ may be written $e_{\text{ave}}(t) = \|w_{\text{ave}}(t) - w\|$, where $w$ is the $m$-vector whose elements are all $b^T b$, and $w_{\text{ave}}(t)$ is the vector of current approximate values. This may be bounded in terms of the initial error as $e_{\text{ave}}(t) \leq \left(1 - \frac{4}{m\text{diam}_Q}\right)^t e_{\text{ave}}(0)$, where we have used [31, Equation (6.10)] to lower bound the algebraic connectivity of the network of nodes. Thus the number of steps required to guarantee a particular error is bounded by, $t^*_{\text{ave}} \in O\left(-\log^{-1}\left(1 - \frac{4}{m\text{diam}_Q}\right)\right)$. Applying the bound on the growth of the network diameter yields $t^*_{\text{ave}} \in O\left(-\log^{-1}\left(1 - \frac{4}{m\sqrt[d]{m}}\right)\right)$. For large $m$, we can replace the logarithm with the series representation. The higher order terms drop off and we are left with $t^*_{\text{ave}} \in O\left(m\sqrt[d]{m}\right)$. At each step of the algorithm, each node stores a single value for each neighbor, and a constant number of other values. Thus the space requirement is bounded simply by the number of neighbors, which is in $O(1)$ by assumption. Communication complexity is bounded by a single message over each channel of the network at each iteration. The total number of such messages from each node is bounded by a constant, which gives us the final result. ■

**Remark 5.4.** In two and three dimensions, the maximum diameter requirement has been shown to be consistent with a hexagonal grid network [32, 33], which is also consistent (in terms of number of neighbors) with the average case for large Voronoi networks [34]. The requirement of bounded degree (maximum number of neighbors) would also be satisfied by a hexagonal grid. •

Since $\tilde{\mathcal{A}}^{(k)}$ uses only measurements correlated in time, the size of the matrices and vectors is limited to a constant multiple of $n$. Recall the definitions from Section 4.1 of $\mathbf{K}_{\text{c}}^{(k)}$, respectively $n_{\text{c}}^{(k)}$, the correlation matrix, respectively number, of samples in the *current* update block.

**Proposition 5.5. (Leader election complexity)** *Under the assumptions of Proposition 5.3, the leader election algorithm may be run on the network of nodes to calculate the quantity* $\max_{i\in\{1,\dots,n_c^{(k)}\}} \sum_{j=1}^{n_c^{(k)}} [\mathbf{K}_c^{(k)}]_{ij}$, *with communication complexity in* $O\left(m\sqrt[d]{m}\right)$, *time complexity in* $O\left(\sqrt[d]{m}\right)$, *and space complexity in* $O(1)$.

**Proof.** First, note that for each $j \in \{1,\dots,m\}$, $S_j$ can calculate the row sums which correspond to samples within $V_j(Q)$ and subsequently their maximum, so calculating the maximum row sum is simply a matter of finding the largest value in the network. At each step of the algorithm, each node sends a single message to each neighbor. The algorithm is complete after a number of iterations equal to the diameter of the network. This proves the time and communication complexities. At each iteration, all nodes store a single value, the current max, which takes care of the space complexity. ■

For the following algorithms, the distribution of samples in the region defines two different regimes for complexity. We will consider both the worst case and the average based on a uniform distribution.

**Proposition 5.6. (JOR complexity)** *Under the assumptions of Proposition 5.5, assume that there is some constant* $\epsilon_\lambda \in (0,1)$, *known a priori, such that* $\lambda_{min}(\mathbf{K}_c^{(k)}) > \epsilon_\lambda$. *Regarding the sparsity of* $\mathbf{K}_c^{(k)}$, *assume that any one sample is correlated to at most $N_{cor} \in \mathbb{N}$ others, and that, for any $j \in \{1, \dots, m\}$, the number of samples in $\mathcal{D} \setminus V_j(Q)$ which are correlated to samples in $V_j(Q)$ is upper bounded by a constant, $N_{msg} \in \mathbb{N}$. Let $b = (b_1, \dots, b_{n_c^{(k)}})^T \in \mathbb{R}^{n_c^{(k)}}$ be distributed on the network of nodes in the sense that if $S_j$ knows $col_i(\mathbf{K}_c^{(k)})$, then $S_j$ knows $b_i$. Then the distributed JOR algorithm may be run to calculate $(\mathbf{K}_c^{(k)})^{-1}b$ with communication complexity in $O\left(m \sqrt[d]{m}\right)$, time complexity in $O\left(\sqrt[d]{m}\right)$, and space complexity in $O(n)$ worst case, $O(\frac{n}{m})$ average case.*

**Proof.** The first step of the JOR algorithm is to calculate the relaxation parameter. For correlation matrices, Appendix II describes a near optimal relaxation parameter in the sense of minimizing the completion time. Using two leader election algorithms, the network can calculate the maximum off-diagonal element, $\beta = \max_{i \neq j \in \{1,\dots,n\}} [\mathbf{K}_\text{c}^{(k)}]_{ij}$ and the maximum off-diagonal row sum $\alpha = \max_{i \in \{1,\dots,n\}} \sum_{j \neq 1}^{n} [\mathbf{K}_\text{c}^{(k)}]_{ij}$. The relaxation parameter is then given by, $h^* = \frac{2}{2+\alpha-\beta}$. The time complexity of the JOR algorithm can be broken down into the maximum number of messages any node sends over any channel times the number of iterations. The number of messages $S_j$ will send is equal to the number of nonzero off-diagonal entries $[\mathbf{K}_\text{c}^{(k)}]_{ii'}$, $i \neq i'$, where $s_i \in V_j(Q)$ and $s_{i'} \in V_{j'}(Q)$, with $j \neq j'$. By assumption, this number is bounded by $N_\text{cor}$. The error at iteration $t$ of the JOR algorithm may be written $e_\text{JOR}(t) = \|w_\text{JOR}(t) - (\mathbf{K}_\text{c}^{(k)})^{-1}b\|$, where $w_\text{JOR}(t)$ is the vector of current approximate values. An upper bound on the error at step $t$ may be obtained by [26] $e_\text{JOR}(t) \leq \left(\text{sprad}\left(\boldsymbol{I} - h\mathbf{K}_\text{c}^{(k)}\right)\right)^t e_\text{JOR}(0)$, where $\boldsymbol{I}$ is the $n_\text{c}^{(k)} \times n_\text{c}^{(k)}$ identity matrix. By Proposition II.4, we have the bounds, $0 \leq \text{sprad}\left(\boldsymbol{I} - h\mathbf{K}\right) < 1 - h^*\epsilon_\lambda$. The assumption of sparsity, and the fact that $\mathbf{K}_\text{c}^{(k)}$ is a correlation matrix give us $0 \leq \alpha < N_\text{msg}$, and $0 \leq \beta < 1$, which results in $1 - h^*\epsilon_\lambda < 1 - \frac{2\epsilon_\lambda}{2+N_\text{msg}}$. Since both $\epsilon_\lambda$ and $N_\text{msg}$ are positive, we have $1 - \frac{2\epsilon_\lambda}{2+N_\text{msg}} < 1$. The minimum number of steps required to reach an error of $e^*$ is then upper bounded by $t^* = (e(0) - e^*)\left(-\log^{-1}\left(1 - \frac{2\epsilon_\lambda}{2+N_\text{msg}}\right)\right)$. Thus the number of iterations required to reach a given error tolerance is in $O(1)$. The time complexity is then dominated by the time complexity of the leader election algorithm outlined in Proposition 5.5. For space complexity, we note that the maximum number of samples in a given cell is bounded by $n_\text{c}^{(k)}$, while the average number is $\frac{n_\text{c}^{(k)}}{m}$. The space complexity is dominated by the requirement to store vectors of length given by the number of samples in the cell, and the same number of rows of $\mathbf{K}_\text{c}^{(k)}$. This yields the given result. For communication complexity, the overall algorithm requires a maximum of one message to be sent per nonzero off-diagonal entry in $\mathbf{K}_\text{c}^{(k)}$, each iteration, plus the number of messages required for the leader election. ■

**Remark 5.7.** The assumptions on the sparsity of $\mathbf{K}_{\mathrm{c}}^{(k)}$ in Proposition 5.6 have the following interpretation: samples do not cluster in space as measured with respect to the distribution of the Voronoi cells and their size relative to the correlation range. •

**Proposition 5.8. (Complexity of the DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM)** *Under the assumptions of Proposition 5.5 and Table IV, the* DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM *may be completed with communication complexity in* $O\left(m^2 \sqrt[d]{m}\right)$*, time complexity in* $O\left(m \sqrt[d]{m}\right)$*, and space complexity in* $O(n^2)$ *worst case,* $O\left(\frac{n^2}{m^2}\right)$ *average case.*

**Proof.** First it should be noted that the number of iterations to completion of both the DISTRIBUTED LINE SEARCH ALGORITHM and the DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM are independent of $n$ and $m$. Thus the complexities of each are given by the requirements of a single step. The space complexity is dominated by the need to store the inverse covariance matrix of known samples required for $\tilde{\mathcal{A}}_j^{(k)}$. The worst case corresponds to all $n_{\mathrm{c}}^{(k)}$ samples correlated to one Voronoi region, and the average to samples distributed uniformly. The time and communication complexities are dominated by the requirement of the consensus algorithm. ■

*5.1.1. Broadcast method for comparison* One way to judge the efficiency of our method would be to compare it against a simple algorithm which floods the network with new information at each sample time. This algorithm would work as follows. At each timestep, all samples and locations are disseminated throughout the network, such that each node obtains the entire vectors $\underline{x}$ and $\underline{y}$. Each node then runs a projected gradient descent given global information. Since all nodes have the same information, they should converge to the same final locations. Since this method is only given for comparison, we will assume that this is the case. Once a node has calculated the next location for all of the agents which will be in that Voronoi cell, the control vectors may be transmitted to those agents. The information dissemination in this algorithm corresponds to an all-to-all broadcast in which each node begins with a distinct message of length $|\mathrm{I}_{\mathrm{Local}}^{(k+1)}(j, P)|$ units.

**Proposition 5.9. (Complexity of the broadcast method)** *Under the assumptions of Proposition 5.8, local minima of* $\tilde{\mathcal{A}}^{(k)}$ *may be found by all to all broadcast of agent positions and subsequent local projected gradient descent with,*

- *communication complexity in* $O\left(m^2 n\right)$ *worst case,* $O\left(mn\right)$ *average case*
- *time complexity in* $O\left(n \sqrt[d]{m}\right)$ *worst case,* $O\left(\frac{n}{m} \sqrt[d]{m}\right)$ *average case*

- *space complexity in $O(n^2)$*

**Proof.** The time and communication complexity of this method are dominated by the requirement of an all to all broadcast. Given a network of $m$ nodes, each of which has a distinct message of $M$ bits, an absolute lower bound on the number of bits that must be transmitted can be found by noting that each node must end up with all $(m-1)M$ bits from the other nodes. Thus the number of message units to be transmitted must be at least $m(m-1)M$. The worst and average cases correspond to $M \propto n$ (for all agents in a single region) and $M \propto \frac{n}{m}$ (for agents spread among regions). The longest path any message must take in a standard all to all broadcast is the diameter of the network. The time complexity is then given by the number of bits in a message times the diameter, and by assumption, $\mathrm{diam}_Q \in O(\sqrt[d]{m})$. The space complexity of the algorithm is dominated by the requirement to store the entire inverse covariance matrix at each node. Even though the covariance matrix is sparse, the inverse is in general not, requiring the whole $n_{\mathrm{c}}^{(k)} \times n_{\mathrm{c}}^{(k)}$ storage space. ■

**Remark 5.10 (Broadcast method requires global positioning)** *It should be noted here that while the* DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM *might be extended to systems with relative positioning (see Remark 5.2), the broadcast method requires global coordinates.*

Table V lists the complexities side by side for comparison. It can be seen that the distributed method scales better overall with the number of mobile agents. The results with respect to increasing the number of static nodes are less favorable, but it should be noted that the results presented here for the broadcast method are known lower bounds on the communication complexity, not representative of any achievable algorithm. In particular, we did not account for redundant transmissions, multi-hop path requirements, or stalled communication channels. Any implementable all-to-all broadcast algorithm must balance communication complexity against time complexity to account for these issues.

## 5.2. Simulations

We show here an implementation of the DISTRIBUTED PROJECTED GRADIENT DESCENT ALGORITHM with the following parameters: $m = 5$ static nodes, $n = 20$ robotic agents, and the convex polygon $\mathcal{D}$ with vertices $\{(0,.1)$, $(2.5,.1)$, $(3.45,1.6)$, $(3.5,1.7)$, $(3.45,1.8)$, $(2.7,2.2)$, $(1,2.4)$, $(0.2,1.3)\}$. We used the separable covariance function defined by $\mathrm{Cov}[Z(s_1,t_1), Z(s_2,t_2)] = C_{\mathrm{trunc}}(\|s_1 - s_2\|, 0.3) C_{\mathrm{trunc}}(|t_1 - t_2|, 3.5)$, where

| Complexity Type | Broadcast | | Distributed PGD | |
|---|---|---|---|---|
| | Worst | Average | Worst | Average |
| Communication | $O(m^2 n)$ | $O(mn)$ | $O\left(m^2 \sqrt[d]{m}\right)$ | $O\left(m^2 \sqrt[d]{m}\right)$ |
| Time | $O(n \sqrt[d]{m})$ | $O(\frac{n}{m} \sqrt[d]{m})$ | $O\left(m \sqrt[d]{m}\right)$ | $O\left(m \sqrt[d]{m}\right)$ |
| Space | $O(n^2)$ | $O(n^2)$ | $O\left(n^2\right)$ | $O\left(\left(\frac{n}{m}\right)^2\right)$ |

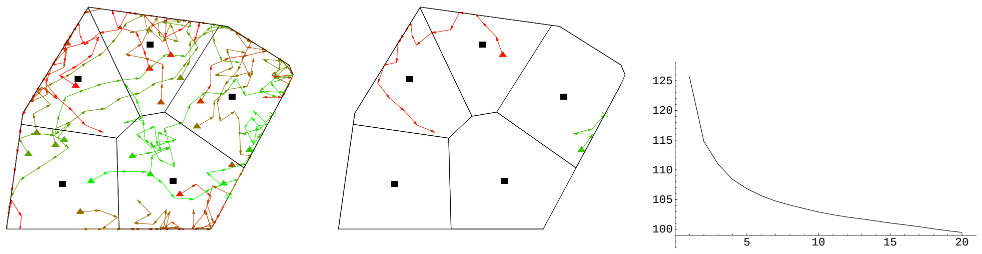
Table V. Algorithm complexities. The worst and average cases are over distributions of samples, with the average corresponding to a uniform distribution in $\mathcal{D}$. The time and communication complexities for the broadcast method are lower bounds, not simultaneously achievable by any known algorithm.

$$C_{\text{trunc}}(\delta, r_{\text{s}}) = \begin{cases} e^{-15\left(\frac{\delta}{r_{\text{s}}}\right)^2} & \text{if } \delta \leq r_{\text{s}}, \\ 0 & \text{otherwise.} \end{cases}$$

While the covariance function is not $C^1$ everywhere, the difference lies within the error margin of the simulation. We use $\omega = 0.02$ and $u_{\text{max}} = 0.3$. The values of our hyperparameters were $\nu = 0.1$, $q = 2$, $\beta_0 = \mathbf{0}$, and $\mathbf{K}_0 = \boldsymbol{I}$. We simulated the sampled data by drawing random variables from the distribution $N(\beta_0, \sigma_0^2 \mathbf{K}_u)$, where $\sigma_0^2 = \frac{q\nu}{\nu - 2}$, the prior mean of $\sigma^2$, and $\mathbf{K}_u$ is the correlation matrix of $\underline{y}_u$. For the mean regression functions $f_i$, we used $\mathbf{f}(((x,y),t)) = (1,x,y)^T$. To illustrate the robustness to failure, $R_2$ ceased communications after timestep 2, and $R_5$ after timestep $4$. Figure 2 shows the trajectories taken by the robots. Note that the curve of the objective function begins with a steep decline, then begins to level off after a few iterations. This may be seen as a natural progression from an exploration phase, when much information is being gained about the covariance, to exploitation phase, where new samples are chosen mostly to optimize the conditional variance. This example is representative of cases for which the data samples lie within a reasonable range of the predictive model. In the cases where the samples do not match the model, the surface of $\tilde{\mathcal{A}}^{(k)}$ is relatively flat, signifying that the amount of information to be gained is not significantly different whether the agents move or not. As information is a model-dependent quantity, this is not surprising.

## 6. Conclusions and future work

We have considered a network of static computing nodes and mobile robotic sensing platforms taking measurements of a time-varying random process with covariance known up to a scaling parameter.



(a) (b) (c)

Figure 2. (a) Trajectories of all robots, (b) two representative robot trajectories and (c) evolution of the objective function. The filled squares represent the (static) positions of the nodes, and the filled triangles show the starting positions of the robots. The X's represent the positions of the two robots who dropped communication.

We have used a Bayesian approach, treating the field as a spatiotemporal Gaussian random process, and developed a novel approximation of the variance of the posterior predictive distribution which may be calculated in a sequential and distributed fashion. Using this formulation, we have developed a projected gradient descent algorithm which is distributed over the network of nodes and robots. We have examined the complexity of this approach, and compared it against the lower bound complexity of a more centralized "broadcast" method, showing that the distributed approach scales better with the number of mobile agents. Future work will focus on theoretical guarantees on the accuracy of the approximation $\tilde{\mathcal{A}}^{(k)}$ and on the robustness to failure of the proposed coordination algorithm. As mentioned in our discussion, special care must be taken when generating local approximations for the universal kriging model. A topic of future work will be to provide rigorous methods for handling this case.

## 7. Acknowledgments

The authors thank B. Sansó and H. Lee for several conversations on Bayesian spatial statistics. This research was partially supported by NSF CAREER Award ECS-0546871.

## REFERENCES

1. Gaudard M, Karvson M, Linder E, Sinha D. Bayesian spatial prediction. *Environmental and Ecological Statistics* 1999; **6**:147–171.

2. Lee ND, Zidek JV. *Statistical Analysis of Environmental Space-Time Processes*. Springer Series in Statistics, Springer: New York, 2006.
3. Chaloner K, Verdinelli I. Bayesian experimental design, a review. *Statistical Science* 1995; **10**(3):273–304.
4. Stein ML. The screening effect in kriging. *The Annals of Statistics* Feb 2002; **30**(1):298–323.
5. Furrer R, Genton MG, Nychka D. Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics* 2006; **15**(3):502–523.
6. Pukelsheim F. *Optimal Design of Experiments*, *Classics in Applied Mathematics*, vol. 50. SIAM: Philadelphia, PA, 2006.
7. Liski EP, Mandal NK, Shah KR, Sinha BK. *Topics in Optimal Design*, *Lecture Notes in Statistics*, vol. 163. Springer: New York, 2002.
8. Willcox JS, Bellingham JG, Zhang Y, Baggeroer AB. Performance metrics for oceanographic surveys with autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering* 2001; **26**(4):711–725.
9. Popa DO, Sreenath K, Lewis FL. Robotic deployment for environmental sampling applications. *International Conference on Control and Automation*, Budapest, Hungary, 2005; 197–202.
10. Leonard NE, Paley D, Lekien F, Sepulchre R, Fratantoni DM, Davis R. Collective motion, sensor networks and ocean sampling. *Proceedings of the IEEE* 2007; **95**(1):48–74.
11. Zhang F, Fiorelli E, Leonard NE. Exploring scalar fields using multiple sensor platforms: tracking level curves. *IEEE Conf. on Decision and Control*, New Orleans, LA, 2007; 3579–3584.
12. Cortés J. Distributed Kriged Kalman filter for spatial estimation. *IEEE Transactions on Automatic Control* 2010; **55**(4). To appear.
13. Lynch KM, Schwartz IB, Yang P, Freeman RA. Decentralized environmental modeling by mobile sensor networks. *IEEE Transactions on Robotics* 2008; **24**(3):710–724.
14. Martínez S. Distributed interpolation schemes for field estimation by mobile sensor networks. *IEEE Transactions on Control Systems Technology* 2009; To appear.
15. Demetriou MA, Hussein II. Estimation of spatially distributed processes using mobile spatially distributed sensor network. *SIAM Journal on Control and Optimization* 2009; **48**(1):266–291.
16. Hoffmann GM, Tomlin CJ. Mobile sensor network control using mutual information methods and particle filters. *IEEE Transactions on Automatic Control* 2009; Submitted.
17. Kitanidis PK. Parameter uncertainty in estimation of spatial functions: Bayesian analysis. *Water Resources Research* 1986; **22**:449–507.
18. Wiens DP. Robustness in spatial studies I: minimax prediction and II: minimax design. *Environmetrics* 2005; **16**:191–203 and 205–217.
19. Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
20. Krause A, Guestrin C. Nonmyopic active learning of Gaussian processes: an exploration-exploitation approach. *International Conference on Machine Learning*, Corvallis, OR, 2007.
21. Graham R, Cortés J. A cooperative deployment strategy for optimal sampling in spatiotemporal estimation. *IEEE Conf. on Decision and Control*, Cancun, Mexico, 2008; 2432–2437.
22. Graham R, Cortés J. Cooperative adaptive sampling of random fields with unknown covariance. *American Control Conference*, St. Louis, MI, 2009. 4543-4548.
23. Bertsekas DP. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control* 1976; **21**(2):174–184.
24. Leonard T, Hsu JSJ. *Bayesian Methods*, *Cambridge series in statistical and probabilistic mathematics*, vol. 1. Cambridge

University Press: Cambridge, UK, 1999.

25. Robert CP, Casella G. *Monte Carlo statistical methods*. Springer texts in statistics, Springer: New York, 2004.
26. Bertsekas DP, Tsitsiklis JN. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.
27. Olfati-Saber R, Murray RM. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control* 2004; **49**(9):1520–1533.
28. Bullo F, Cortés J, Martínez S. *Distributed Control of Robotic Networks*. Applied Mathematics Series, Princeton University Press, 2009. Electronically available at http://coordinationbook.info.
29. Lynch NA. *Distributed Algorithms*. Morgan Kaufmann, 1997.
30. Peleg D. *Distributed Computing. A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications, SIAM, 2000.
31. Mohar B. The Laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, vol. 2, Alavi Y, Chartrand G, Oellermann OR, Schwenk AJ (eds.). John Wiley, 1991; 871–898.
32. Carle J, Myoupo JF. Topological properties and optimal routing algorithms for three dimensional hexagonal grid networks. *High performance computing in the Asia-Pacific region*, 2000; 116–121.
33. Chen MS, Shin KG, Kandlur DD. Addressing, routing, and broadcasting in hexagonal mesh multiprocessors. *IEEE Transactions on Computers* jan 1990; **39**(1):10–18.
34. Okabe A, Boots B, Sugihara K, Chiu SN. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. 2 edn., Wiley Series in Probability and Statistics, John Wiley, 2000.
35. Bernstein DS. *Matrix Mathematics*. Princeton University Press: Princeton, NJ, 2005.
36. Udwadia FE. Some convergence results related to the JOR iterative method for symmetric, positive-definite matrices. *Applied Mathematics and Computation* 1992; **47**(1):37–45.

## APPENDIX

### I. Predictions with a subset of measurements

We present here a series of results concerning the relationship between subsets of sample locations and hypothetical predictions made from partial information. Let $\underline{y} \in \mathbb{R}^n$ denote a full set of measurements at locations $\underline{x} \in \mathcal{D}_e^n$. Let $n_1, n_2 \in \mathbb{N}$ such that $n_1 + n_2 = n$. Consider a partition of the measurements $\underline{y} = (\underline{y}_1^T, \underline{y}_2^T)^T$ such that $\underline{y}_1 \in \mathbb{R}^{n_1}$ and $\underline{y}_2 \in \mathbb{R}^{n_2}$ and a similar partition of $\underline{x}$. We will use $\mathbf{K}_1$, respectively $\mathbf{K}_2$, to denote the correlation matrix of locations $\underline{x}_1$, respectively $\underline{x}_2$, and analogous notation for the matrices $\mathbf{F}_1, \mathbf{F}_2, \mathbf{E}_1, \mathbf{E}_2$. Let $\mathbf{K}_{12} = \mathbf{K}_{21}^T \in \mathbb{R}^{n_1 \times n_2}$ denote the matrix of cross-correlation between the two location vectors.

We begin with a multivariate version of the posterior predictive variance from Proposition 2.1, which can be considered the hypothetical distribution of the measurements at locations $\underline{x}_2$ given the samples $\underline{y}_1$. As in the univariate case, this result can be obtained by applying Bayes Theorem to the prior model.

**Lemma I.1 (Multivariate posterior predictive distribution)** *Under the Bayesian model* (2), *the multivariate posterior predictive distribution at locations* $\underline{x}_2 \in \mathcal{D}_e^{n_2}$ *from data* $\underline{y}_1$ *is the* $n_2$*-variate shifted Students t distribution with* $\nu + n_1$ *degrees of freedom, which takes the form,*

$$\underline{y}_2|\underline{y}_1, \underline{x} \propto \mathrm{Var}[\underline{y}_2|\underline{y}_1, \underline{x}]^{-\frac{1}{2}} \left(1 + \frac{\left(\underline{y}_2 - \mathrm{E}[\underline{y}_2|\underline{y}_1, \underline{x}]\right)^T \mathrm{Var}[\underline{y}_2|\underline{y}_1, \underline{x}]^{-1} \left(\underline{y}_2 - \mathrm{E}[\underline{y}_2|\underline{y}_1, \underline{x}]\right)}{\nu + n_1 - 2}\right)^{-\frac{\nu+n_1+n_2}{2}}.$$

*Here, the expectation is given by*

$$\mathrm{E}[\underline{y}_2|\underline{y}_1, \underline{x}] = \xi_{2|1}^T(\mathbf{E}_1 + \mathbf{K}_0^{-1})^{-1}\left(\hat{\beta}_1 + \mathbf{K}_0^{-1}\beta_0\right) + \mathbf{K}_{21}\mathbf{K}_1^{-1}\underline{y}_1,$$

*where* $\hat{\beta}_1 = \mathbf{E}_1^{-1}\mathbf{F}_1\mathbf{K}_1^{-1}\underline{y}_1$, *and* $\xi_{2|1} = \mathbf{F}_2 - \mathbf{F}_1\mathbf{K}_1^{-1}\mathbf{K}_{12}$. *The covariance matrix is given by*

$$\mathrm{Var}[Z|\underline{y}, \underline{x}] = \varphi(\underline{y}_1, \underline{x}_1)\phi(\underline{x}_2; \underline{x}_1),$$

*where, with a slight abuse of notation, we have used* $\phi(\underline{x}_2; \underline{x}_1)$ *to denote the following multivariate extensions of* $\phi$ *and* $\varphi$,

$$\begin{aligned}
\phi(\underline{x}_2; \underline{x}_1) &= \mathbf{K}_2 - \mathbf{K}_{21}\mathbf{K}_1^{-1}\mathbf{K}_{12} + \xi_{2|1}^T\left(\mathbf{K}_0^{-1} + \mathbf{E}_1\right)^{-1}\xi_{2|1}, \\
\varphi(\underline{y}_1, \underline{x}_1) &= \frac{1}{\nu + n_1 - 2}\left(q\nu + \left(\underline{y}_1 - \mathbf{F}_1^T\beta_0\right)^T\left(\mathbf{K}_1 + \mathbf{F}^T\mathbf{K}_0\mathbf{F}\right)^{-1}\left(\underline{y}_1 - \mathbf{F}_1^T\beta_0\right)\right).
\end{aligned}$$

If we treat all parameters as known then a generalized least squares (GLS) technique may be used to estimate one vector of samples from the values of another. The following lemma is useful for isolating a correlated block of samples in a GLS estimate.

**Lemma I.2 (Generalized least squares by block)** *Assume that* $\underline{y}_1$ *is partitioned according to* $\underline{y}_1 = (\underline{y}_{11}^T, \underline{y}_{12}^T)^T$, *and assume that* $\mathrm{Cor}[\underline{y}_{11}, \underline{y}_{12}] = \mathbf{0}$ *and* $\mathrm{Cor}[\underline{y}_{11}, \underline{y}_2] = \mathbf{0}$. *Let* $\hat{\underline{y}}_{LS} = \mathbf{K}_{21}\mathbf{K}_1^{-1}\underline{y}_1$ *be the generalized least squares estimate of* $\underline{y}_2$ *based on samples* $\underline{y}_1$ *(conditional on all parameters). Then we may write,*

$$\hat{\underline{y}}_{LS} = \mathrm{Cor}[\underline{y}_2, \underline{y}_{12}]\mathrm{Cor}[\underline{y}_{12}, \underline{y}_{12}]^{-1}\underline{y}_{12},$$

*i.e., the generalized least squares estimate may be calculated from only those samples in the block correlated to* $\underline{y}_2$.

**Proof.** Since the two parts of $\underline{y}_1$ are uncorrelated, we have,

$$\mathbf{K}_1^{-1} = \begin{bmatrix} \mathrm{Cor}[\underline{y}_{11}, \underline{y}_{11}] & \mathbf{0} \\ \mathbf{0} & \mathrm{Cor}[\underline{y}_{12}, \underline{y}_{12}] \end{bmatrix}^{-1} = \begin{bmatrix} \mathrm{Cor}[\underline{y}_{11}, \underline{y}_{11}]^{-1} & \mathbf{0} \\ \mathbf{0} & \mathrm{Cor}[\underline{y}_{12}, \underline{y}_{12}]^{-1} \end{bmatrix}.$$

Multiplying by the matrix $\mathbf{K}_{12} = (\mathbf{0}, \mathrm{Cor}[\underline{y}_2, \underline{y}_{12}])$ yields the result. ■

The GLS approximation arises naturally from partitioning the elements of the term $\mathbf{K}^{-1}\underline{y}$. The following lemma allows us to write this in terms of the GLS error.

**Lemma I.3 (Generalized least squares approximations)** *Let $\hat{\underline{y}}_{LS} = \mathbf{K}_{21}\mathbf{K}_1^{-1}\underline{y}_1$ be the generalized least squares estimate of $\underline{y}_2$ based on samples $\underline{y}_1$ (conditional on all parameters) and let $\overline{y}_{LS} = \underline{y}_2 - \hat{\underline{y}}_{LS}$. Then we can write,*

$$\mathbf{K}^{-1}\underline{y} = \begin{bmatrix} \mathbf{K}_1^{-1}\underline{y}_1 \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} -\mathbf{K}_1^{-1}\mathbf{K}_{12}\left(\mathbf{K}_1 \,|\mathbf{K}\right)^{-1}\overline{y}_{LS} \\ \left(\mathbf{K}_1 \,|\mathbf{K}\right)^{-1}\overline{y}_{LS} \end{bmatrix}. \tag{9}$$

*Furthermore, if $\underline{y}_1$ and $\underline{y}_2$ are uncorrelated (i.e., $\mathbf{K}_{12} = \mathbf{0}$), then this equation takes the form,*

$$\mathbf{K}^{-1}\underline{y} = \begin{bmatrix} \mathbf{K}_1^{-1}\underline{y}_1 \\ \mathbf{K}_2^{-1}\underline{y}_2 \end{bmatrix}. \tag{10}$$

**Proof.** Equation (9) is a direct result of the Schur complement form for the inverse of a partitioned matrix (see, e.g. [35, Proposition 2.8.7]). Equation (10) follows by evaluating at $\mathbf{K}_{12} = \mathbf{0}$. ■

In the sequel, we will find useful the matrix $\mathcal{M} \in \mathbb{R}^{(n+p)\times(n+p)}$ and vector $\mathcal{U} \in \mathbb{R}^{n+p}$ defined as,

$$\mathcal{M} = \begin{bmatrix} \mathbf{K} & \mathbf{F}^T \\ \mathbf{F} & -\mathbf{K}_0^{-1} \end{bmatrix}, \qquad \mathcal{U} = \begin{bmatrix} \underline{y} \\ -\mathbf{K}_0^{-1}\beta_0 \end{bmatrix}.$$

The following lemma gives a useful a restatement of the sigma mean.

**Lemma I.4 (Restated sigma mean)** *The term $\varphi(\underline{y}, \underline{x})$ may be written as,*

$$\begin{aligned}\varphi(\underline{y}, \underline{x}) = q\nu + \beta_0^T\mathbf{K}_0^{-1}\beta_0 + \underline{y}^T\mathbf{K}^{-1}\underline{y} - \\ - \left(\mathbf{K}_0^{-1}\beta_0 + \mathbf{F}\mathbf{K}^{-1}\underline{y}\right)^T\left(\mathbf{K}_0^{-1} + \mathbf{E}\right)^{-1}\left(\mathbf{K}_0^{-1}\beta_0 + \mathbf{F}\mathbf{K}^{-1}\underline{y}\right).\end{aligned} \tag{11}$$

**Proof.** Since $\mathbf{K}$ and $\mathbf{K}_0$ are positive definite, the matrix $\mathbf{K}+\mathbf{F}^T\mathbf{K}_0\mathbf{F}$ is also positive definite. Therefore the matrix $\mathcal{M}$ defined above is nonsingular. We can write,

$$\varphi(\underline{y}, \underline{x}) = q\nu + \beta_0^T\mathbf{K}_0^{-1}\beta_0 + \mathcal{U}^T\mathcal{M}^{-1}\mathcal{U}. \tag{12}$$

Using [35, Proposition 2.8.7] for the inverse of a partitioned matrix, we arrive at the result. ■

**Proposition I.5 (Approximate conditional variance)** *The term $\phi(x_0; \underline{x})$ may be written in terms of locations $\underline{x}_2$ as,*

$$\phi(x_0;\underline{x}) = \phi(x_0;\underline{x}_2) - (\mathbf{k}_1 - \mu_1)^T \, \phi(\underline{x}_1;\underline{x}_2)^{-1} \, (\mathbf{k}_1 - \mu_1) \, , \text{ where}$$
$$\mu_1 = \begin{bmatrix} \mathbf{K}_{21} \\ \mathbf{F}_1 \end{bmatrix}^T M_2^{-1} \begin{bmatrix} \mathbf{k}_2 \\ \mathbf{f}(x_0) \end{bmatrix}, \qquad M_2 = \begin{bmatrix} \mathbf{K}_2 & \mathbf{F}_2^T \\ \mathbf{F}_2 & -\mathbf{K}_0^{-1} \end{bmatrix}$$
$$\mathbf{k}_1 = \text{Cor}[Z(x_0), \underline{y}_1], \qquad \mathbf{k}_2 = \text{Cor}[Z(x_0), \underline{y}_2].$$

*Therefore $\phi(x_0;\underline{x}) \le \phi(x_0;\underline{x}_2)$ with equality if and only if $\mathbf{k}_1 = \mu_1$.*

**Proof.** First, we note that the conditional variance can be written using $\mathcal{M}$ as,

$$\phi(x_0;\underline{x}) = \text{Cor}[Z, Z] - \begin{bmatrix} \mathbf{k} \\ \mathbf{f}(x_0) \end{bmatrix}^T \mathcal{M}^{-1} \begin{bmatrix} \mathbf{k} \\ \mathbf{f}(x_0) \end{bmatrix}.$$

Next, we point out that with the proper partitioning of $\mathcal{M}$, the matrix $\phi(\underline{x}_1;\underline{x}_2)$ is the Schur Complement, $(M_2 \,|\mathcal{M}\,)$. Using this, and a similar partition of the vector $\mathbf{k}$, one arrives at the result. ■

The following lemma, while not directly related to partitions of measurements, is instrumental in approximating the sigma mean from a subset of samples. The proof is a result of applying Bayes Theorem to the prior model.

**Lemma I.6 (Posterior distribution of $\sigma^2$)** *Under the Bayesian model* (2)*, the posterior distribution of $\sigma^2$ after incorporating samples $\underline{y}$ is an inverse Gamma distribution with mean $\varphi\left(\underline{y}, \underline{x}\right)$ and variance $\frac{4\varphi(\underline{y},\underline{x})^2}{(\nu+n-4)}$.*

## II. Near optimal relaxation parameter for JOR

Here we present some results regarding a relaxation parameter for the JOR algorithm which is nearly optimal with respect to the rate of convergence of the algorithm for a certain class of matrices. Specifically we are interested in the class of symmetric, positive definite matrices $C$ with ones on the diagonal. Let $y(t) = (y_1(t), \ldots, y_n(t))^T \in \mathbb{R}^n$ be the vector updated during the JOR iteration in (4). Let $e(t) = \|C^{-1}y - y(t)\|$ denote the error at iteration $t$. We may write,

$$e(t) \le (\text{sprad}(\boldsymbol{I} - hC))^t \, e(0),$$

giving a bound on the error at step $t$ based on the initial error. The value of spread$(\boldsymbol{I} - hC)$ therefore controls the rate of convergence, and choosing the relaxation parameter, $h$, is of vital importance. Throughout this section we will use the shorthand $\lambda_{\text{max}} = \lambda_{\text{max}}(C)$ and $\lambda_{\text{min}} = \lambda_{\text{min}}(C)$. The work [36] provides results concerning the convergence of the JOR algorithm, including an optimal relaxation parameter, which in our case is equivalent to $h_{\text{opt}} = \frac{2}{\lambda_{\text{max}}+\lambda_{\text{min}}}$. In this section we will introduce an approximation to this optimal value which may be calculated in a distributed manner.

**Proposition II.1.** *Assume that $C \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix with all diagonal entries equal to $1$. Let $\beta, \alpha \in \mathbb{R}_{\geq 0}$ denote the maximum off-diagonal entry of $C$, respectively the maximum off-diagonal row sum of $C$, i.e.,*

$$\beta = \max_{i\neq j\in\{1,\dots,n\}} \{c_{ij}\}, \qquad \alpha = \max_{i\in\{1,\dots,n\}} \Big\{\sum_{\substack{j=1\\ j\neq i}}^{n} c_{ij}\Big\}.$$

*Let $h^* = \frac{2}{2+\alpha-\beta}$. Then using $h^*$ as the relaxation parameter in the JOR algorithm to solve $y = C^{-1}b$ results in guaranteed convergence.*

**Proof.** Convergence is guaranteed as long as $h^* \in \left(0, \frac{2}{\lambda_{\text{max}}}\right)$. Since $C$ is symmetric positive definite with $1$'s on the diagonal, all off-diagonal entries must have magnitude strictly less than $1$. Thus $1 - \beta > 0$. The Gershgorin circle theorem tells us that $\lambda_{\text{max}} \leq 1 + \alpha$. Together these two results yield the inequality, $2 + \alpha - \beta > \lambda_{\text{max}}$ which implies that $\frac{2}{2+\alpha-\beta} < \frac{2}{\lambda_{\text{max}}}$. The result follows. ■

**Lemma II.2.** *Under the assumptions of Proposition II.1, $h^* \leq \frac{1}{\lambda_{\text{min}}}$, with equality if and only if $C$ is the $n \times n$ identity matrix.*

**Proof.** First, note the following implication chain,

$$\lambda_{\text{min}} \leq 1 \implies 2\lambda_{\text{min}} \leq 2 + \alpha - \beta \implies h^* \leq \frac{1}{\lambda_{\text{min}}}.$$

Now, assume that $h^* = \frac{1}{\lambda_{\text{min}}}$. This implies that $\lambda_{\text{min}} = 1 + \alpha - \beta$, but $\lambda_{\text{min}} \leq 1$, and $\alpha \geq \beta$. So we must have $\lambda_{\text{min}} = 1$. Since the diagonal entries of $C$ are all $1$, the smallest eigenvalue can only be $1$ if all off-diagonal entries are zero, i.e., if $C = \boldsymbol{I}_n$. ■

**Lemma II.3.** *Under the assumptions of Proposition II.1, $|1 - h^*\lambda_{\text{min}}| \geq |1 - h^*\lambda_{\text{max}}|$.*

**Proof.** Using Lemma II.2, we have $|1 - h^*\lambda_{\text{min}}| = 1 - h^*\lambda_{\text{min}}$. The result may then be shown by two separate cases. First, note that if $h^* \leq \frac{1}{\lambda_{\text{max}}}$ then we have,

$$|1 - h^*\lambda_{\text{max}}| = 1 - h^*\lambda_{\text{max}} \leq |1 - h^*\lambda_{\text{min}}|,$$

so the result holds in this case. For the second case, assume that $h^* > \frac{1}{\lambda_{\text{max}}}$. Then $|1 - h^*\lambda_{\text{max}}| = h^*\lambda_{\text{max}} - 1$. The inclusion principle and the fact that $C$ is positive definite give us the bounds $0 < \lambda_{\text{min}} \leq 1 - \alpha$. Combined with the previously mentioned Gershgorin bound, $\lambda_{\text{max}} \leq 1 + \alpha$, this allows us to write,

$$\begin{aligned}\frac{\lambda_{\text{max}} + \lambda_{\text{min}}}{2 + \alpha - \beta} &\leq 1 \\ 2\frac{\lambda_{\text{max}} + \lambda_{\text{min}}}{2 + \alpha - \beta} &\leq 2 \\ h^*\left(\lambda_{\text{max}} + \lambda_{\text{min}}\right) &\leq 2 \\ h^*\lambda_{\text{max}} - 1 &\leq 1 - h^*\lambda_{\text{min}}.\end{aligned}$$

Thus in all cases, $|1 - h^*\lambda_{\text{max}}| \leq |1 - h^*\lambda_{\text{min}}|$. ■

**Proposition II.4.** *Under the assumptions of Proposition II.1, further assume that $\lambda_{min} \geq \epsilon_\lambda$ for some $\epsilon_\lambda \in (0, 1)$. Then $0 \leq$ sprad$(\boldsymbol{I} - h^*C) < 1 - \frac{2\epsilon_\lambda}{2+\alpha-\beta}$*

**Proof.** First note that the spectral radius is given by $\max\{|1 - h^*\lambda_{\text{min}}|, |1 - h^*\lambda_{\text{max}}|\}$, and is clearly nonnegative. From Lemma II.3, we have sprad$(\boldsymbol{I} - h^*C) = |1 - h^*\lambda_{\text{min}}|$. From Lemma II.2, we can infer sprad$(\boldsymbol{I} - h^*C) = 1 - h^*\lambda_{\text{min}}$. The upper bound follows by comparing $1 - h^*\lambda_{\text{min}}$ and comparing against $1 - h^*\epsilon_\lambda$. ■