Distributed Gaussian Process Regression Under Localization Uncertainty

Sungjoon Choi
Department of Electrical and
Computer Engineering
ASRI, Seoul National University
Email: sungjoon.choi@cpslab.snu.ac.kr

Mahdi Jadaliha
Department of Mechanical Engineering
Michigan State University
Email: jadaliha@egr.msu.edu

Jongeun Choi
Department of Mechanical Engineering
Department of Electrical and Computer Engineering
Michigan State University
Email: jchoi@egr.msu.edu

Songhwai Oh *
Department of Electrical and Computer Engineering, ASRI
Seoul National University
Email: songhwai.oh@cpslab.snu.ac.kr

In this paper, we propose distributed Gaussian process regression for resource-constrained distributed sensor networks under localization uncertainty. The proposed distributed algorithm, which combines Jacobi over-relaxation and discrete-time average consensus, can effectively handle localization uncertainty as well as limited communication and computation capabilities of distributed sensor networks. We also extend the proposed method hierarchically using sparse Gaussian process regression to improve its scalability. The performance of the proposed method is verified in numerical simulations against the centralized maximum a posteriori solution and a quick-and-dirty solution. We show that the proposed method outperforms the quick-and-dirty solution and achieves an accuracy comparable to the centralized solution.

1 Introduction

The advances in embedded processors and distributed sensor networks technologies allow a number of important and successful applications in environmental monitoring, such as monitoring complex interactions in wildlife habitats and disaster management of harmful algal blooms in water bodies [1]. A distributed sensor network consists of a number of resource-constrained agents with limited processing power, communication bandwidth, and battery capacity, to name a few. These limitations play an important role in designing an application using distributed sensor networks [1]. In order to handle these physical constraints and take a full capability as a team, it is important to process information in a distributed manner [1–4]. In [4], a distributed learning and control algorithm is developed for mobile sensor networks for estimating an unknown field of interest from noisy measurements and coordinating multiple agents to discover peaks of the unknown field. In [5], a distributed Gaussian process regression algorithm is proposed using compactly supported covariance functions.

Gaussian process regression (GPR) has been widely used to model spatio-temporal phenomena of changing environments [6]. Due to its Bayesian non-parametric property, it can be easily deployed to an actual environment without a tedious parameter tuning step. Moreover, GPR can provide predictive uncertainties which can be used as a metric for the optimal sensor placement as well as cooperative control for exploration [7, 8].

While GPR is effective for a number of practical applications, GPR suffers from two major drawbacks: its heavy computational cost and the difficulty of handling localization errors. The computational complexity grows as $O(N^3)$, where N is the number of training data. A number of approximation schemes have been proposed to reduce the computational complexity of GPR, including [9–11]. In [8], Xu et al. proposed Gaussian process regression based on truncated observations for mobile sensor networks with limited memory and computational power. In particular, Chen et al. [12] proposed a decentralized version of partially independent training conditional (PITC), a sparse Gaussian process regression

*Corresponding Author

method described in [11]. By decentralizing the algorithm, they boosted the speed of PITC while guaranteeing that its predictive performance is equivalent to that of the centralized version.

For environmental monitoring, better mapping of the environment is possible when accurate sensing locations are available. However, there can be many situations where sensing locations cannot be measured accurately, e.g., GPS denied areas. In addition, there can be a significant localization error with inexpensive GPS receivers. A number of localization algorithms have been proposed to address this issue, including [13, 14]. Oguz-Ekim et al. [13] iteratively maximized likelihoods of position estimates given measurements and Karlsson et al. [14] applied particle filtering for surface and underwater navigation as a supplement to the GPS. Mysorewala et al. [15] combined a neural network and an extended Kalman filter (EKF) with greedy search heuristics and developed a distributed multi-scale sampling strategy for environmental monitoring. The use of an EKF allowed the method to handle localization and measurement uncertainties. However, the proposed method is based on a parametric model and it is difficult to apply the method directly to highly complex time-varying real-world situations.

Jadaliha et al. [16] incorporated the localization uncertainty into the Gaussian process regression framework. Since the proposed predictive mean and variance estimators have no closed-form solutions, they suggested two approximation schemes, Laplace approximation and Monte Carlo importance sampling. They also proposed a simple Laplace approximation method which uses maximum a posteriori (MAP) estimates of noisy position reports. In this paper, we extend [16] by developing a distributed version of GPR, which can handle both localization and measurement uncertainties, such that it can be suitable for resource-constrained distributed sensor networks. In contrast to the preliminary version [17], we extend the proposed algorithm hierarchically to improve its scalability in this paper. The proposed hierarchical method first forms groups of agents using distributed spectral clustering based on the network topology. Then each group estimates the field of interest in a distributed manner using the proposed distributed Gaussian process regression. Finally, group estimates are combined using the decentralized sparse GPR method [12].

The remainder of this paper is organized as follows. In Section 2, Gaussian process regression is briefly described. In Section 4 and 5, we describe the problem formulation and present a method for estimating predictive statistics, which incorporate both localization and measurement errors into a single Bayesian framework. In Section 6, we propose a distributed Gaussian process algorithm for estimating predictive statistics. The hierarchical extension of the distributed Gaussian process regression algorithm is described in Section 7. The performance of the proposed algorithm is extensively evaluated in Section 8.

2 Gaussian Process Regression

A Gaussian process (GP) is completely specified by its mean function and covariance function and it is formally defined as a collection of random variables, any finite number of which have a joint Gaussian distribution [6]. Let us denote the mean function by $m(x)$ and the covariance function by $k(x,x')$ for a Gaussian process $f(x)$ describing an environmental parameter. Then $f(x)$ can be represented as:

$$f(x) \sim GP(m(x),\ k(x,x')). \tag{1}$$

Suppose that $x \in \mathbb{R}^{n_x}$ is an input and $y \in \mathbb{R}$ is an output (or a target), such that $y = f(x) + w$, where w is a white Gaussian noise. When the target y is continuous (respectively, discrete), we have a regression (respectively, classification) problem. In this paper, we assume y takes a continuous value. Suppose that there are N observations, $\{(x_i, y_i) | i = 1, \cdots, N\}$. Given observations, Gaussian process regression (GPR) can predict an output $y_\star$ for a new input vector $x_\star$. Throughout this paper, it is assumed that $x \in \mathbb{R}^2$ and $y \in \mathbb{R}$.

For notational simplicity, we assume a zero-mean Gaussian process. A Gaussian process with a nonzero mean can be treated by a change of variables. There are a number of choices for the covariance function and a widely used covariance function is the squared exponential (SE) kernel:

$$k(x,x') = \sigma_f^2 \exp\left(-\frac{||x-x'||^2}{2\sigma_x^2}\right), \tag{2}$$

where σ_f^2 and σ_x^2 are hyperparameters which can be estimated by maximizing the log-likelihood [6].

Assuming that $y_i = f(x_i) + w_i$ and $w_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma_w^2)$, the covariance between y_i and y_j can be computed as

$$\mathbf{cov}(y_i, y_j) = k(x_i, x_j) + \sigma_w^2 \delta_{ij} \tag{3}$$

and we represent the covariance in the following matrix form

$$\mathbf{cov}(\mathbf{y}) = K(\mathbf{x},\mathbf{x}) + \sigma_w^2 I, \tag{4}$$

where $\mathbf{y} = (y_1, \ldots, y_N)^T$, $\mathbf{x} = (x_1^T, \ldots, x_N^T)^T$, and $K(\mathbf{x},\mathbf{x})$ is the covariance matrix computed from N data points.

Let $D^o = \{(x_i, y_i) \mid i = 1, \cdots, N\}$ be a set of input-output pairs. The conditional distribution of $y_\star$ at a new input $x_\star$ given data becomes

$$y_\star | D^o \sim N(\mu_\star(x_\star|D^o),\ \sigma_\star^2(x_\star|D^o)), \tag{5}$$

where

$$\mu_\star(x_\star|D^o) = k(x_\star, \mathbf{x})^T (K(\mathbf{x},\mathbf{x}) + \sigma_w^2 I)^{-1} \mathbf{y} \tag{6}$$

and

$$\sigma_\star^2(x_\star|D^o) = \sigma_f^2 - k(x_\star,\mathbf{x})^T(K(\mathbf{x},\mathbf{x})+\sigma_w^2 I)^{-1}k(x_\star,\mathbf{x}). \quad (7)$$

Here, $k(x_\star,\mathbf{x}) \in \mathbb{R}^N$ is a covariance vector between $\mathbf{y}$ and $y_\star$.

Note that (6) and (7) require an inversion of a covariance matrix, which has the time complexity of $O(N^3)$. Considering limited capabilities of distributed sensor networks, the computation can be prohibitive when the number of measurements becomes large. A number of approximation methods have been proposed to address this issue [9–11, 18]. A distributed algorithm, such as the one described in Section 6, can be considered as another solution to reduce the computational demand.

3 Distributed Spectral Clustering

In this section, we propose a distributed spectral clustering algorithm for a multi-agent system. We assume that a large number of agents capable of sensing at its location are deployed in a certain field and each agent can only communicate with its neighbors.

In order to perform spectral clustering [19] in a distributed manner, we first have to calculate top k eigenvectors of a Laplacian matrix computed from the network topology of a distributed sensor network. Decentralized orthogonal iteration (DOI) can be used in this purpose [20]. Once each agent a_i has its spectral feature, i-th row of a matrix whose column is top k eigenvectors of a Laplacian matrix, we can use primal-dual k-means algorithm to cluster each agent to groups [21].

3.1 Spectral Clustering

Spectral clustering is a method that finds good clusters using top k eigenvectors of a matrix derived from the distance between points. In [19], Ng. et al. proposed a spectral clustering algorithm based on a Laplacian matrix as well as detailed analysis of the algorithm using matrix perturbation theory. The overall spectral clustering algorithm is summarized in Algorithm 1.

In order to use the spectral clustering algorithm in a distributed sensor network, we have to compute top k eigenvectors of a Laplacian matrix and perform clustering in a distributed manner.

3.2 Distributed Spectral Analysis

Decentralized orthogonal iteration (DOI) computes the eigenvectors of a weighted graph, where the computation is performed at the nodes in the graph [20]. In a distributed sensor network, each sensor node can share information with its neighbors, and computes the corresponding weights between neighbors. Usually, a weight function is a non-increasing continuous function of a distance. The goal of this algorithms is for each node to compute its corresponding value of k principal eigenvectors.

Algorithm 1 Spectral Clustering

```
1: Given a set of points $S = \{s_i \in \mathbb{R}^l | i = 1, \dots, n\}$.
2: Form the affinity matrix $A \in \mathbb{R}^{n\times n}$ where $A_{ij} = exp(= ||s_i - s_j||^2/2\sigma_s^2)$ if $i \neq j$, and $A_{ii} = 0$.
3: Let $L = D^{-1/2}AD^{-1/2}$, where $D$ is a diagonal matrix whose $(i, i)$-element is the sum of $A$'s $i$-th row.
4: Let $X = [x_1 x_2 \dots x_k] \in \mathbb{R}^{n\times k}$ whose column is the top $k$ eigenvectors of $L$.
5: Let $Y$ be the normalized matrix of $X$, i.e., $Y_{ij} = X_{ij}/(\sum_j X_{ij}^2)^{1/2}$.
6: Cluster $S$ by treating $i$-th row of $Y$ as a feature of $i$-th point via K-means.
```

DOI computes k principal eigenvectors by emulating Orthogonal iteration (OI), a simple algorithm that computes the top k eigenvectors of a graph. Orthogonal iteration is summarized in Algorithm 2.

Algorithm 2 Orthogonal Iteration (OI)

```
1: Given $A \in \mathbb{R}^{n\times n}$
2: begin Choose a random $n \times k$ matrix $Q$.
3: loop
4:     Let $V = AQ$.
5:     Let $Q = Orthonormalize(V)$.
6: end loop
7: return $Q$ as the top $k$ eigenvectors of $A$.
```

In DOI, each agent is responsible for the corresponding rows of V and Q associated with it, denoted V_i and Q_i. Then $V = AQ$ can be calculated using only its local information , i.e., information of its own and from its neighbors with coeficieants A_{ij} indicating the weight between i-th and j-th agents.

The orthonomalization can be decentralized using the following property. Orthonormalized matrix $Q \in \mathbb{R}^{n\times k}$ can be obtained by finding $R \in \mathbb{R}^{k\times k}$ satisfying $V = QR$ and calculating $Q = VR^{-1}$. By using the fact that

$$K = V^TV = R^TQ^TQR = R^TR, \quad (8)$$

we can first compute K, then compute V. Since $K = \sum_i V_i^T V_i = \sum_I K_i$, where V_i is the i-th row of V, each agent can calculate K_i locally, and K by using a consensus algroithm. In DOI, we use push-sum algorithm shown in Algorithm 3 for this purpose. By combining push-sum algorithm with Cholesky factorization in (8) , we can formulate DOI for each i-th agent.

Algorithm 3 Push-Sum(B, $K^{(i)}$)

```
1: Given An arbitrary stochstic matrix $B \in \mathbb{R}^{n\times n}$, $K^{(i)}$
2: begin One agent $i$ starts with $w_i = 1$ , all others with $w_{ni} = 0$, $S_i = K^{(i)}$
3: loop
4:     Set $S_i = \sum_{j\in N(i)} b_{ji} S_j$
5:     Set $w_i = \sum_{j\in N(i)} b_{ji} w_j$
6: end loop
7: output $K^{(i)} = \frac{S_i}{w_i}$.
```

Algorithm 4 Decentralized Orthogonal Iteration (DOI)

```
1: Given $i$-th row of the adjacency matrix $A_i$
2: begin Chose a random vector $Q_i \in \mathbb{R}^k$
3: loop
4:     Set $V_i = \sum_{j\in N(i)} A_{ij} Q_j$
5:     Compute $K^{(i)} = V_i^T V_i$
6:     Set $K$ = Push-Sum($B$, $K^{(i)}$) in Algorithm 3
7:     Compute the Cholesky factorization$K = R^T R$
8:     Set $Q_i = V_i R^{-1}$
9: end loop
10: output $Q_i$ is the $i$-th componenct of each eigenvector
```

Algorithm 5 Primal-Dual k-means clustering

```
1: Given A learning rate $\mu > 0$.
2: begin Each agent $a_i$ randomly draws $\{\mathbf{c}_{i,k}^{(0)} \mid k \in \mathcal{K}\}$.
3: loop
4:     Each agent $a_i$ assigns its measurement $x_i \in \mathbb{R}^m$ to the cluster $C_k$ whose center $\mathbf{c}_{i,k}^{(t)}$ is closest to $x_i$.
5:     if $|C_{i,k}^{(t)}| \neq 0$ then
6:         Each agent $a_i$ updates centers $\mathbf{c}_{i,k}^{(t)}, k \in \mathcal{K}$ as $\mathbf{c}_{i,k}^{(t+1)} = \frac{1}{|C_{i,k}^{(t)}|}\left(\sum_{x_i\in C_{j,k}^{(t)}} x_i - \sum_{j\in N_i}(\lambda_{i,k}^{j\,(t)} - \lambda_{j,k}^{i\,(t)})\right)$
7:     else
8:         Each agent $a_i$ updates center $\mathbf{c}_{i,k}^{(t)}, k \in \mathcal{K}$ as $\mathbf{c}_{i,k}^{(t+1)} = \frac{1}{|N_i|}\sum_{j\in N_i}\left(c_{j,k} - \frac{\lambda_{i,k}^{i\,(t)} - \lambda_{j,k}^{i\,(t)}}{2\mu}\right)$
9:     end if
10:    Each agents $a_i$ updates the value of its Lagrange multipliers $\lambda_{i,k}^{i\,(t+1)}$, $k \in \mathcal{K}$, $j \in N_i$ as $\lambda_{i,k}^{j\,(t+1)} = \lambda_{i,k}^{j\,(t)} + \mu(\mathbf{c}_{i,k}^{(t+1)} - \mathbf{c}_{j,k}^{(t+1)})$.
11: end loop
12: output Each agents $a_i$ knows its affiliated cluster $C_k$.
```

3.3 Distributed Spectral Clustering

Once the eigenvectors of a Laplacian matrix is computed, we have to cluster the agents using corresponding eigenvectors in a distributed manner. In [21], Forero et al. proposed a decentralized k-means algorithm for clustering observations collected at spatially deployed wireless sensor network. Sensors get to know which clusters they belong to only by exchanging sufficient information between its neighbors. It is guaranteed to converge into local minimum of the centralized problem [22].

Consider a connected set of agents $\mathcal{A} = \{a_1, a_2, ..., a_J\}$, where each agent $a_i \in \mathcal{A}$ can only communicate with its neighbors, denoted by N_j. Each agent a_j in $\mathcal{A}$ is has its own value (measurement) denoted by $x_j \in \mathbb{R}^m$. Let $\mathcal{K} = \{1, 2, ..., K\}$ represent the set of classes and $\mathcal{C} = \{C_1, C_2, ..., C_K\}$ denotes the superset of set of agents corresponding to class $k \in \mathcal{K}$, e.g., if agent $1, 2, 5$ are belonged to cluster 3, then $c_3 = \{1, 2, 5\}$, and $\mathbf{c} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K\}$ denotes the center of each cluster. Then, primal-dual k-means algorithm is a distributed version of following optimization problem and summarized in Algorithm 5.

$$\hat{\mathcal{C}} = \arg\min_{\mathcal{C}} \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} \sum_{x_j\in c_k} \frac{1}{2}||x_j - c_k||^2 \tag{9}$$

Algorithm 6 Distributed Spectral Clustering

```
1: Compute $k$ principal eigenvectors of a weight grapah in a distributed manner using Algorithm 4
2: Perfrom distributed spectral clustering using Algorithm 5
```

Overall disributed spectral clustering algorithm is summarized in Algorithm 6.

4 Gaussian Process Regression Under Localization Uncertainty

In a realistic situation, acquiring samples $\{x_i, y_i\}$ with exact localization for x_i is often impossible. As Gaussian processes incorporate a measurement noise naturally, it is desirable to incorporate a localization noise into Gaussian processes. In [16], Gaussian process regression was reformulated to incorporate noisy location measurements $\bar{\mathbf{x}} = \{\bar{x}_1, \ldots, \bar{x}_N\}$, where $\bar{x}_i = x_i + v_i$ and v_i is a zero-mean white Gaussian noise with variance σ_v^2. Let $D = \{(\bar{x}_i, \bar{y}_i) \mid i = 1, \ldots, N\}$, where $\bar{y}_i = f(x_i) + w_i$. Under this new formulation, the following predictive mean estimator (PME) and predictive variance estimator (PVE) are derived in [16] as follows:

$$\mathbb{E}(y_\star|D) = \frac{\int_X \mu_\star(x_\star|D)p(\bar{\mathbf{y}}|\mathbf{x})p(\mathbf{x}|\bar{\mathbf{x}})d\mathbf{x}}{\int_X p(\bar{\mathbf{y}}|\mathbf{x})p(\mathbf{x}|\bar{\mathbf{x}})d\mathbf{x}} \tag{10}$$

and

$$\begin{aligned}\mathbf{var}(y_\star|D) &= \frac{\int_X (\sigma_\star^2(x_\star|D) + \mu_\star^2(x_\star|D))p(\bar{\mathbf{y}}|\mathbf{x})p(\mathbf{x}|\bar{\mathbf{x}})d\mathbf{x}}{\int_X p(\bar{\mathbf{y}}|\mathbf{x})p(\mathbf{x}|\bar{\mathbf{x}})d\mathbf{x}} \\ &\quad -\mathbb{E}^2(y_\star|D),\end{aligned} \tag{11}$$

where $x_\star$ is the location of interest and $\mu_\star(x_\star|D)$ and $\sigma^2_\star(x_\star|D)$ are given by (6) and (7).

This new formulation of Gaussian process regression incorporates both localization and observation noises in a Bayesian framework. However, there are no closed-form solutions for the PME and PVE given in (10) and (11), respectively. In [16], three approaches are proposed to approximate the PME and PVE and they are the fully-exponential Laplace approximation method [23–26], Monte Carlo importance sampling, and a simple Laplace approximation method. The simple Laplace approximation makes predictions based on the mode of the posterior distribution of the position of an input, i.e., the maximum a posteriori (MAP) estimator of the input, and described in the next section.

5 Simple Laplace Approximation

The simple Laplace approximation method for estimating (10) and (11) is based on the maximum a posteriori (MAP) estimation and requires less computation compared to the fully-exponential Laplace approximation method and Monte Carlo importance sampling [16]. The simple Laplace approximation method will be called MAP-GPR in the remainder of this paper. The following proposition from [16] demonstrates that MAP-GPR can provide good estimates.

Proposition 1. *Let $\hat{\mathbf{x}}$ be an asymptotic mode of order $O(n^{-1})$ for $-h(x)$, such that*

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\bar{\mathbf{y}}|\mathbf{x})p(\mathbf{x}|\bar{\mathbf{x}}) \tag{12}$$

$$h(\mathbf{x}) = -\frac{1}{n}\log(p(\bar{\mathbf{y}}|\mathbf{x})p(\mathbf{x}|\bar{\mathbf{x}})). \tag{13}$$

Assume that $\{h, \widehat{\mathbf{x}}\}$ satisfies the regularity conditions described in [16]. Now consider the following approximations for $\mathbb{E}(y_\star|D)$ and $\mathbf{var}(y_\star|D)$

$$\hat{\mathbb{E}}(y_\star|D) = k^T(x_\star, \widehat{\mathbf{x}})(K(\widehat{\mathbf{x}}, \widehat{\mathbf{x}}) + \sigma^2_w I)^{-1}\bar{\mathbf{y}}, \tag{14}$$

$$\widehat{\mathbf{var}}(y_\star|D) = \sigma^2_f - k^T(x_\star, \widehat{\mathbf{x}})(K(\widehat{\mathbf{x}}, \widehat{\mathbf{x}}) + \sigma^2_w I)^{-1}k(x_\star, \widehat{\mathbf{x}}), \tag{15}$$

where $x_\star$ is the location of interest and $K(\widehat{\mathbf{x}}, \widehat{\mathbf{x}}) \in \mathbb{R}^{n\times n}$ and $k(x_\star, \widehat{\mathbf{x}}) \in \mathbb{R}^n$ are covariance matrices obtained using $\widehat{\mathbf{x}}$ and $x_\star$. Then, we have the following orders of errors:

$$\hat{\mathbb{E}}(y_\star|D) = \mathbb{E}(y_\star|D) + O(n^{-1})$$
$$\widehat{\mathbf{var}}(y_\star|D) = \mathbf{var}(y_\star|D) + O(n^{-1}).$$

Remark 1. *Note that the MAP-GPR predictive mean and variance in (14) and (15) take the same form as the original predictive mean (6) and variance (7), but the MAP estimator $\widehat{\mathbf{x}}$ from (12) is used instead.*

6 Distributed GPR Algorithm

In this section, we introduce a distributed algorithm for an individual agent (sensor) to estimate an environmental parameter of the surveillance region S only by exchanging local information within r-disk neighbors. Consider a distributed sensor network consisting of q sensing agents distributed in S. This distributed approach can be implemented for a class of kernel functions $K(\cdot,\cdot)$ that have compact supports. The following kernel function is considered in the development of a distributed algorithm.

$$k(x, x') = \sigma^2_f \lambda\left(\frac{||x - x'||}{r}\right), \tag{16}$$

where

$$\lambda(h) = \begin{cases} (1-h)\cos(\pi h) + \frac{1}{\pi}\sin(\pi h), & \text{if } h \le 1, \\ 0, & \text{otherwise.} \end{cases}$$

Notice that the kernel function K in (16) has a compact support, i.e., $K_{ij} = K(x^{(i)}, x^{(j)})$ is non-zero if and only if $r_{ij} = ||x^{(i)} - x^{(j)}||$ is less than the support r and, similarly, $k_i = K(x^{(i)}, x_\star)$ is non-zero if and only if $d_{i\star} = ||x_i - x_\star||$ is less than the support r. Consider a case in which each agent in a sensor network can only communicate with other agents within a limited communication range of R. In addition, we assume that there exists no central station in the development of the distributed GPR algorithm.

The index of a distributed sensor is denoted using $I = \{1, \cdots, q\}$. The position of agent i is denoted by $x^{(i)}$. Agent i can only communicate with its neighbors in a limited range of R. The adjacency matrix Q indicates whether two agents are neighbors or not. If the element in the i-th row and j-th column of Q is one, i.e., $Q_{ij} = 1$, then agent i and agent j are neighbors and they have a communication link, and if $Q_{ij} = 0$, then they are not neighbors.

$$Q_{ij} = \begin{cases} 1, & \text{if } ||x^{(i)} - x^{(j)}|| \le R \text{ and } i \ne j \\ 0, & \text{otherwise,} \end{cases}$$

where R is the communication range between neighbors.

The assumptions made for the resource-constrained distributed sensor networks are listed as follows.

Assumption 1. *The radius r of the support of the kernel function (16) satisfies that $0 < r < R$.*

Assumption 2. *Agent i can only communicate with neighbors in $N(i) = \{j \in \mathrm{I} \mid Q_{ij} = 1\}$.*

These assumptions indicate that the communication range must be longer than the measurement radius of each agent. If these assumptions are not held, the multi-agent system cannot reach a consensus due to an inability to make communication with each other.

As a result of the specified covariance matrix in (16) and Assumption 1, agent i knows the i-th row of K, i.e., $[K]_{(i)}$, where $K_{ij} \neq 0$ if and only if $j \in N(i)$.

6.1 Jacobi Over-Relaxation

Jacobi over-relaxation (JOR) is a method for solving $Ax = b$, where $A \in \mathbb{R}^{N\times N}$ and $x, b \in \mathbb{R}^N$ [27]. This method assumes that agent i knows $[A]_{(i)} \in \mathbb{R}^N$ and b_i, and $a_{ij} = (A)_{ij} = 0$ if agent i and agent j are not neighbors which goes along with Assumption 2. The i-th element of the solution $x = A^{-1}b$ can be obtained by the following iterative step:

$$x_i^{(k+1)} = (1-h)x_i^{(k)} + \frac{h}{a_{ii}}\left(b_i - \sum_{j\in N_i} a_{ij}x_j^{(k)}\right), \quad (17)$$

where h is a constant controlling the speed of convergence.

The convergence property of the iterative JOR algorithm with respect to eigenvalues of matrix A is specified in [27]. Moreover, Udwadia et al. [28] proved that if $h < \frac{2}{N}$, the convergence of the JOR algorithm to the solution $x = A^{-1}b$ is guaranteed from any initial point, where A is a symmetric, positive-definite matrix.

Remark 2. *Note that, the solution of the JOR method approaches to the solution $x = A^{-1}b$ without requiring the computation of an inverse of A. As shown in Section 2, a drawback of Gaussian process regression is $O(N^3)$ computational complexity and $O(N^2)$ memory complexity of calculating $(K(\mathbf{x},\mathbf{x}) + \sigma_w^2 I)^{-1}$. However, when it comes to implementing GPR algorithm for distributed sensor networks where no centralized server exists, each resource-constrained agent can hardly handle these complexities. Moreover, even with presence of a centralized server capable of performing such computations , high communication costs are also required to transfer all the spatially distributed information to the server. But, with a distributed algorithm like JOR, each agent is only required to run a few simple operations, such as additions and multiplications. Furthermore, a distributed algorithm is more robust against changes in the network topology [29].*

6.2 Discrete-Time Average Consensus

The discrete-time average consensus (DAC) method is used to compute the arithmetic mean of elements of a vector [29]. Let

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} \in \mathbb{R}^N. \quad (18)$$

If the graph is connected, and agent i knows the i-th element of vector $\mathbf{c}$, the arithmetic mean of $\mathbf{c}$ can be computed by iterating

$$x_i^{(k+1)} = x_i^{(k)} + \varepsilon \sum_{j\in N_i} Q_{ij}(x_j^{(k)} - x_i^{(k)}), \quad (19)$$

with an initial condition $x_i^0 = c_i$, where $Q_{ij} = 1$ if and only if agent i and agent j are connected. It is proven that if ε satisfies

$$0 < \varepsilon < \frac{1}{\max_i(\sum_{j\neq i} Q_{ij})}, \quad (20)$$

then the DAC algorithm converges to the solution [30].

After the convergence, every node in the network knows the arithmetic mean of vector $\mathbf{c}$, i.e., $\frac{1}{N}\sum_{i=1}^N c_i$.

6.3 Distributed MAP Mode Estimation

In order to find the asymptotic mode $\widehat{\mathbf{x}}$ in a distributed manner, the Jacobi over-relaxation method introduced in Section 6.1 is used. Agent i only knows the i-th row and column of K and $\partial K/\partial x^{(i)}$. Note that for $\partial K/\partial x^{(i)}$, the elements that are not on the i-th row or the i-th column are zeros. Define Γ and B_l as follows:

$$\Gamma = (K(\mathbf{x},\mathbf{x}) + \sigma_w^2 I)^{-1}\bar{\mathbf{y}} \quad (21)$$

$$B_l = \frac{1}{2}(K(\mathbf{x},\mathbf{x}) + \sigma_w^2 I)^{-1}\frac{\partial K(\mathbf{x},\mathbf{x})}{\partial x^{(l)}}. \quad (22)$$

Agent i knows i-th row of $(K(\mathbf{x},\mathbf{x}) + \sigma_w^2 I)$ and i-th element of $\bar{\mathbf{y}}$. So $\Gamma^{(i)}$, the i-th element of Γ, can be computed by applying the JOR method over $(K(\mathbf{x},\mathbf{x}) + \sigma_w^2 I)^{-1}\bar{\mathbf{y}}$ based on the following recursion:

$$\begin{aligned}\Gamma^{(i)}(k+1) &= (1-\alpha)\Gamma^{(i)}(k) + \frac{\alpha}{\sigma_f^2 + \sigma_w^2} \\ &\times \left(\bar{y}^{(i)} - \sum_{j\in N(i)} k(\hat{x}^{(i)}, \hat{x}^{(j)})\Gamma^{(j)}(k)\right). \quad (23)\end{aligned}$$

Similarly, agent i knows $[(K + \sigma_w^2 I)]_{(i)}$, the i-th row of $(K + \sigma_w^2 I)$, and the i-th row of B_l for every $l \in \{1, 2, ..., q\}$. So B_{il}, the i-th row of B_l, can be computed by applying the JOR method. However, agent i needs to receive $\frac{\partial K}{\partial x^{(l)}}$ from agent l if they are connected or use zero instead if they are not connected. In other words, agent i will

$$\begin{cases} \text{get } \left[\frac{\partial K}{\partial x^{(l)}}\right]_{(i)} \text{ from agent } l & \text{if } i = l \text{ and } i \in N(l) \\ \text{set } \left[\frac{\partial K}{\partial x^{(l)}}\right]_{(i)} = 0 & \text{if } l \neq i \text{ and } i \notin N(l) \end{cases}$$

and the i-th row of B_l can be computed using

$$B_{il}(m+1) = (1-\alpha)B_{il}(m) + \frac{\alpha}{\sigma_f^2+\sigma_w^2} \times \left(\left[\frac{\partial K}{\partial x^{(l)}} \right]_{(i)} \Bigg|_{\mathbf{x}=\widehat{\mathbf{x}}} - \sum_{j \in N(i)} k(\widehat{\mathbf{x}}^{(i)}, \widehat{\mathbf{x}}^{(j)}) B_{jl}(m) \right) \quad (24)$$

for $m \in \mathbb{Z}_{\geq 0}$, $l \in N(i)$, and $i \in 1,2,\ldots,q$, where $\alpha \in (0, 2\lambda_{min}(K+\sigma_w^2 I))$. At the end of JOR iterations, agent i knows $\Gamma^{(i)}$ and B_{il}.

Proposition 2. *Given $B_{ji}^{(j)}$, the j-th element of B_{ji}, $\frac{\partial h}{\partial x^{(i)}}$ calculated for agent i is*

$$\frac{\partial h}{\partial x^{(i)}} = \frac{x^{(i)} - \bar{x}^{(i)}}{n\sigma_v^2} - \frac{1}{n} \sum_{j \in N(i)} \left(\Gamma^{(j)} \frac{\partial k(x^{(j)},\, x^{(i)})}{\partial x^{(i)}} \Gamma^{(i)} - B_{ji}^{(j)} \right). \quad (25)$$

Proof. h given by (13) can be expressed as

$$h = \frac{d}{2}\log(2\pi\sigma_v^2) + \frac{\sum_{i=1}^n (x^{(i)} - \bar{x}^{(i)})^2}{2m\sigma_v^2} + \frac{1}{2}\log(2\pi) + \frac{1}{2n}\log|K+\sigma_w^2 I| + \frac{1}{2m}\bar{\mathbf{y}}^T (K+\sigma_w^2 I)\bar{\mathbf{y}}.$$

Now take a derivative respect to $x^{(i)}$ to get

$$\frac{\partial h}{\partial x^{(i)}} = \frac{x^{(i)} - \bar{x}^{(i)}}{n\sigma_v^2} + \frac{1}{2n} tr\left(\frac{\partial K}{\partial x^{(i)}} (K+\sigma_w^2 I)^{-1} \right) + \frac{1}{2n}\bar{\mathbf{y}}^T (K+\sigma_w^2 I)^{-1} \frac{\partial K}{\partial x^{(i)}} (K+\sigma_w^2 I)^{-1}\bar{\mathbf{y}}.$$

Using (21) and (22), we get

$$\frac{\partial h}{\partial x^{(i)}} = \frac{x^{(i)} - \bar{x}^{(i)}}{n\sigma_v^2} + \frac{1}{n} tr(B_i^T) - \frac{1}{2n}\Gamma^T \frac{\partial K}{\partial x^{(i)}} \Gamma. \quad (26)$$

It is known that $\frac{\partial K}{\partial x^{(i)}}$ is a sparse matrix with only non-zero elements on the i-th row and column for the neighbors of i. Rewriting the above equation in the summation form leads to (25).

Finally using the recursive gradient method given by (27), agents can update their modes in a distributed manner, where $\bar{x}^{(i)}$ can be used as an initial condition and $\gamma \in \mathbb{R}_{>0}$ is a step size.

$$\hat{x}^{(i)}(t+1) = \hat{x}^{(i)}(t) + \gamma \frac{\partial h}{\partial x^{(i)}} \Bigg|_{\mathbf{x}=\widehat{\mathbf{x}}}. \quad (27)$$

The overall algorithm is summarized in Algorithm 7.

Algorithm 7 Distributed Algorithm for Computing $\hat{\mathbf{x}}$

```
Given Initial position $\bar{\mathbf{x}}$, corrupted measurement $\bar{y}$ and $\varepsilon$ satisfying (20).
begin $\forall_i\ \hat{x}_i^{(0)} = \bar{\mathbf{x}}_i$, $\Gamma_i^{(0)} = 0$, $B_i^{(0)} = \mathbf{0} \in \mathbb{R}^{n\times n}$
repeat
    repeat
        1. Update $\Gamma_i$ using (23).
        2. Update $B_i$ using (24).
    until $\Gamma_i$ and $B_i$ converges.
    1. Compute $\frac{\partial h}{\partial x^{(i)}}$ using (25).
    2. Update $\hat{x}_i$ using (27).
until $\hat{x}_i$ converges
output Estimated position $\hat{\mathbf{x}}$
```

6.4 Distributed Estimation of Posterior Means and Variances

Let us consider the first-order approximation of the PME estimator in (14). The elements of the covariance matrices $k(\hat{x}^{(i)}, x_\star)$ and $k(\hat{x}^{(i)}, \hat{x}^{(j)})$ can be calculated using (16) and the converged asymptotic mode $\widehat{\mathbf{x}}$ of $-h$ can be found.

Proposition 3. *If a sensor network is connected, every agent can compute $\hat{\mathbb{E}}(y_\star|D)$ with the DAC method by exchanging only local information using:*

$$\theta_i(m+1) = \theta_i(m) + \varepsilon \sum_{j \in N(i)} (\theta_j(m) - \theta_i(m)), \quad (28)$$

where $\theta_i(0) = k^{(i)}\Gamma^{(i)}$.

Proof. By Proposition 1 and (21), the PME in (14) can be represented as follows:

$$\hat{\mathbb{E}}(y_\star|D) = k^T(x_\star, \widehat{\mathbf{x}})(K(\widehat{\mathbf{x}}, \widehat{\mathbf{x}}) + \sigma_w^2 I)^{-1}\bar{\mathbf{y}} = k^T(x_\star, \widehat{\mathbf{x}})\Gamma. \quad (29)$$

The solution of (29) can be solved using DAC since (29) can be matched to (18). Recall that (29) is a scalar value and can be represented as $\sum_{i=1}^q k^T(x_\star, \widehat{\mathbf{x}}^{(i)})\Gamma^{(i)}$ and the i-th agent knows both $k^T(x_\star, \widehat{\mathbf{x}}^{(i)})$ and $\Gamma^{(i)}$.

From Section 6.2, we know that θ_i converges to $\frac{1}{q}\hat{\mathbb{E}}[y_\star|D]$ when $0 < \varepsilon < \frac{1}{\max_i \sum_{i \neq j} Q_{ij}}$ and each agent can easily compute $\hat{\mathbb{E}}[y_\star|D]$ by $q \times \theta_i$.

The predictive variance can be approximated similarly with

$$\widehat{\mathbf{var}}(y_\star|D) = \sigma_f^2 - k^T(K+\sigma_w^2 I)^{-1}k. \quad (30)$$

First, suppose that

$$\Phi = (K(\widehat{\mathbf{x}}, \widehat{\mathbf{x}}) + \sigma_w^2 I)^{-1} k(x_\star, \widehat{\mathbf{x}}). \quad (31)$$

Agent i can calculate the i-th element of (31) by the following recursion:

$$\Phi^{(i)}(k+1) = (1-\alpha)\Phi^{(i)}(k) + \frac{\alpha}{\sigma_f^2+\sigma_w^2} \times \left(\bar{y}^{(i)} - \sum_{j\in N(i)} k(\hat{x}^{(i)},\hat{x}^{(j)})\Phi^{(j)}(k) \right). \qquad (32)$$

Once JOR converges, the error variance can be computed using following recursion based on the DAC method

$$\theta_i(k+1) = \theta_i(k) + \varepsilon \sum_{j\in N(i)} (\theta_j(k) - \theta_i(k)), \qquad (33)$$

where

$$\theta_i(0) = k^T(x_\star, \widehat{\mathbf{x}}^{(i)})\Phi^{(i)}. \qquad (34)$$

Without loss of generality, the iterative DAC solution in (28) and (33) can be extended to vector formulation, where $\theta_i \in \mathbb{R}^k$.

Remark 3. *The proposed distributed posterior mean and variance estimators given by (29) and (30) are different from the naive approach (or a quick and dirty solution (QDS)) given by (6) and (7). In the naive approach, we completely neglect the effect of localization uncertainty. In the proposed distributed approach, we estimate the field based on* $\widehat{\mathbf{x}}$*, an MAP estimate of the agent's position, instead of directly using noisy location measurements.*

The distributed GPR (D-GPR) algorithm is outlined in Algorithm 8. Note that the number of agents q can be also estimated using DAC using (28) with an initial value of 1 at one predefined agent and 0 at all other agents, which will converge to $1/q$.

Algorithm 8 Distributed GPR (D-GPR) Algorithm

```
Given $\widehat{\mathbf{x}}$ and $\Gamma$ computed using Algorithm 7, corrupted
measurement $\bar{y}$, unmeasured position $x_\star$.
begin $\theta_i^{(0)} = k(x_\star, \hat{x}_i)\Gamma^{(i)}$
repeat
  1. Update $\theta_i$ using (28).
until $\theta_i$ converges
output $\hat{\mathbb{E}}[y_\star|D] = q \times \theta_i$
```

7 Hierarchical Distributed GPR Under Localization Uncertainty

In this section, we develop a hierarchical extension of the proposed distributed GPR algorithm using sparse Gaussian process regression in order to increase the scalability of the algorithm. We assume the same setup as described in Section 6 but with a larger number of agents.

The maximum a posteriori (MAP) mode estimation and a distributed GPR algorithm proposed in Section 6 can effectively handle limited computational capabilities inherent in a distributed sensor network. However, due to the relatively slow convergence speed, it is difficult to apply the proposed algorithm for a network with a large number of sensors monitoring a larger region. It is known that the convergence time $T_n(\varepsilon)$ of a distributed consensus algorithm is [31]:

$$T_n(\varepsilon) = O(n^3 \log(n/\varepsilon)), \qquad (35)$$

where n is the number of agents and ε is the error bound. Hence, for a large network, it is impractical to run a completely distributed algorithm.

In order to overcome this computational burden, we propose hierarchical distributed GPR (HD-GPR). HD-GPR is a divide-and-conquer version of the D-GPR algorithm. HD-GPR first divides a large distributed sensor network into a set of clusters considering its network topology. Then each cluster performs the proposed distributed GPR algorithm to estimate the sensory field monitored by sensors belong to the cluster. Estimates from different clusters are combined efficiently using a sparse Gaussian process method to estimate the entire field monitored by the whole network. HD-GPR is based on distributed spectral clustering and sparse GPR approximation and they are explained first.

7.1 Distributed Spectral Clustering

Spectral clustering is a method which uses eigenvalues of a similarity matrix for clustering data points. If the similarity matrix is constructed using relative distances between sensing agents, it contains information about the network topology. Hence, spectral clustering is suitable for grouping sensing agents in a sensor network.

In order to perform spectral clustering, we need to obtain first k eigenvectors from a list of eigenvectors of the similarity matrix ordered by the magnitude of its corresponding eigenvalue. While this can be easily computed in a centralized manner, it can be tricky in a distributed sensor network since every computation has to be done in a distributed manner. In [20], Kempe et al. proposed decentralized orthogonal iteration (DOI), a distributed version of orthogonal iteration for computing first k eigenvectors. Using this algorithm, each agent can calculate the eigenvector associated with the agent solely from local communication with its neighboring agents. Once all agents know their corresponding eigenvectors, we can cluster similar agents into a set of groups using eigenvectors. A distributed clustering can be achieved using the primal dual k-means algorithm [21], which is a distributed k-means algorithm for clustering observations collected by a spatially deployed sensor network. It is guaranteed to converge to a local minimum of the centralized solution [22]. Hence, we can partition sensing agents into clusters based on the network topology using the decentralized orthogonal iteration method [20] and the distributed k-means algorithm [21]. An example of distributed spectral clustering is shown in Figure 1, along with the centralized solution for comparison.

7.2 Sparse GPR Approximation

Although Gaussian process regression has been successfully used in a number of applications [6, 7], it suffers from the cubic time complexity in the number of data points for calculating the inverse of a kernel matrix in (6) and (7). To alleviate this computational burden, a number of approximate GP methods have been proposed.

A number of existing approximate GPR methods are based on the sparse subset of data (SoD) approximation method. In particular, partially independent training conditional (PITC) approximation was proposed in [11]. PITC approximates using a block diagonal kernel matrix by grouping training data into clusters. While the clustering has some implementation issues, it can improve the quality of approximation compared to fully independent training conditional (FITC) approximation by considering the conditional dependency of data within the same group.

The conditional distribution of PITC of $y_\star$ at a location of interest $x_\star$ given data D is

$$y_\star|D \sim N(\mu_\star^{PITC}(x_\star|D),\ \Sigma_\star^{PITC}(x_\star|D)) \tag{36}$$

where

$$\mu_\star^{PITC}(x_\star|D) = \Gamma(x_\star,\mathbf{x})^T(\Gamma(\mathbf{x},\mathbf{x})+\Lambda)^{-1}\mathbf{y} \tag{37}$$

and

$$\Sigma_\star^{PITC}(x_\star|D) = k_\star - \Gamma(x_\star,\mathbf{x})^T(\Gamma(\mathbf{x},\mathbf{x})+\Lambda)^{-1}\Gamma(x_\star,\mathbf{x}) \tag{38}$$

with $k_\star = k(x_\star,x_\star)$. Here, Λ is a block-diagonal matrix constructed from the K diagonal blocks of $k(\mathbf{x},\ \mathbf{x})$ and

$$\Gamma(A,\ B) = k(A,\ U)k(U,\ U)^{-1}k(U,\ B), \tag{39}$$

where U is a known support set and $k(A,B) \in \mathbb{R}^{n_A\times n_B}$ is a covariance matrix between data $A \in \mathbb{R}^{n_A}$ and $B \in \mathbb{R}^{n_B}$. A support set is a collection of input points summarizing all data points. The cardinality of a support set determines the rank of the kernel matrix and the quality of the approximation. In other words, if the support set is the same as the given input data, the result of sparse GPR approximation will be the same as original GPR.

In [12], Chen et al. proposed a decentralized data fusion algorithm (D^2FAS) that can provide the same prediction results as the PITC approximation. Let G_k be the k-th cluster of agents and K be the total number of clusters. The local summary for G_k is defined as follows.

Definition 1. *(Local Summary)*
Given a support set U known to all agents. Suppose (X_k, z_{X_k}) be the position and measurement data of the k-th cluster G_k, the local summary of G_k is defined as a tuple ($\dot{z}_U^k$, $\dot{\Sigma}_U^k$), where

$$\dot{z}_U^k \triangleq \left(k(U,X_k)\Sigma_{X_kX_k|U}^{-1}\right) z_{X_k} \tag{40}$$

$$\dot{\Sigma}_U^k \triangleq k(U,X_k)\Sigma_{X_kX_k|U}^{-1}k(X_k,U) \tag{41}$$

such that $\Sigma_{X_kX_k|U} = k(X_k,X_k) - k(X_k,U)k(U,U)^{-1}k(U,X_k)$.

By communicating local summaries with neighbors, each agent can compute the global summary defined as follows.

Definition 2. *(Global Summary)*
Given a support set U known to all agents, the global summary is defined as a tuple $(\bar{z}_U, \bar{\Sigma}_{UU})$, where

$$\bar{z}_U \triangleq \sum_{k=1}^{K} \dot{z}_U^k \tag{42}$$

$$\bar{\Sigma}_{UU} \triangleq k(U,U) + \sum_{k=1}^{K} \dot{\Sigma}_U^k \tag{43}$$

Given the global summary ($\bar{z}_U$, $\bar{\Sigma}_{UU}$), each agent can compute a globally consistent predictive distribution of $y_\star$ at a new input $x_\star$ as follows:

$$y_\star \sim N(\mu_\star^{D^2FAS},\ \Sigma_\star^{D^2FAS}) \tag{44}$$

where

$$\mu_\star^{D^2FAS} = k(x_\star,U)\bar{\Sigma}_{UU}^{-1}\bar{z}_U \tag{45}$$

and

$$\Sigma_\star^{D^2FAS} = k_\star - k(x_\star,U)(k(U,U)^{-1} - \bar{\Sigma}_{UU}^{-1})k(U,x_\star). \tag{46}$$

The proof of the equivalence between PITC and D^2FAS can be found in the appendix of [12].

Note that when it comes to computing covariances, the PITC approximation does not correspond exactly to a Gaussian

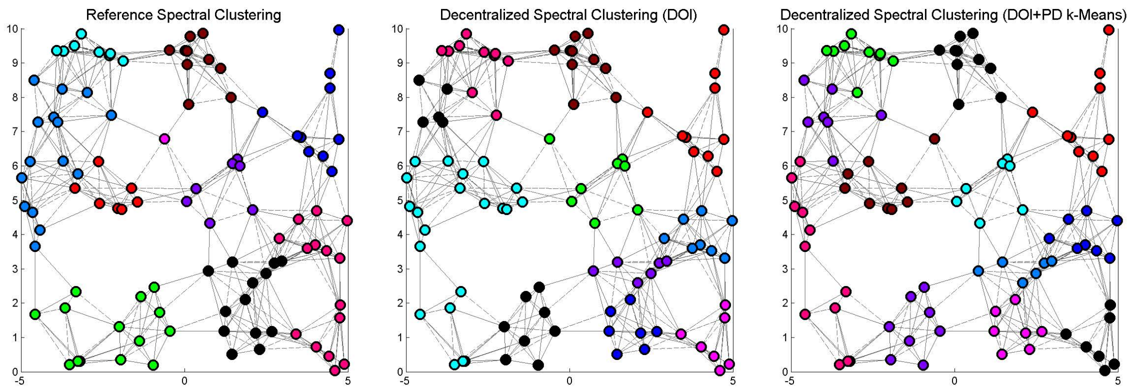


Fig. 1. Results of different spectral clustering methods. The color of each node indicates its cluster membership. (Left) A clustering result using a centralized method. (Middle) A clustering result using DOI for computing eigenvectors and centralized k-means. (Right) Fully distributed spectral clustering using DOI and primal-dual k-means.

process. Under the PITC approximation, the correlation between test and training data is represented by the inducing inputs in the support set and the covariance is minimized if the test input is close to inducing inputs in the support set. Hence, PITC can provide a good approximation to original GPR even if the covariance matrix computed for the PITC approximation is different from the covariance matrix of the original GP.

7.3 Hierarchical Distributed GPR Algorithm

In this section, we describe the hierarchical distributed GPR (HD-GPR) algorithm under localization uncertainty. We assume the sensor network is partitioned into groups or clusters using distributed spectral clustering described in Section 7.1. Once each agent knows the group or cluster it belongs, we can estimate the real positions of agents using the distributed MAP mode estimation in Section 6. Using the estimated position (mode) and the clustering information, we use distributed PITC described in Section 7.2. A local summary is estimated by each group, which can be done by any agent in the group. Local summaries are then exchanged among groups and the global summary is computed. Since all sensing agents can participate and compute local and global summaries, we only assume that at least one agent in each group is responsible for computing local and global summaries. Once the global summary is estimated, it has to propagate to the entire network using flooding if necessary. Based on the global summary, we can estimate environmental parameters of the region of interest using (45) and (46). Using distributed PITC approximation, the number of values to be merged using the consensus algorithm reduces from n to n/k for making local summaries and k for making the global summary, where n is the number of agents and k is the number of groups. Considering the time complexity of a distributed consensus algorithm [31], this divide-and-conquer method can significantly reduce the computational load and improve the scalability of the algorithm. The HD-GPR algorithm is given in Algorithm 9 and illustrated in Figure 3.

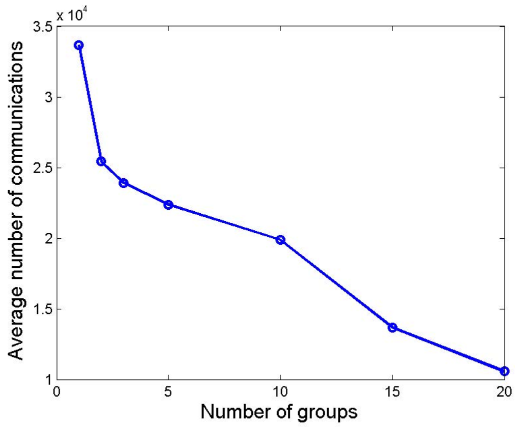
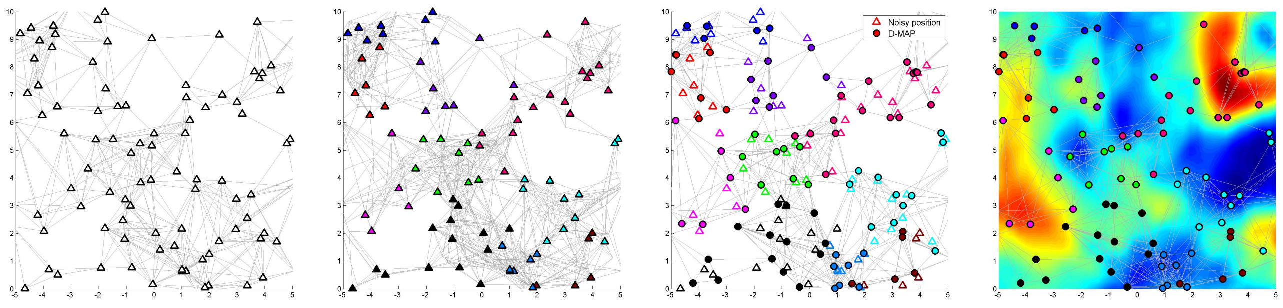


Fig. 2. An average number of communications per agent required to perform Algorithm 7.

Figure 2 shows the average number of communications of an agent per one iteration of Algorithm 7. The experimental setting is same as Section 8.2, i.e., 100 agents are deployed in the field of interest, with different numbers of groups. Considering that the number of required iterations has a cubic complexity in the number of agents, as noted in (35), our proposed algorithm can dramatically reduce the number of required iterations.

Remark 4. *Even with HD-GPR with* 20 *groups, the average number of communications exceeds* 10^4 *as shown in Figure 2. However, this is mainly because we set the threshold for stopping criterion to be sufficiently small. Moreover, when it comes to implementing the algorithm using low-performance processors, e.g., microcontroller unit (MCU), distributed algorithms have an advantage in that the only required computation is simple additions and multiplications as shown in Figure 10 (b).*



(a) (b) (c) (d)

Fig. 3. An overview of the hierarchical distributed GPR (HD-GPR) algorithm. (a) A connected sensor network. (b) Groups of sensing agents formed using distributed spectral clustering. (c) Estimated agent positions using the distributed mode estimator. (d) The field estimated by HD-GPR incorporating both position and measurement noises.

Algorithm 9 Hierarchical Distributed GPR (HD-GPR) Algorithm

```
1: Perform distributed spectral clustering (Section 7.3).
2: Perform distributed MAP estimation to find $\hat{x}_i$ using Algorithm 7.
3: Make the local summary using (40) and (41) within each group.
4: Make the global summary using (42) and (43).
5: Estimate the field using (44).
```

	Method					
	QDS		MAP-GPR		D-GPR	
trial	mean	var	mean	var	mean	var
1	25.93	3.57	**21.28**	1.60	21.73	2.67
2	59.63	8.64	**45.59**	3.68	47.54	3.16
3	33.67	10.56	**24.11**	2.62	25.41	4.18
4	64.52	25.00	42.63	1.74	**41.21**	2.44
5	45.31	5.25	**32.09**	2.32	32.80	4.03
6	29.67	5.24	24.65	2.68	**24.18**	1.66
7	40.37	6.20	**32.98**	1.65	33.03	2.72
8	29.44	5.00	25.62	2.63	**25.15**	2.30
9	41.31	12.14	24.09	1.55	**23.88**	2.12
10	56.96	7.03	**42.60**	2.38	42.85	2.36
Average	42.59		**31.56**		31.78	

Table 1. Mean and variance of reconstruction errors of three algorithms: QDS (quick and dirty solution), MAP-GPR (centralized solution), and D-GPR (proposed approach).

8 Simulation Results

8.1 Distributed Gaussian Process Regression (D-GPR)

In this section, we perform a number of numerical simulations to validate the prediction performance of the proposed distributed GPR algorithm. For simulation, we have randomly generated ten reference fields from a Gaussian process with a covariance function given in (16). We then compare the predicted fields using three different algorithms against the reference field. Three algorithms used in simulations are the quick and dirty solution (QDS) given in (6), which ignores localization uncertainty, the centralized solution using the simple Laplace approximation using the MAP estimator (MAP-GPR) in (14), and the proposed distributed Gaussian process regression (D-GPR).

We assume that there are twenty sensing agents ($q = 20$). For each reference field, sensing agents are randomly located and each agent only knows a noisy position of itself. The variance of the position uncertainty is set to $\sigma_v^2 = 1$. Each agent then makes a noisy measurement from the reference field. The collection of all measurements by all agents is denoted by $D = \{\bar{\mathbf{x}}, \bar{\mathbf{y}}\}$. The hyperparameter σ_f^2 of the kernel function in (16) and the variance of the measurement noise σ_w^2 are estimated by maximizing the likelihood given $D = \{\bar{\mathbf{x}}, \bar{\mathbf{y}}\}$. Then we have performed the reconstruction of the entire field from D using three algorithms. For more accurate validation, we have repeated the above procedure ten times for each reference field, resulting ten independent runs for each reference field. The mean and variance of the root mean squared (RMS) errors between the predicted field and the reference field are shown in Figure 4. The RMS error or the reconstruction error is computed as follows:

$$E_{recon} = \|D_{ref} - D_{recon}\|_F \tag{47}$$

where $\|\cdot\|_F$ is the Frobenius norm and D_{ref} and D_{recon} are the matrices representing reference and reconstructed fields, respectively.

As expected, the centralized algorithm MAP-GPR shows the best performance in terms of the reconstruction error, followed by D-GPR and QDS. However, the mean of the reconstruction error of D-GPR is comparable to that of MAP-GPR, demonstrating the performance of the proposed distributed algorithm. One example of the reference field and the predicted fields using three algorithms are shown in Figure 5.

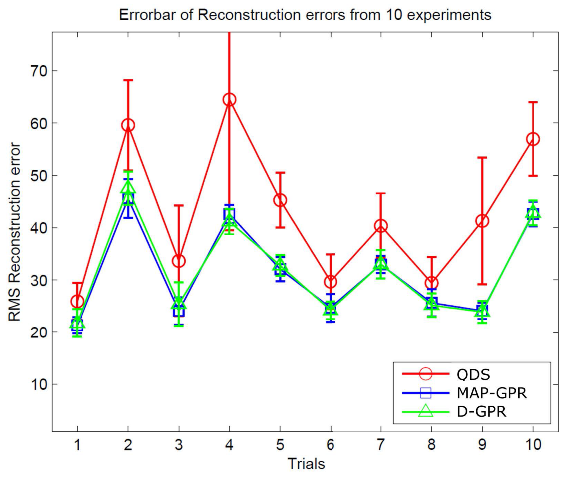


Fig. 4. Reconstruction errors of three algorithms (QDS, MAP-GPR, and D-GPR) for ten different scenarios. Error bars indicate one standard deviation from ten independent runs for each scenario. For each run, 20 agents are deployed in the field.

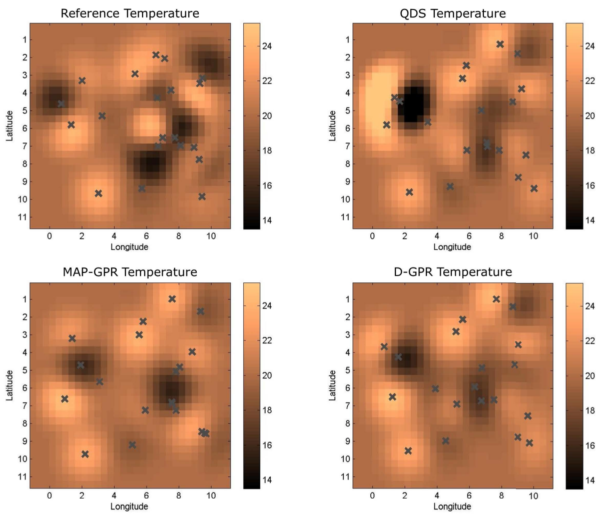


Fig. 5. An example of a reference field and fields reconstructed by three algorithms (QDS, MAP-GPR, and D-GPR). The reference field is shown in the upper left corner and, clockwise from the top, fields reconstructed using QDS, MAP-GPR, and D-GPR. The crosses for the reference field and the reconstructed field using QDS represent true positions and noisy positions, respectively. For the field estimated by MAP-GPR and D-GPR, gray crosses represent the MAP estimates of sensor positions.

Figure 6 and Figure 7 demonstrate the convergence of JOR and DAC used in our algorithm.

Since our objective function (13) is non-convex, we used a gradient-based nonlinear optimization method for both MAP-GPR and D-GPR and this can explain the difference between the reconstruction errors of these two algorithms as shown in Figure 4. However, as shown in Table 1, D-GPR has outperformed QDS in terms of the reconstruction error and its performance is comparable to MAP-GPR in most cases. Additionally, it is worth noting that the variance of MAP-GPR and D-GPR is clearly smaller than that of QDS. This indicates that the prediction performance of both solutions are more robust against diverse situations.

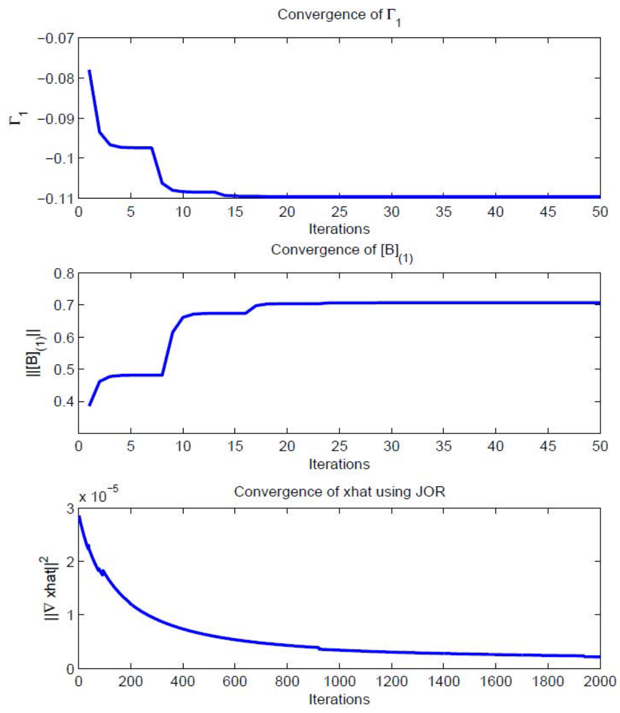


Fig. 6. Convergence of parameters using JOR. The upper and middle figure indicate Γ_1 and $[B]_{(1)}$ of agent 1, respectively. The bottom figure shows the norm of the gradient of $\widehat{\mathbf{x}}$, the solution of the MAP estimator in (12).

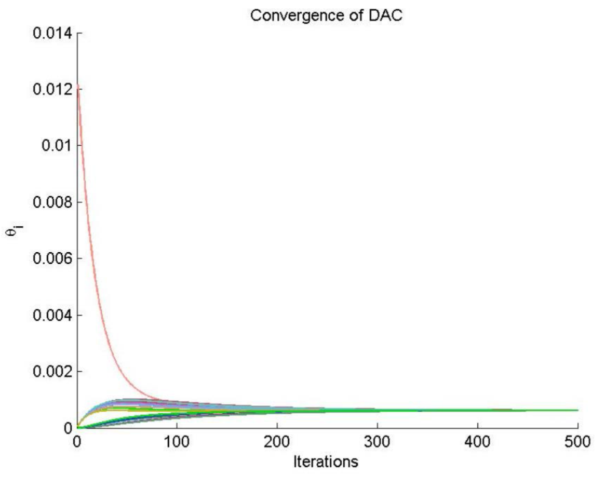


Fig. 7. Convergence of the DAC method. With an increasing number of iterations, θ_i from Algorithm 8 of all agents converges. The value of each agent is represented by a different color.

8.2 Hierarchical Distributed Gaussian Process Regression (HD-GPR)

In this section, we perform a number of numerical simulations to validate the prediction performance of the proposed HD-GPR algorithm. The experimental setting is same as Section 8.1 but 100 agents are deployed in the field. The support set U for PITC is uniformly arranged in the field and the number of supports is 100. Exemplar results using different types of Gaussian process regression methods are shown in Figure 8. From the figure, we can see that full-GP and PITC give almost the same results.

We first evaluate the prediction error while changing the number of groups and the result is shown in Figure 9(a). In particular, the prediction performance with 10 groups shows the best performance. It can be interpreted as follows. If the number of groups is too big, the distributed mode MAP estimator will not work properly since the number of agents in a group is not sufficient for good estimation. On the other hand, if the number of groups is too small, due to the high multimodal property of the objective function and relatively slow convergence speed, the estimated mode is likely to give poor results. The average reconstruction errors from HD-GPR and QDS of ten scenarios (over ten independent runs) are shown in Figure 9(b). In most cases, HD-GPR outperforms the quick-and-dirty solution.

Figure 10(a) illustrates the average reconstruction errors of different algorithms, QDS, MAP-GPR, D-GPR, and HD-GPR. In most cases, the centralized method, MAP-GPR, shows the best performance followed by two proposed methods, D-GPR and HD-GPR, which show similar reconstruction errors. Note that we relaxed the regularity conditions for finding the asymptotic mode for MAP-GPR and fixed the number of maximum iterations for finding the mode of the posterior distribution, since the required computation was too heavy. The average computation times of different algorithms are shown in Figure 10(b). As expected, D-GPR and HD-GPR require less computation time compared to MAP-GPR and HD-GPR requires less time than D-GPR.

Remark 5. *As shown in Figure 4, when there are 20 agents, there was virtually no difference between the centralized algorithm and the distributed algorithm. However, with 100 agents, Figure 10 shows that there is a performance gap between the centralized algorithm (MAP-GPR) and distributed algorithms (D-GPR and HD-GPR). On the other hand, a distributed algorithm requires a fraction of computation time compared to the centralized algorithm as shown in Figure 10(b).*

9 Conclusion

In this paper, we have proposed a distributed Gaussian process regression algorithm for resource-constrained distributed sensor networks under localization uncertainty and extended it hierarchically for scalability. Most of the existing regression methods using Gaussian process regression do not consider the localization uncertainty into its probabilistic framework. The proposed distributed algorithm can effectively handle localization uncertainty and reduce the computational load for a resource-constrained distributed sensor network. The performance of the proposed schemes are verified in numerical simulations against the quick-and-dirty solution, which is often used in practice. We have shown that the proposed algorithms outperform the quick-and-dirty solution and achieves an accuracy comparable to the centralized solution.

Acknowledgements

This research was supported by a grant to Bio-Mimetic Robot Research Center funded by Defense Acquisition Program Administration (UD130070ID) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2009348).

References

[1] Culler, D., Estrin, D., and Srivastava, M., 2004. "Guest editors' introduction: overview of sensor networks". *Computer*, pp. 41–49.

[2] Estrin, D., Culler, D., Pister, K., and Sukhatme, G., 2002. "Connecting the physical world with pervasive networks". *IEEE Pervasive Computing,* **1**(1), pp. 59–69.

[3] Lynch, K., Schwartz, I., Yang, P., and Freeman, R., 2008. "Decentralized environmental modeling by mobile sensor networks". *IEEE Transactions on Robotics,* **24**(3), pp. 710–724.

[4] Choi, J., Oh, S., and Horowitz, R., 2009. "Distributed learning and cooperative control for multi-agent systems". *Automatica,* **45**(12), pp. 2802–2814.

[5] Gu, D., and Hu, H., 2012. "Spatial gaussian process regression with mobile sensor networks". *IEEE Transactions on Neural Networks and Learning Systems,* **23**(8), pp. 1279–1290.

[6] Rasmussen, C., and Williams, C., 2006. *Gaussian processes for machine learning*, Vol. 1. MIT press Cambridge, MA.

[7] Choi, J., Lee, J., and Oh, S., 2008. "Swarm intelligence for achieving the global maximum using spatio-temporal Gaussian processes". In Proc. of the American Control Conference, pp. 135–140.

[8] Xu, Y., Choi, J., and Oh, S., 2011. "Mobile sensor network navigation using Gaussian processes with truncated observations". *IEEE Transactions on Robotics,* **27**(6), pp. 1118–1131.

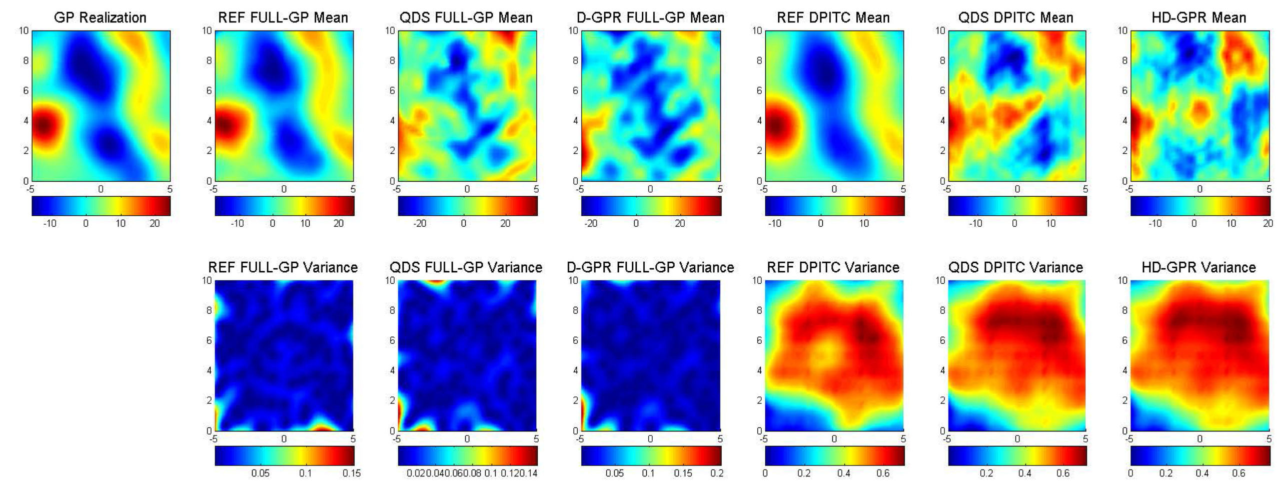
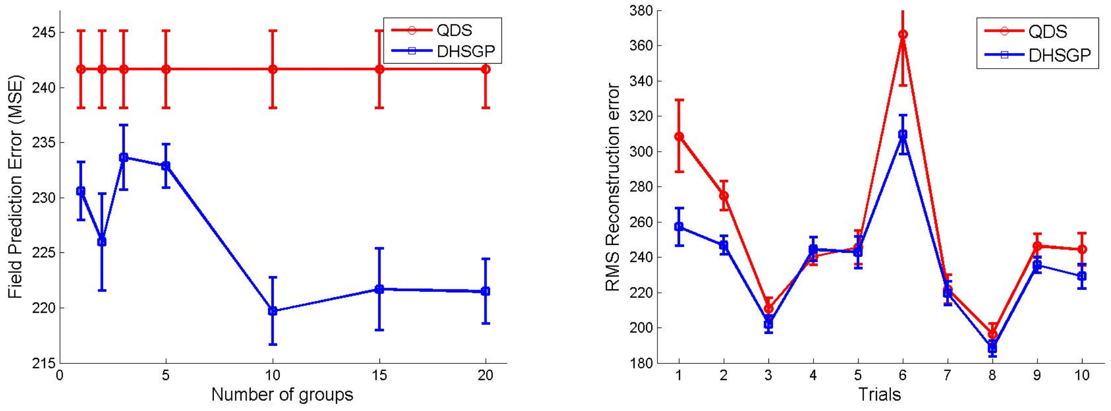
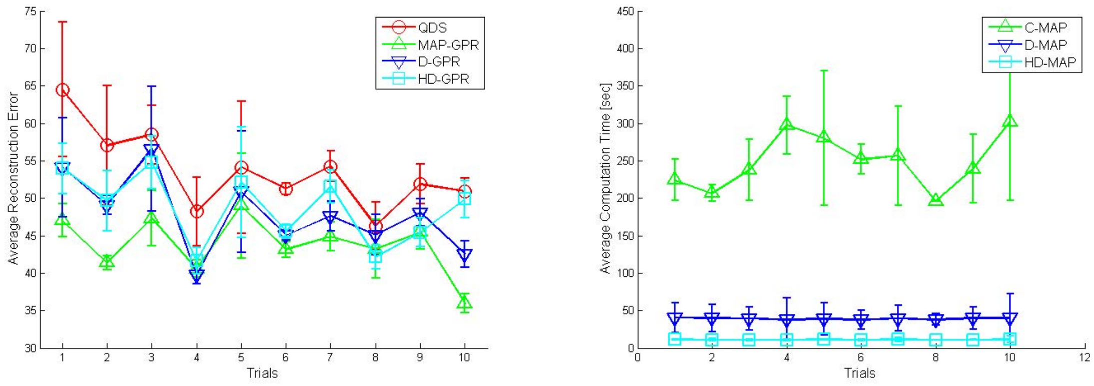


Fig. 8. Results of different Gaussian process regression methods. From left to right, we have the reference field, the predicted field using a centralized Gaussian process regression (full-GPR) with exact locations, full-GP with noisy locations, D-GPR with noisy locations, PITC with exact locations, PITC with noisy locations, and HD-GPR with noisy locations. Full-GPR and PITC indicate original Gaussian process regression and partially independent training conditional (PITC) approximation of GPR, respectively. Excluding the first column, the first row indicates the predicted mean and the second row indicates the predicted variance of each algorithm.



(a) (b)

Fig. 9. (a) Average reconstruction error as a function of the number of groups. (b) Average reconstruction errors of ten scenarios.



(a) (b)

Fig. 10. (a) Average reconstruction errors of ten scenarios. Sensory fields are fixed in each scenario. (b) Average computation time per agent.

[9] Williams, C., and Seeger, M., 2001. "Using the Nystrom method to speed up kernel machines". In Proc. of the Advances in Neural Information Processing Systems.

[10] Lawrence, N., Seeger, M., and Herbrich, R., 2002. "Fast sparse Gaussian process methods: The informative vector machine". In Proc. of the Advances in Neural Information Processing Systems.

[11] Quiñonero-Candela, J., and Rasmussen, C. E., 2005. "A unifying view of sparse approximate Gaussian process regression". *The Journal of Machine Learning Research,* **6**, pp. 1939–1959.

[12] Chen, J., Low, K. H., Tan, C. K.-Y., Oran, A., Jaillet, P., Dolan, J. M., and Sukhatme, G. S., 2012. "Decentralized data fusion and active sensing with mobile sensors for modeling and predicting spatiotemporal traffic phenomena". *arXiv preprint arXiv:1206.6230.*

[13] Oguz-Ekim, P., Gomes, J., Xavier, J., and Oliveira, P., 2011. "Robust localization of nodes and time-recursive tracking in sensor networks using noisy range measurements". *IEEE Transactions on Signal Processing,* **59**(8), pp. 3930–3942.

[14] Karlsson, R., and Gustafsson, F., 2006. "Bayesian surface and underwater navigation". *IEEE Transactions on Signal Processing,* **54**(11), pp. 4204–4213.

[15] Mysorewala, M., Popa, D., and Lewis, F., 2009. "Multi-scale adaptive sampling with mobile agents for mapping of forest fires". *Journal of Intelligent and Robotic Systems,* **54**(4), pp. 535–565.

[16] Jadaliha, M., Xu, Y., Choi, J., Johnson, N., and Li, W., 2013. "Gaussian process regression for sensor networks under localization uncertainty". *IEEE Transactions on Signal Processing,* **61**(2), pp. 223–237.

[17] Choi, S., Jadaliha, M., Choi, J., and Oh, S., 2013. "Distributed Gaussian process regression for mobile sensor network under localization uncertainty". In Proc. of the IEEE Conference on Decision and Control.

[18] Smola, A., and Bartlett, P., 2001. "Sparse greedy Gaussian process regression". In Proc. of the Advances in Neural Information Processing Systems.

[19] Ng, A. Y., Jordan, M. I., and Weiss, Y., 2001. "On spectral clustering: Analysis and an algorithm". In Proc. of the Advances in Neural Information Processing Systems, MIT Press, pp. 849–856.

[20] Kempe, D., and McSherry, F., 2008. "A decentralized algorithm for spectral analysis". *Journal of Computer and System Sciences,* **74**(1), pp. 70–83.

[21] Forero, P. A., Cano, A., and Giannakis, G. B., 2008. "Consensus-based k-means algorithm for distributed learning using wireless sensor networks". In Proc. Workshop on Sensors, Signal and Info. Process., Sedona, AZ.

[22] Forero, P. A., Cano, A., and Giannakis, G. B., 2010. "Convergence analysis of consensus-based distributed clustering". In IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), IEEE, pp. 1890–1893.

[23] Tierney, L., and Kadane, J., 1986. "Accurate approximations for posterior moments and marginal densities". *Journal of the American Statistical Association,* **81**(393), pp. 82–86.

[24] Tierney, L., Kass, R., and Kadane, J. B., 1989. "Fully exponential Laplace approximations to expectations and variances of nonpositive functions". *Journal of the American Statistical Association,* **84**(407).

[25] Miyata, Y., 2004. "Fully exponential Laplace approximations using asymptotic modes". *Journal of the American Statistical Association,* **99**(468), pp. 1037–1049.

[26] Miyata, Y., 2010. "Laplace approximations to means and variances with asymptotic modes". *Journal of Statistical Planning and Inference,* **140**(2), pp. 382–392.

[27] Bertsekas, D., and Tsitsiklis, J., 1989. *Parallel and distributed computation: Numerical Methods.* Prentice-Hall, Englewood Cliffs, NJ.

[28] Udwadia, F., 1992. "Some convergence results related to the JOR iterative method for symmetric, positive-definite matrices". *Applied Mathematics and Computation,* **47**(1), pp. 37–45.

[29] Cortés, J., 2009. "Distributed kriged Kalman filter for spatial estimation". *IEEE Transactions on Automatic Control,* **54**(12), pp. 2816–2827.

[30] Olfati-Saber, R., Fax, J. A., and Murray, R. M., 2007. "Consensus and cooperation in networked multi-agent systems". In Proc. of the IEEE, Vol. 95, pp. 215–233.

[31] Olshevsky, A., and Tsitsiklis, J. N., 2009. "Convergence speed in distributed consensus and averaging". *SIAM Journal on Control and Optimization,* **48**(1), pp. 33–55.

A Appendix A: Regularity Conditions

In this section, we review a set of regularity conditions for the Laplace approximation [24] as well as simple Laplace approximation proposed in Section 5. Let B_δ denote the open ball of radius δ centered at $\hat{\mathbf{x}}$, i.e., $B_\delta(\hat{\mathbf{x}}_a) = \{\mathbf{x} \in \mathcal{X} : ||\mathbf{x} - \hat{\mathbf{x}}_a|| < \delta\}$. Let $\hat{\mathbf{x}}_a$ be the asymptotic mode of order n^{-1}. The followings are regularity conditions.

1. $h(\mathbf{x}) \in C^6$
2. $\int_{\mathcal{X}} e^{-nh(\mathbf{x})} d\mathbf{x} < \infty$
3. $||\partial^c h(\mathbf{x})/\partial \mathbf{x}_{j_1} \cdots \mathbf{x}_{j_c}|| < M \in \mathbb{R}_{>0}$, for all $\mathbf{x} \in B_\varepsilon(\hat{\mathbf{x}}_a)$ and all $1 \leq j_1, \cdots, j_c \leq m$ with $c = 1, \cdots, 6$, where $\mathbf{x} \in \mathbb{R}^m$.
4. $\nabla^2 h(\hat{\mathbf{x}}_a)$ is positive definite and $|\nabla^2 h(\hat{\mathbf{x}}_a)| > \xi \in \mathbb{R}_{>0}$.

5. $B_\delta(\hat{\mathbf{x}}_a) \subseteq \mathcal{X}$ and
$\frac{|n\nabla^2 h(\hat{\mathbf{x}}_a)|^{\frac{1}{2}}}{C(\hat{\mathbf{x}}_a)} \int_{\mathcal{X}-B_\delta(\hat{\mathbf{x}}_a)} e^{-n[h(\mathbf{x})-h(\hat{\mathbf{x}}_a)]} d\mathbf{x} = O(n^{-2})$,

for all δ for which $0 < \delta < \varepsilon$.
where $C(\hat{\mathbf{x}}_a) = e^{\left(\frac{n}{2}\nabla h(\hat{\mathbf{x}}_a)(\nabla^2 h(\hat{\mathbf{x}}_a))^{-1}\nabla(\hat{\mathbf{x}}_a)^T\right)}$.