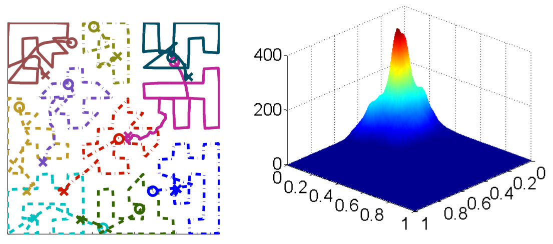
Robust Adaptive Coverage Control for Robotic Sensor Networks

Mac Schwager, *Member, IEEE*, Michael P. Vitus, *Member, IEEE*, Samantha Powers, Daniela Rus, *Fellow, IEEE*, and Claire J. Tomlin, *Fellow, IEEE*

***Abstract*—This paper presents a distributed control algorithm to drive a group of robots to spread out over an environment and provide adaptive sensor coverage of that environment. The robots use an on-line learning mechanism to estimate the areas in the environment that require more concentrated sensor coverage, while simultaneously providing this coverage. The algorithm differs from previous approaches in that both provable robustness is emphasized in the learning mechanism, and decentralization over a communication network is emphasized in the control. To achieve a provable bound on the learning error the robots explore the environment to gather sufficient data. They then switch to a Voronoi-based coverage controller to position themselves for sensing. Robots coordinate their learning, control, and mode switching in a fully distributed fashion over their mesh network. It is proved that the robots approximate the weighting function with a known bounded error, and that they converge to locations that are locally optimal for sensing. Simulations and multiple experiments with six iRobot Create robots are presented to illustrate the performance of the method.**

I. Introduction

In this paper, we present a distributed control algorithm to drive a group of robots to explore an unknown environment, while providing adaptive sensor coverage of interesting areas within the environment. This algorithm can drive a teams of robots to perform a range of sensing tasks, such as search and rescue missions, environmental monitoring, or automatic surveillance of rooms, buildings, or towns. Our main emphasis in this work over previous works is in providing provable robustness in learning the areas of importance in the environment. We also carefully consider the details of decentralizing the algorithm over a wireless mesh network. Specifically, we consider a function, which we call the weighting function, to capture the fact that some areas of the environment are more important than others. Our algorithm drives the robots to explore the environment, thereby collecting the necessary information to learn the weighting function with provably bounded maximum error, and finally to move to positions to provide locally optimal sensor coverage over the environment, as illustrated in Fig. 1. The coordination between the robots



(a) Robot Trajectories (b) Estimated Weighting Function

Fig. 1. As shown by the robot trajectories in 1(a), our coverage algorithm drives the robots to fully explore the environment. Different robots are indicated by different line styles and colors. This leads to a provably robust reconstruction of the weighting function, as shown in 1(b). The robots then switch asynchronously into a coverage mode to provide provable sensor coverage over the environment.

is carried out in a decentralized way over their network. We provide experimental results with six iRobot Creates as well as simulated results using experimentally collected data to demonstrate the algorithm.

This work is motivated by the need in many applications to have a provable bound on the estimation quality of the weighting function, giving a guarantee on the quality of the coverage provided by the sensing robots. As an example, consider the earthquake and consequent tsunami that struck the northern coast of Japan in March 2011, causing serious damage to the nuclear reactors at Fukushima. Following the disaster, human workers were critically limited in their ability to assess the radiation levels in the area, and braved the risk of dangerous radiation exposure as a result. In this example, the unknown weighting function represents the radiation levels in the area. Had the technology been available, a team of robots could have been deployed to monitor the levels of radiation around the reactor. Using the algorithm proposed in this paper, sensing robots would learn the radiation levels, and then concentrate on areas where the radiation was most dangerous. Most importantly, a rigorous bound on the precision of the radiation estimate could be established *a priori*. This information could be used to direct human workers, ensuring that they remain safe from the danger of radiation exposure. If existing coverage algorithms were used, without the robustification proposed in this paper, the radiation levels could be approximated arbitrarily poorly, with large errors in regions not visited by the robots. In pathological cases, the approximation could even become unstable, providing unbounded radiation estimates, and leading to saturated control inputs for the robots. Our robust adaptive coverage algorithm prevents these effects and leads the robots to converge to

This work was funded in part by ONR MURI Grants N00014-07-1-0829, N00014-09-1-1051, and N00014-09-1-1031, and the SMART Future Mobility project. We are grateful for this financial support.

M. Schwager is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA, schwager@stanford.edu.

M. P. Vitus and C. J. Tomlin are with the Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, CA 94708, USA, {vitus,tomlin}@eecs.berkeley.edu.

S. Powers and D. Rus are with the Computer Science and Artificial Intelligence Lab, MIT, Cambridge, MA 02139, USA, powerss@alum.mit.edu, rus@csail.mit.edu.

locally optimal positions, and to approximate the weighting function with a provable maximum error.

Sensor coverage algorithms have received a great deal of attention in recent years. Cortés et al. [1] proposed a distributed control algorithm for finding a locally optimal sensing configuration for a group of sensing robots, adapting concepts from locational optimization [2], [3]. There have been several extensions to this formulation of coverage control. The paper [4] used a distributed interpolation scheme to recursively estimate the weighting function so that it did not have to be known *a priori*. In [5], the robots were assumed to have a limited sensing or communication range. Pimenta et al. [6] incorporated heterogeneous robots, and extended the algorithm to handle nonconvex environments. Other extensions to nonconvex environments were proposed in [7], [8]. Similar frameworks have also been formulated for a stochastic setting [9]–[11]. Most relevant to our work, [12] removed the requirement of knowing the weighting function *a priori* by learning a basis function approximation of the weighting function online. This strategy has provable convergence properties, but requires that the weighting function lie in a known set of functions (the set spanned by the basis functions). If the function does not lie in this known set, the algorithm from [12] can lead to a divergent weighting function approximation and saturated control inputs for the robots; or, conversely, it may cause the robots to remain in a local neighborhood of their initial positions resulting in a poor global approximation of the weighting function. The purpose of the present work is to guarantee a high-quality estimate of the weighting function, and a locally optimal coverage configuration for the robots, regardless of the weighting function.

The algorithm we propose here drives the robots to explore the entire space, which enables them to learn the weighting function with provable robustness. Specifically, the contributions of this paper are as follows:

1) We propose a new distributed motion control algorithm in which the robots proceed asynchronously through three different modes (`partition`, `explore`, and `cover`), by coordinating over their communication network.
2) We introduce an online distributed function approximation algorithm with a dead-zone based robustification mechanism that the robots use to learn the weighting function.
3) We prove that the combination of the motion control algorithm and robust function approximate algorithm lead to (a) the robots learn the weighting function with bounded maximum error, and (b) they approach final positions that are locally optimal for coverage.
4) We show results of hardware experiments with the algorithm running on six iRobot Creates, as well as simulation results with experimentally collected sensor data.

The robots first partition the environment in `partition` mode using a distributed K-means clustering algorithm. They then asynchronously switch to perform a distributed exploration of the environment using an approximate Traveling Salesperson Problem (TSP) solver in `explore` mode. Finally, they asynchronously switch to a distributed `cover` mode in which the robots move to a centroidal Voronoi configuration, which is locally optimal for sensing. The switching between these modes is accomplished with a network algorithm that ensures that the last robot finishes one mode before the first robot starts the next. Concurrently with these `partition`, `explore`, and `cover` modes, the robots use an on-line learning algorithm to approximate the weighting function from sensor measurements. Since we do not assume the robots can perfectly approximate the weighting function, the parameter adaptation law for learning this weighting function uses a dead-zone modification to be robust to function approximation errors.

In real-world applications the form of the weighting function is not known *a priori*, and if the learning algorithm does not have an explicit robustness mechanism, then it may chatter between models or even diverge [13]. Several techniques have been proposed in the adaptive control literature to handle this kind of robustness, including the dead-zone modification [14], [15], the σ-modification [13], the e_1-modification [16], and the projection modification [17]. We adapt a method inspired by a dead-zone technique from adaptive control. We prove that this modification, together with the exploration phase executed by the robots, results in all robots learning a function that has bounded maximum point-wise error from the true function. Furthermore, all robots' learned functions approach a common function, and the robots converge to positions that are locally optimal for sensing with respect to the learned function.

There are a number of other notions of multi-robot sensor coverage different from the formulation in this paper (e.g. [18], [19] and [20], [21]). Another influential method that uses a basis function-based approximation scheme and distributed Kalman filtering is described in [22]. Other works have considered the related problem of finding peaks, or sources, in unknown scalar fields with a group of robots [23]–[25]. Some researchers have also used the gradient of mutual information as an objective for controlling robots to build estimates of environmental fields, as in [26]–[28] and to avoid hazardous regions while building a field estimate, as in [29]. In this paper we adopt the locational optimization approach to coverage control for its interesting possibilities for analysis and its compatibility with existing ideas in adaptive control [30]–[32]. We also utilize Gaussian networks for function approximation, which is a well-known technique that has been used in many other contexts, for example [33].

This paper includes an updated and expanded version of much of the material from [34], as well as new theoretical, experimental, and simulation results. In Section II we introduce notation and formulate the problem. In Section III we describe the function approximation strategy and the control algorithm, and we prove the main convergence result. Section IV gives the results of a numerical simulation with a weighting function that was determined from empirical measurements of light intensity in a room. Section V describes the hardware implementation on six iRobot Creates, and gives results from a total of 14 experimental runs in two different experimental configurations. Finally, conclusions and future work are discussed in Section VI.

II. Problem Formulation

In this section we build a model of the robotic sensing network, the environment, and the weighting function defining areas of importance in the environment. Much of this material is background from existing work, for example [12]. We then formulate the robust adaptive coverage problem with respect to this model. The main objective is to design a distributed control and function approximation algorithm by which the robots can (i) explore the environment, (ii) build an approximation of the weighting function over the environment with provable error bounds, and (iii) drive the robots to a position to provide sensor coverage of the environment, as illustrated in Fig. 1.

Let there be n robots with positions $p_i(t)$ in a planar environment $Q \subset \mathbb{R}^2$. The environment is assumed to be compact and convex. We call the tuple of all robot positions $P = (p_1, \dots, p_n) \in Q^n$ the configuration of the multi-robot system, and we assume that the robots move with integrator dynamics

$$\dot{p}_i = u_i, \tag{1}$$

so that we can control their velocities directly through the control input u_i. We define the Voronoi partition of the environment to be $V(P) = \{V_1(P), \dots, V_n(P)\}$, where

$$V_i(P) = \{q \in Q \mid \|q - p_i\| \le \|q - p_j\|, \forall j \neq i\},$$

and $\|\cdot\|$ is the ℓ^2-norm. We think of each robot i as being responsible for sensing in its associated Voronoi cell V_i. Next, we define the communication network as an undirected graph, in which all robots whose Voronoi cells touch share an edge in the graph. This graph is known as the Delaunay graph. Then the set of neighbors of robot i is defined as $\mathcal{N}_i := \{j \mid V_i \cup V_j \neq \emptyset\}$.

We now define a weighting function over the environment $\phi : Q \mapsto \mathbb{R}_{>0}$ (where $\mathbb{R}_{>0}$ denotes the strictly positive real numbers). This weighting function is *not known* by the robots. Intuitively, we want a high density of robots in areas where $\phi(q)$ is large and a lower density where it is small. The weighting function represents regions of importance or interest. Examples may be radiation concentration levels, temperature, concentration of a chemical contaminant, or the density of a quantity of interest such as population density, or traffic density. Finally, suppose that the robots have sensors with which they can measure the value of the weighting function at their own position, $\phi(p_i)$ with high precision (i.e. without noise), but that their quality of sensing at arbitrary points, $\phi(q)$, degrades quadratically in the distance between q and p_i. That is to say the cost of a robot at p_i sensing a point at q is given by $\frac{1}{2}\|q - p_i\|^2$. This can be extended to non-quadratic sensing models as described in [5], but here we assume the quadratic model for notational simplicity. Since each robot is responsible for sensing in its own Voronoi cell, the cost of all robots sensing over all points in the environment is given by

$$\mathcal{H}(P) = \sum_{i=1}^{n} \int_{V_i(P)} \frac{1}{2}\|q - p_i\|^2 \phi(q)\, dq. \tag{2}$$

This is the overall objective function that we would like to minimize by controlling the configuration of the multi-robot system.[1]

The gradient of $\mathcal{H}$ can be shown[2] to be given by

$$\frac{\partial \mathcal{H}}{\partial p_i} = -\int_{V_i(P)} (q - p_i)\phi(q)\, dq = -M_i(P)(C_i(P) - p_i), \tag{3}$$

where we define $M_i(P) := \int_{V_i(P)} \phi(q)\, dq$ and $C_i(P) := 1/M_i(P) \int_{V_i(P)} q\phi(q)\, dq$. We call M_i the mass of the Voronoi cell i and C_i its centroid, and for efficiency of notation we will henceforth write these without the dependence on P. We would like to control the robots to move to their Voronoi centroids, $p_i = C_i$ for all i, since from (3), this is a critical point of $\mathcal{H}$, and if we reach such a configuration using gradient descent, we know it will be a local minimum. Global optimization of $\mathcal{H}$ is known to be computationally intractable (NP-hard for a given discretization of the environment), hence it is standard in the literature to only consider local optimality.

A. Approximate Weighting Function

Note that the cost function (2) and its gradient (3) rely on the weighting function $\phi(q)$, which is not known to the robots. In this paper we provide a means by which the robots can robustly approximate $\phi(q)$ online in a distributed way and move to decrease (2) with respect to this approximate $\phi(q)$.

To be more precise, each robot maintains a separate approximation of the weighting function, which we denote $\hat{\phi}_i(q, t)$. These approximate weighting functions are generated from a linear combination of m static basis functions, $\mathcal{K}(q) = [\mathcal{K}_1(q) \cdots \mathcal{K}_m(q)]^{\mathrm{T}}$, where each basis function is a radially symmetric Gaussian of the form

$$\mathcal{K}_j(q) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{\|q - \mu_j\|^2}{2\sigma^2}\right\}, \tag{4}$$

with fixed width σ and fixed center μ_j. Furthermore, the centers are arranged in a regular grid over Q. Each robot then forms its approximate weighting function as a weighted sum of these basis functions $\hat{\phi}_i(q, t) = \mathcal{K}(q)^{\mathrm{T}} \hat{a}_i(t)$, where $\hat{a}_i(t)$ is the parameter vector of robot i. This function approximation scheme is illustrated in Fig. 2.[3] Each element in the parameter vector is constrained to lie within some lower and upper bounds $0 < a_{\min} < a_{\max} < \infty$ so that $\hat{a}_i(t) \in [a_{\min}, a_{\max}]^m$. These bounds are set based on standard design considerations so as to prevent memory overflow and control actuator saturation. This does not limit the class of functions $\phi(q)$ that can be approximated, it only increases the minimum approximation error. Robot i's approximation of its Voronoi cell mass and

[1]We have pursued an intuitive development of this cost function, though more rigorous arguments can also be made [35]. This function is known in several fields of study including the placement of retail facilities [3] and data compression [36].

[2]The computation of this gradient is more complex than it may seem, because the Voronoi cells $V_i(P)$ depend on P, which results in extra integral terms. Fortunately, these extra terms all sum to zero, as shown in, e.g. [37].

[3]This is a well-known form of universal function approximator. There are several delicate choices that must be made in designing such a function approximation network, such as values for the Gaussian width σ and the number and density of basis functions to use. Issues such as these have been addressed in many previous works, for example [33].

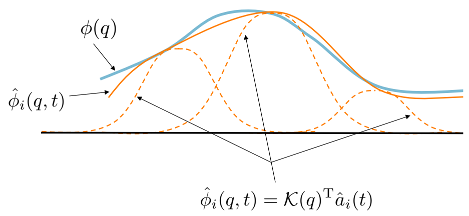


Fig. 2. The weighting function approximation is illustrated in this simplified 2-D schematic. The true weighting function $\phi(q)$ is approximated by robot i to be $\hat{\phi}_i(q,t)$. The basis function vector $\mathcal{K}(q)$ is shown as three Gaussians (dashed curves), and the parameter vector $\hat{a}_i(t)$ denotes the weighting of each Gaussian.

centroid can then be defined as $\hat{M}_i(P,t) := \int_{V_i(P)} \hat{\phi}_i(q,t)\, dq$ and $\hat{C}_i(P,t) := 1/\hat{M}_i(P,t) \int_{V_i(P)} q\hat{\phi}_i(q,t)\, dq$, respectively. Again, we will drop the dependence of $\hat{M}_i$ and $\hat{C}_i$ on (P,t) for notational simplicity.

We measure the difference between an approximate weighting function and the true weighting function as the L^∞ function norm of their difference, so that the best approximation is given by

$$a := \underset{\hat{a}\in[a_{\min},a_{\max}]^m}{\arg\min} \max_{q\in Q} |\mathcal{K}(q)^{\mathrm{T}}\hat{a} - \phi(q)|, \tag{5}$$

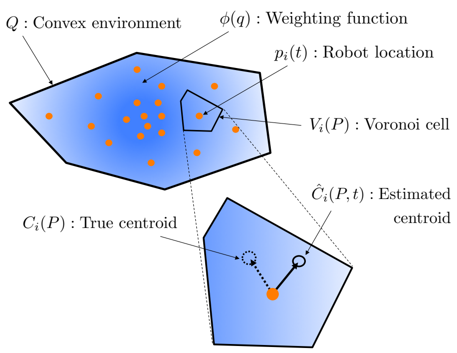
and the optimal function approximation error is given by

$$\phi_\epsilon(q) := \mathcal{K}(q)^{\mathrm{T}}a - \phi(q). \tag{6}$$

It will be shown in the proof of the main theorem, Theorem 1, that the L^∞ norm gives the tightest approximation bound with our proof technique. The only restriction that we put on $\phi(q)$ is that it is bounded over the environment, or equivalently, the approximation error is bounded, $|\phi_\epsilon(q)| \leq \phi_{\epsilon_{\max}} < \infty$. We assume that the robots have knowledge of this bound, $\phi_{\epsilon_{\max}}$. In practice, the parameter $\phi_{\epsilon_{\max}}$ is essentially an error tolerance set by the user, and it must be chosen through domain expertise. Such parameters are common in adaptive control with robustification features, as described in, e.g., [30]. The theoretical analysis in the previous work [12] was not robust to function approximation errors in that it required $\phi_\epsilon(q) \equiv 0$. One of the main contributions here is to formulate an algorithm that is provably robust to function approximation errors. We only require that the robots have a known bound for the function approximation error, $\phi_{\epsilon_{\max}}$.

Function approximation errors can occur in real-world applications for many reasons. The field may be too complex to be represented exactly by a finite set of basis functions. For example, the field may be noisy, or the robots may have noisy sensors, leading to a "spiky" field that cannot be exactly represented without some error, as in (6). Hence one of the main motivations for our robust approach is robustness to noise in real-world applications.

Finally, we define the parameter error as $\tilde{a}_i(t) := \hat{a}_i(t) - a$. In what follows we describe an online tuning law by which robot i can tune its parameters, $\hat{a}_i$, to approach a neighborhood of the optimal parameters, a. Our proposed controller then causes the robots to converge to their approximate centroids, $p_i \to \hat{C}_i$ for all i. An overview of the geometrical objects involved in our set-up is shown in Fig. 3.



Q : Convex environment
$\phi(q)$: Weighting function
$p_i(t)$: Robot location
$V_i(P)$: Voronoi cell
$\hat{C}_i(P,t)$: Estimated centroid
$C_i(P)$: True centroid

Fig. 3. A graphical overview of the quantities involved in the controller is shown. The robots move to cover a bounded, convex environment Q their positions are $p_i(t)$, and they each have a Voronoi region $V_i(P)$ with a true centroid $C_i(P)$ and an estimated centroid $\hat{C}_i(P,t)$. The true centroid is determined using a sensory function $\phi(q)$, which indicates the relative importance of points q in Q. The robots do not know $\phi(q)$, so they calculate an estimated centroid using an approximation $\hat{\phi}_i(q,t)$ learned from sensor measurements of $\phi(q)$.

III. Robust Adaptive Coverage Algorithm

In this section we describe the algorithm that drives the robots to spread out over the environment while simultaneously approximating the weighting function online. The algorithm naturally decomposes into two parts: (1) the parameter adaptation law, by which each robot updates its approximate weighting function, and (2) the control algorithm, which drives the robots to explore the environment before moving to their final positions for coverage. We describe these two parts in separate sections and then prove performance guarantees for the two working together.

Throughout this section, we assume that each robot has access to its own Voronoi cell V_i, from which it can compute the mass, centroid, and lengths of edges of V_i as needed. The robots can compute V_i in a distributed way using well-known distributed algorithms [1]. These algorithms require that each robot can communicate directly with its Voronoi neighbors (which tend to be the closest neighbors in the network). In other words, we assume that the graph describing the communication network contains the Delaunay graph.

A. Online Function Approximation

The parameters $\hat{a}_i$ used to calculate $\hat{\phi}_i(q,t)$ are adjusted according to a set of adaptation laws which are introduced below. First, we define two quantities,

$$\Lambda_i(t) = \int_{s\in\Omega_i(t)} \mathcal{K}(s)\mathcal{K}(s)^{\mathrm{T}}\, ds, \quad \text{and} \tag{7}$$

$$\lambda_i(t) = \int_{s\in\Omega_i(t)} \mathcal{K}(s)\phi(s)\, ds, \tag{8}$$

where $\Omega_i(t) = \{s \mid s = p_i(\tau),\ 0 \leq \tau \leq t\}$ is the set of points in the trajectory of p_i from time 0 to time t. The quantities in (7) can be calculated differentially by robot i using $\dot{\Lambda}_i(t) = \mathcal{K}_i(t)\mathcal{K}_i(t)^{\mathrm{T}}\|\dot{p}_i(t)\|$ and $\dot{\lambda}_i(t) = \mathcal{K}_i(t)\phi_i(t)\|\dot{p}_i(t)\|$ with zero initial conditions, where we introduced the shorthand notation $\mathcal{K}_i(t) := \mathcal{K}(p_i(t))$ and $\phi_i(t) := \phi(p_i(t))$. As previously stated, robot i can measure $\phi_i(t)$ continuously in time with its sensors. Now we define another quantity

$$F_i = \frac{\int_{V_i} \mathcal{K}(q)(q-p_i)^{\mathrm{T}}\, dq \int_{V_i} (q-p_i)\mathcal{K}(q)^{\mathrm{T}}\, dq}{\int_{V_i} \hat{\phi}_i(q)\, dq}. \tag{9}$$

Notice that F_i can also be computed by robot i as it does not require any knowledge of the true weighting function, ϕ.

The "pre" adaptation law for $\hat{a}_i$ is now defined as

$$\dot{\hat{a}}_i^{\mathrm{pre}} = -\gamma B_i^{\mathrm{dz}}(\Lambda_i \hat{a}_i - \lambda_i) - \zeta \sum_{j\in\mathcal{N}_i} l_{ij}(\hat{a}_i - \hat{a}_j) - kF_i\hat{a}_i. \tag{10}$$

where γ, ζ, and k are positive gains, l_{ij} is the length of the shared Voronoi edge between robots i and j, and $B_i^{\mathrm{dz}}(\cdot)$ is a dead zone function which gives a zero if its argument is below some value. We will give B_i^{dz} careful attention in what follows as it is the main tool to ensure robustness to function approximation errors. Before describing the dead zone in detail, we note that the three terms in (10) have an intuitive interpretation. The first term is an integral of the function approximation error over the robot's trajectory, so that the parameter $\hat{a}_i$ is tuned to decrease this error. The second is the difference between the robot's parameters and its neighbors' parameters. This term will be shown to lead to parameter consensus; the parameter vectors for all robots will approach a common vector. The third term compensates for uncertainty in the centroid position estimate, and will be shown to ensure convergence of the robots to their estimated centroids. A more in-depth explanation of each of these terms can be found in [12].

Finally, we give the parameter adaptation law by restricting the "pre" adaptation law so that the parameters remain within their prescribed limits $[a_{\min}, a_{\max}]$ using a projection operator. We introduce a matrix I_i^{proj} defined element-wise as

$$I_i^{\mathrm{proj}}(j) := \begin{cases} 0 & \text{for } a_{\min} < \hat{a}_i(j) < a_{\max} \\ 0 & \text{for } \hat{a}_i(j) = a_{\min} \text{ and } \dot{\hat{a}}_i^{\mathrm{pre}}(j) \geq 0 \\ 0 & \text{for } \hat{a}_i(j) = a_{\max} \text{ and } \dot{\hat{a}}_i^{\mathrm{pre}}(j) \leq 0 \\ 1 & \text{otherwise,} \end{cases} \tag{11}$$

where (j) denotes the j^{th} element for a vector and the j^{th} diagonal element for a matrix. The entries of I_i^{proj} are only nonzero if the parameter is about to exceed its bound. Now the parameters are changed according to the adaptation law

$$\dot{\hat{a}}_i = \Gamma(\dot{\hat{a}}_i^{\mathrm{pre}} - I_i^{\mathrm{proj}} \dot{\hat{a}}_i^{\mathrm{pre}}), \tag{12}$$

where $\Gamma \in \mathbb{R}^{m\times m}$ is a diagonal, positive definite gain matrix. Although the adaptation law given by (12) and (10) is notationally complicated, it has a straightforward interpretation, it is of low computational complexity, and it is composed entirely of quantities that can be computed by robot i.

As mentioned above, the key innovation in this adaptation law compared with the one in [12] is the dead zone function B_i^{dz}. We design this function so that the parameters are only changed in response to function errors that could be reduced with different parameters. Without this modification, the algorithm may appear to work in some instances, but in others cases the parameters may go unbounded or they may chatter, leading to divergent robot trajectories. More specifically, the minimal function error that can be achieved is ϕ_ϵ, as shown in (6). Therefore if the integrated parameter error $(\Lambda_i\hat{a}_i - \lambda_i)$ is less than ϕ_ϵ integrated over the robot's path, we have no reason to change the parameters. We will show that the correct form for the dead zone to prevent unnecessary parameter adaptation is

$$B_i^{\mathrm{dz}}(x) = \begin{cases} 0 & \text{if } \|x\| < \|\beta_i\|\phi_{\epsilon_{\max}} \\ \Lambda_i x(\|x\| - \|\beta_i\|\phi_{\epsilon_{\max}}) & \text{otherwise,} \end{cases} \tag{13}$$

where $\beta_i := \int_{s\in\Omega_i(t)} \mathcal{K}(s)\, ds$. The dead zone function can be evaluated by robot i since $\beta_i(t)$ can be computed differentially from $\dot{\beta}_i = \mathcal{K}_i\|\dot{p}_i\|$ with zero initial conditions. With this definition, using (7) and (8), we also have that

$$B_i^{\mathrm{dz}}(\Lambda_i\hat{a}_i - \lambda_i) = B_i^{\mathrm{dz}}(\Lambda_i\tilde{a}_i + \lambda_{\epsilon_i}) \tag{14}$$

where $\lambda_{\epsilon_i} := \int_{s\in\Omega_i(t)} \mathcal{K}(s)\phi_\epsilon(s)\, ds$.

B. Control Algorithm

We propose to use a control algorithm that is composed of a set of control modes, with switching conditions to determine when the robots change from one mode to the next. The robots first move to partition the basis function centers among one another, so that each center is assigned to one robot, then each robot executes a Traveling Salesperson Problem (TSP) tour through all of the basis function centers that have been assigned to it. This tour will provide sufficient information so that the weighting function can be estimated well over all of the environment. Then the robots carry out a centroidal Voronoi controller using the estimated weighting function to drive to final positions. We call the first mode the "partition" mode, the second the "explore" mode, and the third the "cover" mode. This sequence of control modes is executed asynchronously in a distributed fashion, during which the function approximation parameters are updated continually with (12) and (10).

We would like the robots to switch between the three modes, in sequence. However, in order to ensure that the required task from one mode is globally completed before proceeding to the next mode, we require a provably-correct global coordination algorithm for the robots. To do this in a distributed and asynchronous way, in this paper we let every robot maintain an estimate of the mode of every other robot, and we update this estimate using a flooding communication protocol. More formally, for each robot we define a mode variable $\mathcal{M}_{ii} \in \{\texttt{partition}, \texttt{explore}, \texttt{cover}\}$. In order to coordinate their mode switches, each robot also maintains an estimate of the modes of all the other robots, so that $\mathcal{M}_{ij}$ is the estimate by robot i of robot j's mode, and $\mathcal{M}_i := (\mathcal{M}_{i1}, \ldots, \mathcal{M}_{in})$ is an n-tuple of robot i's estimates of all robots' modes. Furthermore, the modes are ordered with respect to one another by $\texttt{partition} < \texttt{explore} < \texttt{cover}$, so that the $\max(\mathcal{M}_i, \mathcal{M}_j)$ function is a tuple containing the element-wise maximum of the two mode estimate tuples, $\mathcal{M}_i$ and $\mathcal{M}_j$, according to this ordering. These mode estimates are updated using the flooding communication protocol described below in Algorithm 2. The conditions for a robot to switch from one mode to another are given formally in Algorithm 1.

Most importantly, Algorithm 1 ensures that the *last* robot has finished the `partition` mode before the *first* robot begins the `explore` phase, guaranteeing that each basis function center is visited by one of the robots. The algorithm provides a mechanism for "artificially synchronizing" the robots, even though they do not operate on synchronous clocks. The two algorithms run concurrently in different threads. The control laws within each mode are then defined as follows.

Algorithm 1 Switching Control Algorithm (executed by robot i)

```
Require: Communication with Voronoi neighbors.
Require: Knowledge of position, $p_i$, in global coordinate frame.
Require: Knowledge of the total number of robots $n$.
Require: Knowledge of flooding algorithm (Algorithm 2) update period, $T$.
Require: Access to mode estimates $\mathcal{M}_i$ updated from Algorithm 2.
  while $\mathcal{M}_i \neq (\texttt{explore}, \ldots, \texttt{explore})$ do
    if $\mathcal{M}_{ii} == \texttt{explore}$ then
      $u_i = [0,\ 0]^{\mathrm{T}}$
    else
      $u_i = u_i^{\texttt{partition}}$
    end if
    if Distance to mean of basis function centers $< \epsilon^{\texttt{partition}}$
    and $\mathcal{M}_{ii} == \texttt{partition}$ then
      $\mathcal{M}_{ii} = \texttt{explore}$
    end if
  end while
  Compute TSP tour through basis function centers $\mathcal{N}_i^{\mu}$
  Wait for $nT$ seconds with $u_i = [0,\ 0]^{\mathrm{T}}$
  Execute TSP tour
  $\mathcal{M}_{ii} = \texttt{cover}$
  while $\mathcal{M}_{ii} == \texttt{cover}$ do
    $u_i = u_i^{\texttt{cover}}$
  end while
```

Algorithm 2 Mode Estimate Flooding (executed by robot i)

```
Require: The network is connected.
Require: The robots have synchronized clocks with which they broadcast during a pre-assigned time slot.
  Initialize $\mathcal{M}_i = (\texttt{partition}, \ldots, \texttt{partition})$
  while 1 do
    if Broadcast received from robot $j$ then
      $\mathcal{M}_i = \max(\mathcal{M}_i, \mathcal{M}_j)$ (where $\texttt{partition} < \texttt{explore} < \texttt{cover}$)
    end if
    if Robot $i$'s turn to broadcast then
      Broadcast $\mathcal{M}_i$
    end if
  end while
```

a) Partition: In the partition mode, each robot uses the controller

$$u_i^{\texttt{partition}} = k\Big(\frac{1}{|\mathcal{N}_i^{\mu}|}\sum_{j\in\mathcal{N}_i^{\mu}} \mu_j - p_i\Big), \tag{15}$$

where $\mathcal{N}_i^{\mu} := \{\mu_j \mid \|\mu_j - p_i\| \leq \|\mu_j - p_k\|\, \forall k \neq i\}$ is the set of the closest basis function centers to robot i, μ_j are the basis function centers from (4), and $|\mathcal{N}_i^{\mu}|$ is the number of elements in $\mathcal{N}_i^{\mu}$. Since each robot knows its Voronoi cell, it can determine which basis function centers lie inside the cell, and these will be the ones closer to it than to any other robot. The positions $p_i(t)$, $p_k(t)$ and the set $\mathcal{N}_i^{\mu}$ evolve continuously in time. Together with (1) this implements a continuous time K-means clustering algorithm. This mode serves to divide the basis function centers into n approximately equal sized groups, one group for each robot, in order to "balance" the load in the `explore` mode. However, if the groups are not equally sized, this has no bearing on the correctness or convergence of our control algorithm.

b) Explore: In the explore mode, each robot drives a tour at a constant speed $\|\dot{p}_i\| = v$ through each basis function center in its neighborhood, μ_j for $j \in \mathcal{N}_i^{\mu}$. Any tour that visits all basis function centers is sufficient for learning (as proved in Lemma 1 below). We choose the TSP tour since it is the shortest such tour, and hence will conserve the robots' energy. Although solving the TSP is NP-hard in the worst case, in practice TSP tours can be computed efficiently with freely available exact or approximate TSP solvers (e.g. the Concorde TSP Solver `http://www.math.uwaterloo.ca/tsp/concorde/`).

c) Cover: Finally, for the cover mode, each robot moves toward the centroid of its Voronoi cell using

$$u_i^{\texttt{cover}} = k(\hat{C}_i - p_i), \tag{16}$$

where k is the same positive gain from (10).

Using the above control and function approximation algorithm, we can prove that all robots converge to the estimated centroid of their Voronoi cells, that all robots' function approximation parameters converge to the same parameter vector, and that this parameter vector has a bounded error with the optimal parameter vector. This is stated formally in the following theorem.

Theorem 1 (Convergence)**.** *A network of robots with dynamics (1) using Algorithm 1 for control, Algorithm 2 for communication, and (10) and (12) for online function approximation has the following convergence guarantees:*

$$\lim_{t\to\infty} \|p_i(t) - \hat{C}_i(P,t)\| = 0 \quad \forall i, \tag{17}$$

$$\lim_{t\to\infty} \|\hat{a}_i(t) - \hat{a}_j(t)\| = 0 \quad \forall i, j, \tag{18}$$

$$\textit{and} \quad \lim_{t\to\infty} \|\hat{a}_i(t) - a\| \leq \frac{\sum_{j=1}^{n} 2\|\beta_j(t)\|\phi_{\epsilon_{\max}}}{\operatorname{mineig}\big(\sum_{j=1}^{n}\Lambda_j(t)\big)} \quad \forall i. \tag{19}$$

The proof of this theorem depends upon the following two Lemmas, whose proofs are given in the Appendix.

Lemma 1 (Complete Exploration)**.** *If every basis function center is visited by at least one robot, then $\sum_{i=1}^{n}\Lambda_i(t) > 0$ for all $t > \max_j t_j$, where t_j is the first time that the jth basis function center was visited.*

Lemma 2 (Uniform Continuity)**.** *The following functions are uniformly continuous:*

$$\psi_1(t) = -\sum_{i=1}^{n} \|\hat{C}_i - p_i\|^2 k\hat{M}_i,\ \psi_2(t) = -\zeta \sum_{j=1}^{m} \hat{\alpha}_j^T L(P)\hat{\alpha}_j,$$
$$\textit{and}\quad \psi_3(t) = -\sum_{i=1}^{n} \gamma \tilde{a}_i^T B_i^{dz}(\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}).$$

Proof of Theorem 1: The proof has two parts. The first part is to show that all robots reach "cover" mode and stay in "cover" mode. The second part uses a Lyapunov type proof technique similar to the one in [12] to show that once all robots are in "cover" mode, the convergence claims of (17), (18), and (19) follow.

Firstly, the "partition" mode simply implements a K-means clustering algorithm [38], in which the basis function centers are the points to be clustered, and the robots move to the cluster means. This algorithm is well-known to converge in the sense that for any $\epsilon^{\texttt{partition}}$ there exists a time $T_i^{\texttt{partition}}$ at which the distance between the robot and the centers' mean is less than $\epsilon^{\texttt{partition}}$, therefore according to Algorithm 1 all robots will reach $\mathcal{M}_{ii} = \texttt{explore}$ at some finite time. After this time, according to Algorithm 1, a robot will remain stopped until all of its mode estimates have switched to "explore."

Now we prove that for any robot, at the time when all of its mode estimates have switched to "explore," all of the other robots have completed the "partition" mode and they are stopped in fixed positions, thus ensuring a correct partition of the environment. Suppose the first robot to achieve $\mathcal{M}_i = (\texttt{explore}, \ldots, \texttt{explore})$ does so at time T_f. This means that at some time in the past all the other robots, j, have switched to $\mathcal{M}_{jj} = \texttt{explore}$ and stopped moving, but none of them have $\mathcal{M}_j = (\texttt{explore}, \ldots, \texttt{explore})$ (otherwise they would be the first). Therefore at T_f all robots have completed "partition" mode and are stopped. Suppose the last robot to achieve $\mathcal{M}_i = (\texttt{explore}, \ldots, \texttt{explore})$ does so at T_l. From the properties of the standard flooding algorithm, Algorithm 2, we know that $T_l - T_f \leq nT$ (the maximum time between the first robot to obtain $\mathcal{M}_i = (\texttt{explore}, \ldots, \texttt{explore})$ and the last robot to do so is nT). Therefore, at time T_l the first robot to have $\mathcal{M}_i = (\texttt{explore}, \ldots, \texttt{explore})$ will still be stopped, because it waits for nT seconds after achieving $\mathcal{M}_i = (\texttt{explore}, \ldots, \texttt{explore})$. We have established that all robots are stopped in the time interval $[T_f, T_l]$. When any robot obtains $\mathcal{M}_i = (\texttt{explore}, \ldots, \texttt{explore})$ it will do so at some time T_i satisfying $T_f \leq T_i \leq T_l$, hence all robots will be stopped at this time. At time T_i robot i computes its TSP tour, according to Algorithm 1. Even though the robots may compute their TSP tours at different times, they are all at the same fixed positions when they do so, according to the above reasoning. Therefore, each basis function center is in at least one robot's TSP tour.

After all robots finish their TSP tours in "explore" mode, we have $\operatorname{mineig}(\sum_{i=1}^n \Lambda_i) > 0$, from Lemma 1 since all basis function centers have been visited by at least one robot. This ensures that the bound in (19) is meaningful. Intuitively, when the robots have completely explored the environment, $\operatorname{mineig}(\sum_{i=1}^n \Lambda_i)$ is large, making the bound in (19) small.

Each tour is finite length, so it will terminate in finite time, hence each robot will eventually enter the "cover" mode. Furthermore, when some robots are in "cover" mode, and some are still in "explore" mode, the robots in "cover" mode will remain inside the environment Q. This is because Q is convex, and since $\hat{C}_i \in V_i \subset Q$ and $p_i \in Q$, by convexity, the segment connecting the two is in Q. Since the robots have integrator dynamics (1), they will stay within the union of these segments over time, $p_i(t) \in \cup_{\tau > 0}(C_i(\tau) - p_i(\tau))$, and therefore remain in the environment Q. Thus at some finite time, $T^{\texttt{cover}}$, all robots reach "cover" mode and are at positions inside Q. Furthermore, they never change from "cover" mode once they have reached it, as indicated in Algorithm 1, which completes the first part of the proof.

Now for the second part of the proof, define a Lyapunov-like function

$$\mathcal{V} = \mathcal{H} + \frac{1}{2}\sum_{i=1}^{n} \tilde{a}_i^{\mathrm{T}} \Gamma^{-1} \tilde{a}_i, \tag{20}$$

which incorporates the sensing cost $\mathcal{H}$, and is quadratic in the parameter errors $\tilde{a}_i$. We will use several applications of Barbălat's lemma [39], [40] to prove (17), (18), and (19). Barbălat's lemma states that for a function $\psi(t)$, if the limit of its integral exists and is bounded, $\int_0^\infty \psi(\tau)\,d\tau = \Psi_\infty$ with $|\Psi_\infty| < \infty$, and if $\psi(t)$ is uniformly continuous, then $\lim_{t\to\infty} \psi(t) = 0$.

Notice that $\mathcal{V} \geq 0$. We will first show that $\dot{\mathcal{V}} \leq 0$, which implies that $\lim_{t\to\infty} \mathcal{V} = \mathcal{V}_\infty$ exists and is finite (this is a basic theorem of real analysis). Taking the time derivative of $\mathcal{V}$ along the trajectories of the system and simplifying with (3) gives

$$\dot{\mathcal{V}} = \sum_{i=1}^{n}\left(-\|\hat{C}_i - p_i\|^2 k\hat{M}_i + \tilde{a}_i^{\mathrm{T}} k F_i \hat{a}_i + \tilde{a}_i^{\mathrm{T}} \Gamma^{-1} \dot{\hat{a}}_i\right).$$

Substituting for $\dot{\hat{a}}_i$ with (12), (10), and (14) gives

$$\dot{\mathcal{V}} = -\sum_{i=1}^{n}\Big(\|\hat{C}_i - p_i\|^2 k\hat{M}_i + \gamma \tilde{a}_i^{\mathrm{T}} B_i^{\mathrm{dz}}(\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}) + \zeta \tilde{a}_i^{\mathrm{T}} \sum_{j\in\mathcal{N}_i} l_{ij}(\hat{a}_i - \hat{a}_j) + \tilde{a}_i^{\mathrm{T}} I_i^{\mathrm{proj}} \dot{\hat{a}}_i^{\mathrm{pre}}\Big).$$

Rearranging terms we get

$$\dot{\mathcal{V}} = -\sum_{i=1}^{n}\Big(\|\hat{C}_i - p_i\|^2 k\hat{M}_i + \gamma \tilde{a}_i^{\mathrm{T}} B_i^{\mathrm{dz}}(\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}) + \tilde{a}_i^{\mathrm{T}} I_i^{\mathrm{proj}} \dot{\hat{a}}_i^{\mathrm{pre}}\Big) - \zeta \sum_{j=1}^{m} \hat{\alpha}_j^{\mathrm{T}} L(P) \hat{\alpha}_j, \tag{21}$$

where $\hat{\alpha}_j := [\hat{a}_{1j} \cdots \hat{a}_{mj}]^{\mathrm{T}}$ is the j^{th} element in every robot's parameter vector, stacked into a vector, and $L(P) \geq 0$ is the graph Laplacian for the Delaunay graph which defines the robots communication network (please refer to the proof of Theorem 2 in [12] for details). Define $\psi_1(t) := -\sum_{i=1}^n \|\hat{C}_i - p_i\|^2 k\hat{M}_i$, $\psi_2(t) := -\zeta \sum_{j=1}^m \hat{\alpha}_j^{\mathrm{T}} L(P)\hat{\alpha}_j$,

$\psi_3(t) := -\sum_{i=1}^n \gamma \tilde{a}_i^{\mathrm{T}} B_i^{\mathrm{dz}}(\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i})$, and $\psi_4(t) := -\sum_{i=1}^n \tilde{a}_i^{\mathrm{T}} I_i^{\mathrm{proj}} \dot{\hat{a}}_i^{\mathrm{pre}}$, which gives $\dot{\mathcal{V}} = \psi_1(t) + \psi_2(t) + \psi_3(t) + \psi_4(t)$. Notice that $\psi_1(t)$ is a negative sum of the square of norms, and therefore is non-positive, and $\psi_2(t)$ is non-positive because $L(P) \geq 0$. Furthermore, $\psi_4(t)$ is non-positive by the design of the projection operation in (11) (this is shown in the proof of Theorem 1 in [12]). Therefore, $\psi_3(t)$ is the only term in question, and this term distinguishes the Lyapunov construction here from the one in the proof of Theorem 1 in [12].

We now show that $\psi_3(t)$ is also non-positive by design. Suppose $\|\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}\| < \|\beta_i\| \phi_{\epsilon_{\max}}$ from (13) for some i. Then $B_i^{\mathrm{dz}}(\Lambda \tilde{a}_i + \lambda_{\epsilon_i}) = 0$ and the ith contribution to the term $\psi_3(t)$ is zero. Now suppose $\|\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}\| \geq \|\beta_i\| \phi_{\epsilon_{\max}}$ for some i. In that case we have

$$\begin{aligned} 0 &\leq \|\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}\| - \Big\| \int_{\Omega_i(t)} \mathcal{K}(s) \phi_\epsilon(s)\, ds \Big\| \\ &\leq \|\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}\|^2 - \lambda_{\epsilon_i}^{\mathrm{T}} (\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}) \\ &\leq \tilde{a}_i^{\mathrm{T}} \Lambda_i (\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}). \end{aligned} \tag{22}$$

Then from the definition of the dead-zone operator, $B_i^{\mathrm{dz}}(\cdot)$, we have

$$\begin{aligned} &\tilde{a}_i^{\mathrm{T}} B_i^{\mathrm{dz}}(\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}) = \\ &\qquad \tilde{a}_i^{\mathrm{T}} \Lambda_i (\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i})(\|\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}\| - \|\beta_i\| \phi_{\epsilon_{\max}}) \geq 0, \end{aligned}$$

since the $\tilde{a}_i^{\mathrm{T}} \Lambda_i (\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i})$ was shown to be non-negative in (22), and $\|\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}\| - \|\beta_i\| \phi_{\epsilon_{\max}}$ is non-negative by supposition. Therefore the ith contribution to $\psi_3(t)$ is non-positive in this case. Then $\psi_3(t)$ is a sum of non-positive quantities, hence $\psi_3(t) \leq 0$. We conclude therefore that $\dot{\mathcal{V}} \leq 0$, and since $\mathcal{V} \geq 0$, this means $\lim_{t\to\infty} \mathcal{V} = \mathcal{V}_\infty$ exists and is finite.

We now apply Barbălat's lemma to $\psi_1(t)$, $\psi_2(t)$, and $\psi_3(t)$ to get the three main results of the theorem. We already established that each of the $\psi_i(t)$ terms is non-positive, therefore their integrals must be non-positive, $\Psi_i(t) := \int_0^t \psi_i(\tau)\, d\tau \leq 0$ for all t and for all $i = 1, 2, 3, 4$. We have that $\mathcal{V} = \sum_{i=1}^4 \Psi_i(t) + \mathcal{V}_0 \geq 0$, hence $\Psi_i(t) \geq -\mathcal{V}_0$, $\forall i$. Now since $\Psi_i(t)$ are all lower bounded by $-\mathcal{V}_0$ and they have non-positive time derivatives, $\psi_i(t) \leq 0$, we conclude that each one reaches a finite limit $\Psi_{i,\infty}$ (again, this is a fundamental theorem of real analysis). Now, to apply Barbălat's lemma to $\psi_1(t)$, $\psi_2(t)$, and $\psi_3(t)$, we only need to show that they are uniformly continuous.[4] This is given by Lemma 2, which is proved in the Appendix, hence we conclude that $\psi_k(t) \to 0$, $k = 1, 2, 3$.

Now we show that $\psi_k(t) \to 0$, $k = 1, 2, 3$ implies the convergence claims stated in the theorem. It is clear that $\psi_1(t) \to 0$ gives the position convergence (17), and $\psi_3(t) \to 0$ gives parameter consensus (18) (again, see [12] for more details on these). Finally, we verify the parameter error convergence (19). We know that the dead-zone term approaches zero, $\psi_3(t) \to 0$, therefore for all i either $\lim_{t\to\infty} \tilde{a}_i^{\mathrm{T}} \Lambda_i (\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}) = 0$, or $\lim_{t\to\infty} (\|\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}\| - \|\beta_i\| \phi_{\epsilon_{\max}}) < 0$. We already saw from (22) that $\tilde{a}_i^{\mathrm{T}} \Lambda_i (\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}) = 0$ implies $\|\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}\| - \|\beta_i\| \phi_{\epsilon_{\max}} \leq 0$, thus we only consider this later case. Then from $\lim_{t\to\infty} (\|\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}\| - \|\beta_i\| \phi_{\epsilon_{\max}}) \leq 0$ we have

$$0 \geq \lim_{t\to\infty} \big(\|\Lambda_i \tilde{a}_i + \lambda_{\epsilon_i}\| - \|\beta_i\| \phi_{\epsilon_{\max}} \big),$$

and because of parameter consensus (18), $0 \geq \lim_{t\to\infty} \big(\|\Lambda_j \tilde{a}_i + \lambda_{\epsilon_j}\| - \|\beta_j\| \phi_{\epsilon_{\max}} \big)$ for all j. Then summing over j, we have

$$\begin{aligned} 0 &\geq \lim_{t\to\infty} \Big(\sum_{j=1}^n \|\Lambda_j \tilde{a}_i + \lambda_{\epsilon_j}\| - \sum_{j=1}^n \|\beta_j\| \phi_{\epsilon_{\max}} \Big) \\ &\geq \lim_{t\to\infty} \Big(\Big\| \sum_{j=1}^n \Lambda_j \tilde{a}_i + \sum_{j=1}^n \lambda_{\epsilon_j} \Big\| - \sum_{j=1}^n \|\beta_j\| \phi_{\epsilon_{\max}} \Big) \\ &\geq \lim_{t\to\infty} \Big(\Big| \Big\| \sum_{j=1}^n \Lambda_j \tilde{a}_i \Big\| - \Big\| \sum_{j=1}^n \lambda_{\epsilon_j} \Big\| \Big| - \sum_{j=1}^n \|\beta_j\| \phi_{\epsilon_{\max}} \Big). \end{aligned}$$

The last condition has two possibilities; either $\lim_{t\to\infty} \big(\| \sum_{j=1}^n \Lambda_j \tilde{a}_i \| > \| \sum_{j=1}^n \lambda_{\epsilon_j} \| \big)$, in which case

$$\begin{aligned} 0 \geq \lim_{t\to\infty} \Big(\Big\| \sum_{j=1}^n \Lambda_j \tilde{a}_i \Big\| - \Big\| \sum_{j=1}^n \lambda_{\epsilon_j} \Big\| - \sum_{j=1}^n \|\beta_j\| \phi_{\epsilon_{\max}} \Big) \geq \\ \lim_{t\to\infty} \Big(\Big\| \sum_{j=1}^n \Lambda_j \tilde{a}_i \Big\| - 2 \sum_{j=1}^n \|\beta_j\| \phi_{\epsilon_{\max}} \Big), \end{aligned} \tag{23}$$

where the last inequality uses the fact that $\| \sum_{j=1}^n \lambda_{\epsilon_j} \| \leq \sum_{j=1}^n \|\beta_j\| \phi_{\epsilon_{\max}} \|$. Otherwise, $\lim_{t\to\infty} \big(\| \sum_{j=1}^n \Lambda_j \tilde{a}_i \| < \| \sum_{j=1}^n \lambda_{\epsilon_j} \| \big)$, which implies $\lim_{t\to\infty} \big(\| \sum_{j=1}^n \Lambda_j \tilde{a}_i \| < \sum_{j=1}^n \|\beta_j\| \phi_{\epsilon_{\max}} \big)$ which in turn implies (23), thus we only need to consider (23). This expression then leads to

$$0 \geq \lim_{t\to\infty} \big(\mathrm{mineig}\big(\sum_{j=1}^n \Lambda_j \big) \|\tilde{a}_i\| - 2 \sum_{j=1}^n \|\beta_j\| \phi_{\epsilon_{\max}} \big),$$

and dividing both sides by $\mathrm{mineig}\big(\sum_{j=1}^n \Lambda_j \big)$ (which is strictly positive according to Lemma 1), gives (19). ■

Remark 1 (Optimality)**.** *The algorithm drives the robots to converge to a locally optimal sensing configuration with respect to their learned weighting function. Furthermore, all robots converge to the same learned weighting function, as ensured by (18). This means that their positions locally optimize the coverage cost function (2) where $\phi(q)$ is replaced with the learned weighting function $\hat{\phi}(q) = \lim_{t\to\infty} \mathcal{K}(q)^T \hat{a}_i(t)$. This locally optimal performance is the state of the art, even when the weighting function is fully known before hand [1]. In fact, it is known that the optimization of (2) is NP-hard when $\phi(q)$ is known [3], hence we cannot expect to find the global optimum, particularly when $\phi(q)$ is not known.*

In practice we find that most locally optimal positions are of nearly equal quality, and they are not appreciably worse than the global optimum. If better local optima are desired, one can implement a simulated annealing style algorithm by adding a small amount of noise to the robots' control input, with asymptotically decreasing magnitude. Indeed, in any hardware implementation there will be actuation noise, which

[4]In fact, $\psi_4(t)$ is not uniformly continuous, but this is inconsequential to our proof.

will lead the robots to dislodge themselves from undesirable local minima.

Remark 2 (Choosing Parameters)**.** *The algorithm has several parameters that must be tuned by the user. These include γ, Γ, k, ζ. The results of the theorem hold regardless of the specific values of these gains. The appropriate choices for the gains depend greatly upon the specific hardware and electronics one uses. In practice they should be tuned roughly in simulation, and then fine-tuned experimentally. The values used in our experiments are recorded in Sec. V.*

Furthermore, one may wonder what is the dependency of the performance on the number of basis functions m, and the size of the known function error bound $\phi_{\epsilon_{\max}}$. It is straightforward to show that the algorithm execution time scales linearly in m. Also, the function approximation error grows no worse than linearly in $\phi_{\epsilon_{\max}}$, as evident from (19) in Theorem 1.

Remark 3 (Failure Conditions)**.** *The results of Theorem 1 hold strictly under the modeling assumptions we describe above. In practice these assumptions will not hold exactly, hence it is natural to wonder about the algorithm performance in these cases. Empirically, we find that the algorithm does not "fail" in the sense of a divergence, or an instability. In the case of noise, robot dynamics, or an incorrect function approximation bound $\phi_{\epsilon_{\max}}$, the algorithm performance degrades gently. This is evidenced by our hardware experiments, in which the algorithm performs as expected despite the presence of these sources of error.*

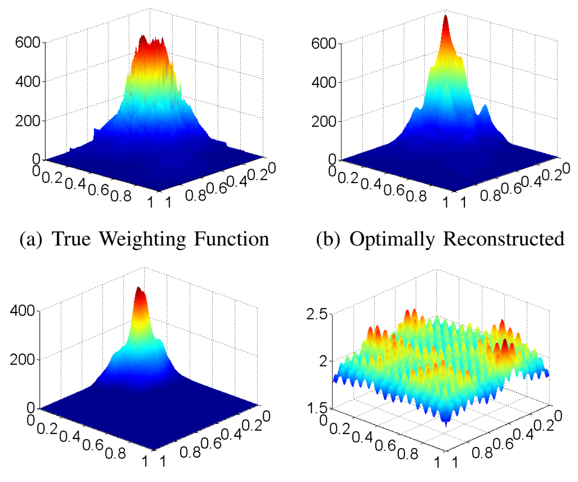
Remark 4 (Robot Initial Positions)**.** *Strictly speaking, robots do not have to start within the region Q. If a robot starts outside Q, one of two things may happen: (i) the robot will never move because its Voronoi cell is empty, or (ii) it will eventually enter Q over the course of the execution of the controller, and once it enters it will not leave.*

Remark 5 (Extensions to Nonconvex Environments and Time Varying Environments)**.** *The algorithm can also be directly extended to several classes of non-convex environment, including ones with occlusions and obstacles, by using the coverage control methods from, e.g., [6]–[8] in place of the basic coverage controller (16). Furthermore, the algorithm will perform well for a time-varying weighting function $\phi(q,t)$, as long as the weighting function changes slowly compared with the speed of the robots and the convergence rate of the learning algorithm.*

IV. Simulations

The proposed algorithm is tested using the data collected from previous experiments [41] in which two incandescent office lights were placed at the position (0,0) of the environment, and the robots used on-board light sensors to measure the light intensity. The data collected during these previous experiments was used to generate a realistic weighting function which cannot be reconstructed exactly by the chosen basis functions. In the simulation, there are 10 robots and the basis functions are arranged on a 15×15 grid in the environment.

The proposed robust algorithm was compared against the standard algorithm from [12], which assumes that the weighting function can be matched exactly. The true and optimally reconstructed (according to (5)) weighting functions are shown in Fig. 4(a) and (b), respectively. As shown in Fig. 4(c) and (d), with the robust and standard algorithm respectively, the proposed robust algorithm significantly outperforms the standard algorithm, and reconstructs the true weighting function well. The robot trajectories for the robust and standard adaptive coverage algorithm are shown in Fig. 5(a) and (b), respectively. Since the standard algorithm does not include the exploration phase, the robots get stuck in a local area around their starting position, which causes the robots to be unsuccessful in learning an acceptable model of the weighting function. In contrast, the robots with the robust algorithm explore the entire space and reconstruct the true weighting function well.


(a) True Weighting Function (b) Optimally Reconstructed

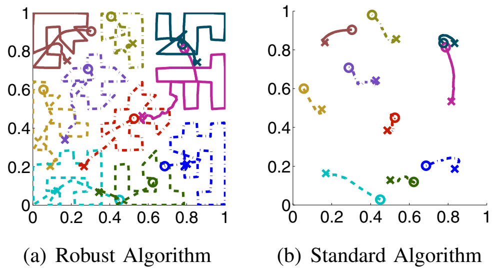
(c) Robust Algorithm (d) Standard Algorithm

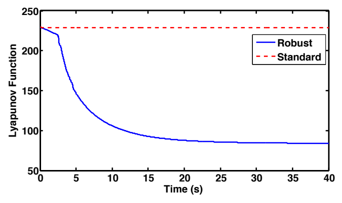
Fig. 4. A comparison of the weighting functions. (a) The true weighting function. (b) The optimal, offline, centrally reconstructed weighting function for the chosen basis functions. (c) The weighting function for the proposed algorithm with dead-zone and exploration. (d) The previously proposed algorithm without dead-zone or exploration (note the different z-axis scale).

V. Hardware Experiments

In addition to verifying the algorithm in simulation, we implemented the algorithm on a group of robots to demonstrate its practical feasibility. Six iRobot Creates with Asus Eee PC 1015PXs were used as the robot platforms. The robots sensed light intensity as the weighting function, as measured by an Advanced Photonix PDV-P9008 photocell mounted on each robot. The Robot Operating System (ROS) was used as the means of communication and Python was chosen as the language of implementation. For localizing the robots, a room outfitted with a Vicon motion capture system was used. As is usually the case, several modifications from the theoretical algorithm were required for a hardware implementation, of which the most significant are described here.

Collisions: While collisions were not treated directly in the development of the algorithm and were not accounted for in the simulation, they had to be addressed in the hardware implementation. Our method for avoiding collisions was to draw a line between the robot and its goal. The robot's


(a) Robust Algorithm (b) Standard Algorithm



(c) Lyapunov Function

Fig. 5. The vehicle trajectories for the different control strategies and the values of their Lyapunov functions are shown. The initial and final vehicle position is marked by a circle and cross, respectively. It is evident from the Lyapunov function values that the robust algorithm significantly outperforms the standard algorithm.

position is then projected along this line for two robot radii to check for imminent collisions. If a collision is detected, the robot determines the relative heading of the other robot with respect to itself, turns 90 degrees in the opposite direction, and travels a fixed distance. This simple algorithm was seen to be sufficiently robust in practice, though a more sophisticated collision avoidance scheme would be preferable if collisions are a critical concern, such as for quadrotors or other aircraft.

Integration: In implementing the time integration required for the controller and parameter adaptation laws, there were two modifications from the mathematical model. Firstly, a fixed time step of 0.01s was used in computing the integration, although in actuality the time taken for a single update of the equations varied considerably from one time step to the next due to variable wireless transmission delays and other hard-to-predict delays in the system. We found that assuming a fixed time step provided a more robust solution despite the variation in actual update times.

Secondly, the dead-zone function used in the experiments was a normalized version of the one described in the theoretical development above. Specifically, the dead-zone was computed as

$$B_i^{\mathrm{dz}}(x) = \begin{cases} 0 & \text{if } C(x) < 0 \\ \frac{x}{\|x\|} C(x) & \text{otherwise,} \end{cases}$$

with

$$C(x) = \|\Lambda_i^{1/2}\| \times \Big(\|\Lambda_i^{-1/2} x\| - \|\Lambda_i^{1/2} \beta_i\| \phi_{\epsilon_{\max}} - \|\Lambda_i^{1/2} \Lambda_0\| a_{\max} \Big),$$

where $\Lambda_0 > 0$ is a small but positive definite initial condition for Λ_i. Although it looks more complicated, this dead-zone is a normalized version of (13). The reason for the change is that the normalized version was found to require less fine tuning of the adaptation gain γ in practice. We chose to use the unnormalized dead-zone in the theoretical development, both because it has a simpler mathematical form, and because it is more straightforward to prove some of the technicalities required for the parameter convergence result in Theorem 1.

Delay: Another difficulty not accounted for directly in the analysis or simulations is delays in communication. The algorithm is fully decentralized, however to avoid practical difficulties with mesh network hardware, we simulated the network on a base computer, and computed the control actions of each robot on this base computer. We then used 802.11 (Wi-Fi) to transmit waypoint commands to the iRobot Creates from the base station. Wi-Fi transmission rates are known to depend on interference and room geometry in hard-to-predict ways. We mitigated the effects of variable delays due to wireless transmission by choosing our control and adaptation gains to be suitably low, and by choosing an environment discretization that was coarse enough to give relatively small wireless packets, while preserving control performance.

Robot Motion: In our analysis and simulation results, robots are represented as noiseless point integrators. In practice, however, the robots' motion incorporate noise, and a controller is required to move the robots to desired positions. For the experiments, a waypoint controller was implemented to rotate the robot to the desired heading within some tolerance, and then drive directly to the waypoint.

A. Experimental Setup

Two groups of experiments were conducted with the six iRobot Creates; the first tested for robustness to real-world complications such as sensing and actuation noise, while the second tested specifically for sensitivity to positional errors in the robots. The control and adaptation gains used in the control algorithm were $k = 3$, $\zeta = 30$, $\gamma = 0.1$, and $\Gamma = I$. For estimating the weighting function we used an 8×8 grid of Gaussian basis functions (hence $m = 64$) with standard deviation $\sigma = 0.55$m. The function error bound was $\phi_{\epsilon_{\max}} = 0.3$, and the minimum and maximum parameters were $a_{min} = 0.001$, $a_{max} = 10$, respectively.

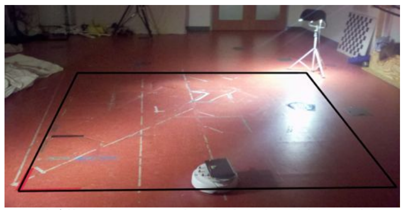


Fig. 6. The environment used in the experiments, with one robot shown. There are light sources in the bottom left and upper right corners, though the source in the lower left is not shown in the frame of the photograph. The robots' environment is the 3m×3m region outlined in black.

Experimental Setup 1: The first experimental setup was run 11 times, each with the same light placement, shown in Fig. 6. The environment was a 3m×3m square shown in the figure. A 10×10 grid was used for the spatial integration to compute $\hat{M}_i$, $\hat{C}_i$ and F_i, as required for the control algorithm. Six robots were used in each run, with their starting locations varied from one run to the next. The light sensors were placed at the center of the robot, so that the robot's position was the same as the sensor's position.

Experimental Setup 2: This experimental setup was repeated for three runs and was the same as Experimental Setup 1 in all respects, except that the light sensors were offset 15 cm from the center from the robot. This placement introduced a significant positional error in the location of the sensor, depending both on the location and the angle of the robot. This was to test the robustness of the algorithm to a significantly source of positional error.

B. Results

The experimental results indicate that the robots are able to repeatably reconstruct the light field and position themselves at locally-optimal positions for sensing, despite a significant function approximation errors, as predicted by our theoretical results.

Robot trajectories and the learned light intensity function from one run with Experimental Setup 1 is shown in Fig. 7. Please see the accompanying movie to view the animated experimental data. The left frame shows the state of the 6 iCreate robots after the `partition` mode, when they have divided the basis function centers amongst themselves. The center frame shows the trajectories of the robots in the `explore` mode, during which they make a tour through all the basis function centers. The right most frame shows the robots in their final coverage configuration at the end of the `cover` mode, with their approximate trajectories drawn. The final learned weighting function is represented by the underlying heat map.

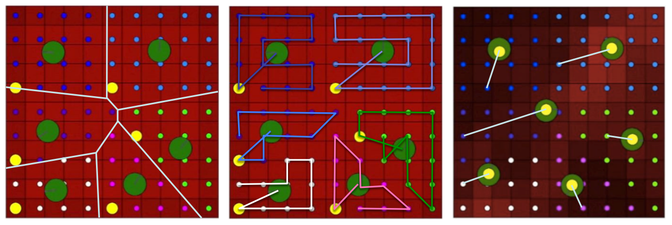


Fig. 7. This figure shows robot positions and trajectories for one of the runs from Experimental Setup 1 with 6 iCreate robots. Green dots indicate robot positions, while yellow dots show the next robot waypoint. Left: The state of the system is shown at the end of `partition` mode, with all waypoints partitioned amongst the robots. Center: The trajectories of the robots during `explore` mode, visiting all basis function centers. Right: The final robot positions in `cover` mode, with the learned weighting function shown as a heat map.

The performance of the algorithm in reconstructing the light field in the two experimental setups is shown in Figs. 8 and 9. The left panels in Figs. 8 and 9 show an ideal centralized, off-line reconstruction of the field, based on the light intensity data collected by one robot over all trials for Experiments 1 and 2 respectively. The middle panels show the light field as reconstructed with our algorithm from one robot, averaged over all trials for Experiments 1 and 2, respectively. The right panels show the difference between the ideal and estimated fields for Experiments 1 and 2, respectively. The figures show that the light field was estimated well in both cases, but the positional errors introduced in Experimental Setup 2 noticeably degrades the learning performance.

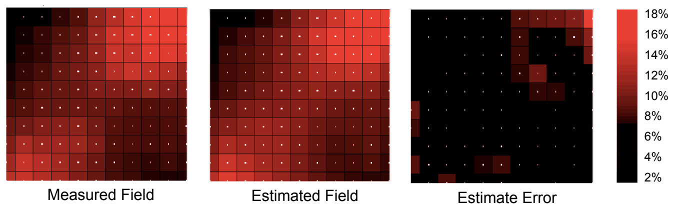


Fig. 8. The field estimation results for Experimental Setup 1 are shown here. Left: The light intensity field as reconstructed offline from all measurements. Middle: The light intensity field estimated by one robot averaged over the 11 runs. Right: The error between the two fields. The shade of red represents light intensity relative to the maximum measurable by the sensor, and the white boxes represent the standard deviation.

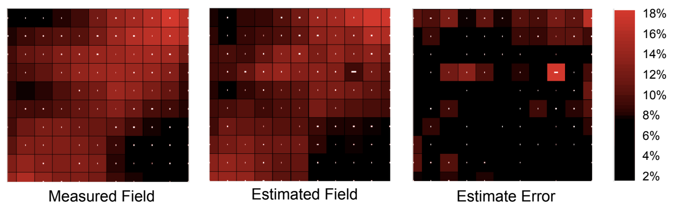
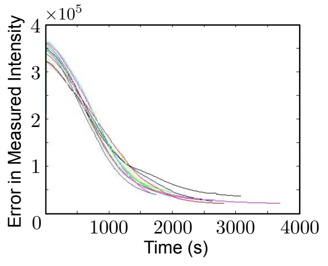
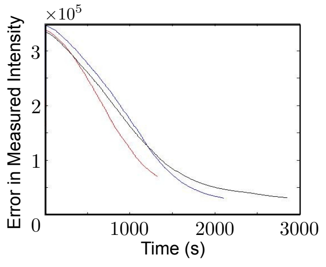


Fig. 9. This figure shows the field estimation results for Experimental Setup 2. Left: The light intensity field as reconstructed offline from all measurements. Middle: The light intensity field estimated by one robot averaged over the 3 runs. Right: The error between the two fields. The shade of red represents light intensity relative to the maximum measurable by the sensor, and the white boxes represent the standard deviation.

The function approximation error, averaged over all runs and all robots for Experimental Setup 1 was 4.6%, while it was 5.8% for Experimental Setup 2. Similarly, the maximum errors were 18.3% and 18.1%, respectively. This indicates that the algorithm performed slightly better in Experimental Setup 1, as would be expected, however the performance did not decrease significantly in Experimental Setup 2, despite the significant positional error of the sensors, indicating that the algorithm is robust to such errors.



(a) Experiment 1



(b) Experiment 2

Fig. 10. The error in the estimated light intensity field as measured by one robot over time is shown here for all runs and for both experimental setups. In all cases, the error decreases over time and settles to a small non-zero value, as predicted by Theorem 1.

The evolution of the function approximation error over time from one robot for each of the 11 trials in Experimental Setup 1, and for each of the 3 trials in Experimental Setup 2, is shown in Fig. 10. One can see that for each trial, the error decreases over time, and eventually settles to a value that is small, but not zero, as predicted by Theorem 1. Performance in Experimental Setup 1 is marginally better than in Experimental

Setup 2, as a slightly lower value of the error is attained. This is what we would expect, since we intentionally introduced positional errors for the sensors in Experimental Setup 2. These results again indicate that the algorithm is robust to real-world errors.

VI. Conclusions

In this paper, we formulated a distributed control and function approximation algorithm for deploying a robotic sensor network to adaptively monitor an environment. The robots robustly learn a weighting function over the environment representing where sensing is most needed. The function learning is enabled by the robots exploring the environment in a systematic and distributed way, and is provably robust to function approximation errors. After exploring the environment, the robots drive to positions that locally minimize a cost function representing the sensing quality of the network. The performance of the algorithm is proven in a theorem, and demonstrated in a numerical simulation with an empirical weighting function derived from light intensity measurements in a room. The algorithm was also implemented on six iRobot Create platforms. Robust performance under realistic conditions was demonstrated in a total of 14 experimental trials in 2 different configurations.

Appendix

Theorem 2 (Gaussian Kernel Matrix is Positive Definite)**.** *The matrix whose entries are given by*

$$k_{ij} = \frac{1}{2\pi\sigma^2}\exp\left\{-\frac{\|\mu_i-\mu_j\|^2}{2\sigma^2}\right\},$$

is positive definite for any m unique vectors $\mu_i \in \mathbb{R}^2$, $i = 1,\ldots,m$.

Proof of Lemma 1: The proof relies on the above well-known theorem from the theory of kernel machines. The theorem follows directly from [42] Corollary 2.1 (c), or more recently, [43] Propositions 6 and 8.

We will show that after $t > \max_j t_j$, the matrix $\sum_{i=1}^n \Lambda_i(t)$ can be expressed as the sum of two quantities

$$\sum_{i=1}^n \Lambda_i(t) = \sum_{j=1}^m \int_{s\in\omega_j} \mathcal{K}(s)\mathcal{K}(s)^{\mathrm{T}}\,ds + \sum_{i=1}^n \int_{s\in\Omega_i(t)\setminus\cup_{j=1}^m\omega_j} \mathcal{K}(s)\mathcal{K}(s)^{\mathrm{T}}\,ds, \quad (24)$$

where the first matrix sum on the right hand side is positive definite, and the second is positive semi-definite, hence their sum is positive definite.

Let i_j be the robot for which $p_{i_j} = \mu_j$ at time t_j. Then we have $\mu_j \in \Omega_{i_j}(t > t_j)$ (recall that $\Omega_i(t)$ is the set of points visited by robot i up until time t). Consider a neighborhood $\omega_j \subset \Omega_{i_j}$ centered at μ_j of length $\delta_s > 0$. More formally

$$\omega_j := \{s | s = p_{i_j}(t_j+\tau), \tau \in [-\delta_s/2v, \delta_s/2v]\},$$

where v is the constant speed at which the robots traverse their TSP tour. Also, let $\delta_s < \min_{j\neq k}\|\mu_j - \mu_k\|$, which ensures that all ω_j are disjoint, $\omega_j \cap \omega_k = \emptyset$, for all $j \neq k$. Now we can divide each $\Omega_i(t)$ into a union of disjoint sets $\Omega_i(t) = \big(\cup_{\{j|p_i(t_j)=\mu_j\}}\omega_j\big) \cup \Omega_i(t)\setminus\cup_{j=1}^m \omega_j$, which leads to

$$\Lambda_i(t) = \sum_{\{j|p_i(t_j)=\mu_j\}} \int_{\omega_j} \mathcal{K}(s)\mathcal{K}(s)^{\mathrm{T}}\,ds + \int_{\Omega_i(t)\setminus\cup_{j=1}^m\omega_j} \mathcal{K}(s)\mathcal{K}(s)^{\mathrm{T}}\,ds.$$

Summing over all i gives the decomposition in (24). Both sums on the right hand side of (24) are positive semi-definite since $\mathcal{K}(s)\mathcal{K}(s)^{\mathrm{T}} \geq 0$ for all s, as it is an outer product of a vector with itself.

We only have to show that $\sum_{j=1}^m \int_{\omega_j}\mathcal{K}(s)\mathcal{K}(s)^{\mathrm{T}}\,ds > 0$. To this end, re-write this as

$$\sum_{j=1}^m \int_{\omega_j} \mathcal{K}(s)\mathcal{K}(s)^{\mathrm{T}}\,ds = \sum_{j=1}^m \mathcal{K}(\mu_j)\mathcal{K}(\mu_j)^{\mathrm{T}}\delta_s + \int_{-\delta_s/2v}^{\delta_s/2v}\sum_{j=1}^m\Big[\mathcal{K}\big(p_{i_j}(t_j+\tau)\big)\mathcal{K}\big(p_{i_j}(t_j+\tau)\big)^{\mathrm{T}} - \mathcal{K}(\mu_j)\mathcal{K}(\mu_j)^{\mathrm{T}}\Big]v\,d\tau. \quad (25)$$

By continuity of the Gaussian and the fact that maxeig$(\cdot)$ is a continuous function, we know that for any $\epsilon > 0$, there exists $\delta_s > 0$ such that

$$\text{maxeig}\Big\{\mathcal{K}\big(p_{i_j}(t_j+\tau)\big)\mathcal{K}\big(p_{i_j}(t_j+\tau)\big)^{\mathrm{T}} - \mathcal{K}\big(p_{i_j}(t_j)\big)\mathcal{K}\big(p_{i_j}(t_j)\big)^{\mathrm{T}}\Big\} < \epsilon$$

for all $\tau \in [-\delta_s/2v, \delta_s/2v]$. Noting that maxeig$(\cdot)$ is convex, we can use Jensen's inequality[5] to move the maxeig outside the sum and the integral to get

$$\text{maxeig}\Big\{\int_{-\delta_s/2v}^{\delta_s/2v}\sum_{j=1}^m\Big[\mathcal{K}\big(p_{i_j}(t_j+\tau)\big)\mathcal{K}\big(p_{i_j}(t_j+\tau)\big)^{\mathrm{T}} - \mathcal{K}(\mu_j)\mathcal{K}(\mu_j)^{\mathrm{T}}\Big]v\,d\tau\Big\} < m\delta_s\epsilon,$$

and taking the mineig of (25) and bounding the right hand side, we have

$$\text{mineig}\Big\{\sum_{j=1}^m\int_{\omega_j}\mathcal{K}(s)\mathcal{K}(s)^{\mathrm{T}}\,ds\Big\} > \text{mineig}\Big\{\sum_{j=1}^m \mathcal{K}(\mu_j)\mathcal{K}(\mu_j)^{\mathrm{T}}\Big\}\delta_s - m\delta_s\epsilon. \quad (26)$$

Finally, we show that $\sum_{j=1}^m \mathcal{K}(\mu_j)\mathcal{K}(\mu_j)^{\mathrm{T}} > 0$ as follows. Let $\mathcal{K}(\mu_j) = [k_{1j},\ldots,k_{mj}]^{\mathrm{T}}$, and we have $\mathcal{K}(\mu_j)\mathcal{K}(\mu_j)^{\mathrm{T}} = [k_{kj}k_{lj}]_{k,l}$ where $[\cdot]_{k,l}$ denotes the matrix with terms ranging over rows k and columns l. Therefore

$$\sum_{j=1}^m \mathcal{K}(\mu_j)\mathcal{K}(\mu_j)^{\mathrm{T}} = \Big[\sum_{j=1}^m k_{kj}k_{lj}\Big]_{k,l} = \Big[\mathcal{K}(\mu_k)^{\mathrm{T}}\mathcal{K}(\mu_l)\Big]_{k,l} = \Big[\mathcal{K}(\mu_1)\cdots\mathcal{K}(\mu_m)\Big]\Big[\mathcal{K}(\mu_1)\cdots\mathcal{K}(\mu_m)\Big]^{\mathrm{T}} = [k_{kl}]_{k,l}^2,$$

[5]Jensen's inequality states that $f(\int g(x)\,dx) \leq \int f(g(x))\,dx$ for any convex function $f(\cdot)$.

where the second equality comes from the fact that $k_{kj} = k_{jk}$ because the Gaussian is radially symmetric. From Theorem 2, we know that $[k_{kl}]_{k,l} > 0$. Denote the minimum eigenvalue of this matrix by $\lambda_{\min} = \text{mineig}\{[k_{k,l}]_{k,l}\} > 0$ to obtain $\text{mineig}\Big\{ \sum_{j=1}^{m} \mathcal{K}(\mu_j)\mathcal{K}(\mu_j)^{\mathrm{T}} \Big\} \geq \lambda_{\min}^2$. Finally, from (26) we have

$$\text{mineig}\Big\{ \sum_{j=1}^{m} \int_{\omega_j} \mathcal{K}(\mu_j)\mathcal{K}(\mu_j)^{\mathrm{T}}\, ds \Big\} > \lambda_{\min}^2 \delta_s - m\delta_s \epsilon,$$

and letting $\epsilon < \lambda_{\min}^2/m$ (since we can choose it to be arbitrarily small) we get the desired result. ■

Proof of Lemma 2: Following a similar line of reasoning as in [12], Lemmas 1 and 2, we show that (*i*) $\psi_1(t)$, $\psi_2(t)$, and $\psi_3(t)$ are continuous, (*ii*) their non-differentiabilities are only at isolated points in time (i.e. there are no pathological intervals of non-differentiability as in the Weierstrass function), and (*iii*) wherever their derivatives exist, those derivatives are uniformly bounded. These three facts imply that the functions are uniformly continuous.

(*i*) Firstly, note that $p_i(t)$, $\tilde{a}_i(t)$, $\Lambda_i(t)$, and $\lambda_{\epsilon_i}(t)$ are continuous, being the solutions to differential equations, and $C_i(P,t)$, $M_i(P,t)$, $F_i(P,t)$, and $L(P)$ are continuous because the Voronoi cells are continuous functions of the positions of the robots, and $\tilde{a}_i(t)$ is continuous. The dead-zone function $B_i^{\mathrm{dz}}(\cdot)$ is also continuous by design. Therefore the only discontinuous quantity in the system is in I_i^{proj}, appearing on the right hand side of $\dot{\tilde{a}}_i$. Neither I_i^{proj} nor $\dot{\tilde{a}}_i$ appear in any of $\psi_1(t)$, $\psi_2(t)$, or $\psi_3(t)$, hence we conclude that these are all continuous functions.

(*ii*) The non-differentiabilities in the system only apear in three varieties. The first variety is in the dead-zone $B_i^{\mathrm{dz}}(\cdot)$ when $C(\Lambda_i\tilde{a}_i+\lambda_{\epsilon_i})$ transitions to 0 which can only occur at an isolated time according to the differential equations defining Λ, $\tilde{a}_i$, and λ_{ϵ_i}. The second is in I_i^{proj} when one of the parameters $\hat{a}_{ij}$ transitions to the upper or lower limit, $a_{\max}$ or $a_{\min}$, which again can only happen at an isolated time due to the differential equation governing $\tilde{a}_i(t)$. The third variety occurs when a robot gains or lose Voronoi neighbors, causing a non-differentiability in $L(P)$, which can only occur at an isolated time due to the differential equation governing p_i.

(*iii*) The boundedness of the derivatives $|\dot{\psi}_k(t)| < \dot{\psi}_k^{\max}$, $k = 1, 2, 3$ can be established directly as follows. Firstly, it was shown in the proof of Theorem 1 that $\psi_k(t)$, $k = 1, 2, 3$ are all bounded, and $\hat{C}_i, p_i \in Q$ are bounded since Q is bounded, therefore $\dot{p}_i(t)$, $\Lambda_i(t)$, $\hat{M}_i(t)$, $\tilde{a}_i(t)$, $L(P)$, and $\lambda_{\epsilon_i}(t)$ are all bounded, as is $B_i^{\mathrm{dz}}(\Lambda_i\tilde{a}_i + \lambda_{\epsilon_i})$. This implies that $\dot{\tilde{a}}_i$ from (12) and (10) is bounded because it is a polynomial function of these quantities. Also $\mathcal{K}(q)$ is a Gaussian function, hence it is bounded, so $\mathcal{K}_i(t)$ is bounded, and $\dot{\Lambda}_i = \mathcal{K}_i\mathcal{K}_i^{\mathrm{T}}\|\dot{p}_i\|$, $\dot{\lambda}_{\epsilon_i} \leq \mathcal{K}_i\|\dot{p}_i\|\phi_{\epsilon_{\max}}$, and $\beta = \mathcal{K}_i\|\dot{p}_i\|$ are all bounded, therefore $\dot{\psi}_3(t)$ is bounded. Since the length of the edge of a Voronoi cell l_{ij} changes as a bounded function of the robots' positions p_i and p_j, $\dot{L}(P)$ is bounded, therefore $\dot{\psi}_2(t)$ is bounded. Finally, $\dot{\hat{C}}_i$ and $\dot{\hat{M}}_i$ are bounded, as they are polynomial functions of the above quantities, and the $\hat{M}_i$ in the denominator of $\dot{\hat{C}}_i$ is bounded away from zero because $a_{\min} > 0$. Therefore $\dot{\psi}_1(t)$ is bounded. We have shown (*i*), (*ii*), and (*iii*), therefore we conclude that $\psi_k(t)$, $k = 1, 2, 3$ are uniformly continuous. ■

References

[1] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, April 2004.

[2] A. Weber, *Theory of the Location of Industries*. Chicago, IL: The University of Chicago Press, 1929, translated by Carl. J. Friedrich.

[3] Z. Drezner, *Facility Location: A Survey of Applications and Methods*, ser. Springer Series in Operations Research. New York: Springer-Verlag, 1995.

[4] S. Martínez, "Distributed interpolation schemes for field estimation by mobile sensor networks," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 419–500, March 2010.

[5] J. Cortés, S. Martínez, and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESIAM: Control, Optimisation and Calculus of Variations*, vol. 11, pp. 691–719, 2005.

[6] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *Proceedings of the IEEE Conference on Decision and Control*, Cancun, Mexico, December 2008.

[7] A. Breitenmoser, M. Schwager, J. C. Metzger, R. Siegwart, and D. Rus, "Voronoi coverage of non-convex environments with a group of networked robots," in *Proceedings of the International Conference on Robotics and Automation (ICRA 10)*, Anchorage, Alaska, USA, May 2010, pp. 4982–4989.

[8] C. Caicedo and M. Žefran, "Performing coverage on nonconvex domains," in *Proceedings of the 2008 IEEE Multi-Conference on Systems and Control*, 2008, pp. 1019–1024.

[9] A. Arsie and E. Frazzoli, "Efficient routing of multiple vehicles with no explicit communications," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 2, pp. 154–164, January 2007.

[10] J. Le Ny and G. J. Pappas, "Sensor-based robot deployment algorithms," in *Proceedings of the IEEE Conference on Decision and Control*. IEEE, 2010, pp. 5486–5492.

[11] ——, "Adaptive algorithms for coverage control and space partitioning in mobile robotic networks," University of Pennsylvania, ESE, Tech. Rep., 2010.

[12] M. Schwager, D. Rus, and J. J. Slotine, "Decentralized, adaptive coverage control for networked robots," *International Journal of Robotics Research*, vol. 28, no. 3, pp. 357–375, March 2009.

[13] P. Ioannou and P. V. Kokotovic, "Instability analysis and improvement of robustness of adaptive control," *Automatica*, vol. 20, no. 5, pp. 583–594, 1984.

[14] B. Peterson and K. Narendra, "Bounded error adaptive control," *IEEE Transactions on Automatic Control*, vol. 27, no. 6, pp. 1161 – 1168, December 1982.

[15] C. Samson, "Stability analysis of adaptively controlled systems subject to bounded disturbances," *Automatica*, vol. 19, no. 1, pp. 81 – 86, 1983.

[16] K. Narendra and A. Annaswamy, "A new adaptive law for robust adaptation without persistent excitation," *IEEE Transactions on Automatic Control*, vol. 32, no. 2, pp. 134 – 145, Feb. 1987.

[17] K. Tsakalis, "Robustness of model reference adaptive controllers: an input-output approach," *IEEE Transactions on Automatic Control*, vol. 37, no. 5, pp. 556 –565, May 1992.

[18] H. Choset, "Coverage for robotics—A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[19] D. T. Latimer IV, S. Srinivasa, V. Shue, S. S. adnd H. Choset, and A. Hurst, "Towards sensor based coverage with robot teams," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, May 2002, pp. 961–967.

[20] Z. J. Butler and D. Rus, "Controlling mobile sensors for monitoring events with coverage constraints," in *Proceedings of the IEEE International Conference of Robotics and Automation*, New Orleans, LA, April 2004, pp. 1563–1573.

[21] P. Ögren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Transactions on Automatic Control*, vol. 49, no. 8, pp. 1292–1302, August 2004.

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

[23] J. Choi, S. Oh, and R. Horowitz, "Distributed learning and cooperative control for multi-agent systems," *Automatica*, vol. 45, no. 12, pp. 2802–2814, 2009.

[24] M. Jadaliha, J. Lee, and J. Choi, "Adaptive control of multiagent systems for finding peaks of uncertain static fields," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 134, no. 5, pp. 051 007–051 007–8, 2012.

[25] N. Atanasov, J. Le Ny, and G. J. Pappas, "Distributed algorithms for stochastic source seeking with mobile robot networks," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 3, pp. 031 004–031 004–9, 2014.

[26] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte, "Information-theoretic control of multiple sensor platforms," in *Procroceedings of the IEEE International Conference on Robotics and Automation (ICRA 03)*, vol. 1, Taipei, Taiwan, September 2003, pp. 1521–1526.

[27] B. Grocholsky, "Information-theoretic control of multiple sensor platforms," Ph.D. dissertation, University of Sydney, 2002.

[28] G. M. Mathews, "Asynchronous decision making for decentralised autonomous systems," Ph.D., University of Sydney, March 2008.

[29] M. Schwager, P. Dames, D. Rus, and V. Kumar, "A multi-robot control policy for information gathering in the presence of unknown hazards," in *Proceedings of the International Symposium on Robotics Research (ISRR 11)*, Flagstaff, AZ, USA, August 2011.

[30] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[31] S. S. Sastry and M. Bodson, *Adaptive control: stability, convergence, and robustness*. Upper Saddle River, NJ: Prentice-Hall, Inc., 1989.

[32] J. J. E. Slotine and W. Li, *Applied Nonlinear Control*. Upper Saddle River, NJ: Prentice-Hall, 1991.

[33] R. Sanner and J. Slotine, "Gaussian networks for direct adaptive control," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 837–863, 1992.

[34] M. Schwager, M. P. Vitus, D. Rus, and C. J. Tomlin, "Robust adaptive coverage for robotic sensor networks," in *Proceedings of the International Symposium on Robotics Research (ISRR 11)*, Flagstaff, AZ, USA, August 2011.

[35] M. Schwager, D. Rus, and J. J. Slotine, "Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment," *International Journal of Robotics Research*, vol. 30, no. 3, pp. 371–383, March 2011.

[36] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[37] L. C. A. Pimenta, M. Schwager, Q. Lindsey, V. Kumar, D. Rus, R. C. Mesquita, and G. A. S. Pereira, "Simultaneous coverage and tracking (SCAT) of moving targets with robot networks," in *Proceedings of the Eighth International Workshop on the Algorithmic Foundations of Robotics (WAFR 08)*, Guanajuato, Mexico, December 2008.

[38] R. Duda, P. Hart, and D. Stork, *Pattern Classification*. New York: Wiley and Sons, 2001.

[39] I. Barbălat, "Systèmes d'équations différentielles d'oscillations non linéaires," *Revue de Mathématiques Pures et Appliquées*, vol. 4, pp. 267–270, 1959.

[40] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

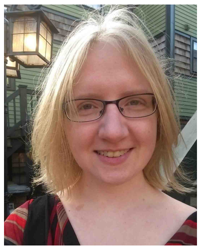
[41] M. Schwager, J. McLurkin, J. J. E. Slotine, and D. Rus, "From theory to practice: Distributed coverage control experiments with groups of robots," in *Experimental Robotics: The Eleventh International Symposium*, vol. 54. Springer-Verlag, 2008, pp. 127–136.

[42] C. Micchelli, "Interpolation of scattered data: distance matrices and conditionally positive definite functions," *Constructive Approximation*, vol. 2, no. 1, pp. 11–22, 1986.

[43] T. Hofmann, B. Schölkopf, and A. Smola, "Kernel methods in machine learning," *The annals of statistics*, pp. 1171–1220, 2008.

Mac Schwager is an assistant professor with the Aeronautics and Astronautics Department at Stanford University. He obtained his BS degree in 2000 from Stanford University, his MS degree from MIT in 2005, and his PhD degree from MIT in 2009. He was a postdoctoral researcher working jointly in the GRASP lab at the University of Pennsylvania and CSAIL at MIT from 2010 to 2012, and was an assistant professor at Boston University from 2012 to 2015. His research interests are in distributed algorithms for control, perception, and learning in groups of robots and animals. He received the NSF CAREER award in 2014.

Michael P. Vitus is a Software Engineer at Google developing visual inertial navigation algorithms for Project Tango. He received his Ph.D. from Stanford University in 2012, his M.S. from Stanford University in 2006, both in Aeronautics and Astronautics, and his B.S. in 2004 from Rensselaer Polytechnic Institute in Mechanical and Aeronautical Engineering. His research interests include simultaneous localization and mapping, sensor fusion, deterministic and stochastic motion planning, unmanned aerial vehicles, and optimization.



Samantha Powers received a B.S. in mechanical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2012. She was a software engineer with Microsoft from 2013 until 2015, and is now pursuing interests in 3D content creation and human computer interfaces.

Daniela Rus is the Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science and Director of the Computer Science and Artificial Intelligence Laboratory (CSAIL) at MIT. Rus research interests are in robotics, mobile computing, and data science. Rus is a Class of 2002 MacArthur Fellow, a fellow of ACM, AAAI and IEEE, and a member of the National Academy of Engineers. She earned her PhD in Computer Science from Cornell University. Prior to joining MIT, Rus was a professor in the Computer Science Department at Dartmouth College.

Claire J. Tomlin received the B.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, the M.Sc. degree in elec- trical engineering from Imperial College London, London, U.K., and Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA. She was an Assistant, an Associate, and a Full Professor with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, USA, from 1998 to 2007. She has held visiting researcher positions with the NASA Ames Research Center, Mountain View, CA, USA, and Honeywell International, Inc., Morristown, NJ, USA. She is currently the Charles A. Desoer Professor with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley. Her current research interests include hybrid control systems, with applications to air traffic systems, unmanned aerial vehicles, and systems biology. Dr. Tomlin was a recipient of the MacArthur Fellowship in 2006, the Okawa Foundation Research Grant in 2006, and the Eckman Award from the American Automatic Control Council in 2003.