Distributed Coverage Control with Sensory Feedback for Networked Robots.

Conference Paper · January 2006

Source: DBLP

3 authors, including:

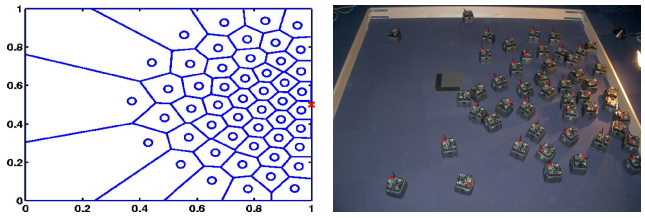
Mac Schwager
Boston University
62 PUBLICATIONS **1,277** CITATIONS
SEE PROFILE

James Mclurkin
Rice University
38 PUBLICATIONS **608** CITATIONS
SEE PROFILE

Distributed Coverage Control with Sensory Feedback for Networked Robots

Mac Schwager, James McLurkin, and Daniela Rus
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Lab
Cambridge, MA 02139 USA
Email: schwager@csail.mit.edu, jamesm@csail.mit.edu, and rus@csail.mit.edu

***Abstract*— This paper presents a control strategy that allows a group of mobile robots to position themselves to optimize the measurement of sensory information in the environment. The robots use sensed information to estimate a function indicating the relative importance of different areas in the environment. Their estimate is then used to drive the network to a desirable placement configuration using a computationally simple decentralized control law. We formulate the problem, provide a practical control solution, and present the results of numerical simulations. We then discuss experiments carried out on a swarm of mobile robots.**



(a) Numerical Simulation (b) Robot Swarm Experiment

Fig. 1. The control strategy was implemented in numerical simulation and on a swarm of mobile robots.

I. Introduction

We consider the problem of controlling networked groups of mobile robots in a decentralized fashion. Such mobile sensor networks promise the ability to collect information over distributed, large-scale domains with minimum infrastructure maintenance. This technology will enable scientific studies on geological and ecological scales previously beyond practical reach, and provide tools for a host of security and surveillance applications.

In this paper, we present a decentralized control law for producing increased robot density in areas of high importance and decreased density in areas of low importance. Specifically, we consider a group of robots that is dispatched over a bounded region of interest. The group's task is to sample a sensory function over the region. The sensory function is a scalar function unknown to the robots that indicates the relative importance of different areas in the region. Our solution composes an approximation of this function from sensory measurements. A decentralize control law then uses this approximation, as well as neighbor positions, to drive the robots to a configuration such that the sampling of the sensory function is near-optimal. This enables the network to record observations about the environment with varying resolution, so that areas with high values of the sensory function receive higher-density data observations than areas with low values.

A. Relationship to Prior Work

This problem belongs to the general category of coverage problems. Various strategies have been introduced for controlling coverage with networked mobile robots, and our work builds on several important results in this category. In [9], mobile sensing agents are controlled using potential functions for inter-agent interactions. Stability results are derived, but the optimality of the network configuration is not addressed. Similarly, in [2] an algorithm is proposed that allows for agents to concentrate in areas of high event density while maintaining area coverage constraints. The algorithm is proved to maintain sensor coverage for a limited case without addressing optimality. Most relevant to this paper is a body of results reported in [1], [5], and [4]. In this work decentralized control laws are derived for positioning mobile sensor networks optimally with respect to a known event probability density. This approach is advantageous because it guarantees that the network (locally) minimizes a cost function relevant to the coverage problem. However, the control strategy requires that each agent have a complete foreknowledge of the event probability density, thus it is not reactive to the sensed environment.

Our control strategy is an extension to the one reported in [5]. We re-interpret the problem in a non-probabilistic framework, and derive a local control law which requires that each agent can measure the value and gradient of a sensory function. In contrast to [5], the robots do not require foreknowledge of this function. Instead, the robots approximate the function from sensor measurements while maintaining or seeking a near-optimal sensing configuration. Also in contrast to [5], our function approximation yields a control law with a computationally efficient closed form. This eliminates the need for numerical integration of an arbitrary function over a polygonal domain at every time step. The control law results in near-optimal, as opposed to optimal, performance, though the difference is shown to be negligible in practice. We demonstrate the effectiveness and computational simplicity of

the algorithm in numerical simulations and in experiments on swarms of 45-50 robots (see Figure 1).

II. Problem Formulation

In this section, we build a function representing the sensing cost associated with a network configuration. A network is said to have optimal coverage if it minimizes this cost function over the region of interest. Following standard results in the field [5], we show that all configurations of a certain type, namely centroidal Voronoi configurations, correspond to local minima of the cost function.

The sensor network consists of a group of identical robots, each with some degree of mobility and the capacity for measuring a sensory function from the environment. The sensory function indicates the relative importance of different areas in the environment. It may be a quantity that is sensed directly, such as temperature, or it may demand more elaborate processing of sensory data, such as would be required to detect the concentration of a chemical, or the intensity of sound of a particular frequency. [1] In addition, we assume that a robot can measure the positions of its Voronoi neighbors relative to itself, and that it can detect the boundaries of the region of interest. We review the formalism introduced in [5] to rigorously model the scenario described above.

Let there be n robots in a known, convex polytope $Q \subset \Re^N$. An arbitrary point in Q is denoted q, the position of the i^{th} robot is denoted p_i, and the set of all robot positions is denoted $P = \{p_1, ..., p_n\}$. Let $\mathcal{W} = \{W_1, ..., W_n\}$ be a partition of Q such that one robot at position p_i lies within each region W_i. Define the sensory function, $\phi(q)$, as a scalar field with continuous first derivatives over Q. The function $\phi(q)$ is not known by the robots in the network.

Let the *unreliability* of the sensor measurement be defined by a function $f(x)$ which is strictly increasing. Specifically, $f(\|q - p_i\|)$ describes how unreliable is the measurement of the information at q by a sensor at p_i (henceforth, $\|.\|$ is used to denote the ℓ^2-norm). This form of $f(x)$ is physically appealing since it is reasonable that sensing will become more unreliable farther from the sensor. We will use $f(x)$ as a tunable parameter to influence the sensitivity of the robots to their sensor readings.

We can formulate the cost incurred by one robot sensing over one region W_i as

$$h_i(p_i, W_i) = \int_{W_i} f(\|q - p_i\|)\phi(q)dq.$$

Notice that unreliable sensing is expensive and high values of $\phi(q)$ are also expensive. Summing over all robots, a function representing the overall sensing cost of a given network configuration can be written

$$\mathcal{H}(P, \mathcal{W}) = \sum_{i=1}^{n} \int_{W_i} f(\|q - p_i\|)\phi(q)dq. \qquad (1)$$

An optimal network configuration corresponds to a particular pair $(P, \mathcal{W})$ which minimizes (1).

To solve this minimization problem, we must introduce the notion of a Voronoi partition. The Voronoi region, V_i, of a given robot is the region of points that are closer to that robot than to any other, that is

$$V_i = \{q \in Q \mid \|q - p_i\| \leq \|q - p_j\|, \forall j \neq i\}.$$

The division of an area into such regions is called a Voronoi partition, denoted $\mathcal{V}(P)$, and is a function of the robot positions. We will use the shorthand $\mathcal{H}_\mathcal{V}(P) = \mathcal{H}(P, \mathcal{V}(P))$.

Because the function $f(x)$ is strictly increasing, the Voronoi partition, $\mathcal{V}$, minimizes the cost function, $\mathcal{H}$, for any fixed configuration, P, of robots. This is clear since, for an arbitrary point $q \in Q$, $q \in V_i$ gives the smallest value of $f(\|q - p_i\|)$ over i, and therefore the smallest contribution to $\mathcal{H}$. Thus we have

$$\min_{P,\mathcal{W}} \mathcal{H} = \min_{P} \mathcal{H}_\mathcal{V}.$$

To find local minima of $\mathcal{H}_\mathcal{V}$, we examine solutions to the expression

$$\nabla \mathcal{H}_\mathcal{V} = [\cdots \frac{\partial \mathcal{H}_\mathcal{V}}{\partial p_i}^T \cdots]^T = \mathbf{0}.$$

It is clear that each partial derivative must be zero for a local minimum. Applying a multi-variable generalization of Leibniz Rule, [2] we can move the differentiation inside the integral sign in (1) to get

$$\begin{aligned} \frac{\partial \mathcal{H}_\mathcal{V}}{\partial p_i} &= \int_{V_i} \frac{\partial f(\|q - p_i\|)}{\partial p_i}\phi(q)dq + \\ &\sum_{j \in \mathcal{N}_i} \int_{\partial V_j} f(\|q - p_i\|)\phi(q)\frac{\partial \partial V_j}{\partial p_i} n_j dq + \\ &\int_{\partial V_i} f(\|q - p_i\|)\phi(q)\frac{\partial \partial V_i}{\partial p_i} n_i dq, \end{aligned} \qquad (2)$$

where ∂V_i denotes the boundary of the region V_i, $n_i(q)$ denotes the outward facing normal of ∂V_i, and $\mathcal{N}_i$ is the set of indices of the neighbors of p_i, excluding i itself. Note that all the integrals in (2) are $N \times 1$ vectors since $\frac{\partial \partial V_i}{\partial p_i}$ is an $N \times N$ matrix and n_i is an $N \times 1$ vector. We assert that the last two terms of (2), in fact, sum to zero. A proof can be outlined as follows. Since p_i only affects ∂V_j at the shared boundary of V_j and V_i, we have that

$$\bigcup_{j \in \mathcal{N}_i} \partial V_j = \partial V_i.$$

Also, an inward normal, $-n_i$, for V_i is equal to an outward normal, n_j, for any of its neighbors V_j, at the boundary which they share. This leads to

$$\begin{aligned} \sum_{j \in \mathcal{N}_i} \int_{\partial V_j} f(\|q - p_i\|)\phi(q)\frac{\partial \partial V_j}{\partial p_i} n_j dq &= \\ -\int_{\partial V_i} f(\|q - p_i\|)\phi(q)\frac{\partial \partial V_i}{\partial p_i} n_i dq, \end{aligned}$$

[1] In contrast, Cortés et al [5] use a probability density function describing the likelihood of an event occurring in a particular area.

[2] This procedure is known in fluid mechanics as the Reynolds Transport Theorem.

giving the desired result.

Using this fact, (2) can simply be written

$$\frac{\partial \mathcal{H}_\mathcal{V}}{\partial p_i} = \int_{V_i} \frac{\partial f(\|q-p_i\|)}{\partial p_i}\phi(q)dq.$$

We can evaluate the partial derivative of $f(x)$ using the chain rule, and move p_i outside of the integral to give

$$\begin{aligned}\frac{\partial \mathcal{H}_\mathcal{V}}{\partial p_i} &= -\int_{V_i} \frac{q}{\|q-p_i\|}\left.\frac{df(x)}{dx}\right|_{\|q-p_i\|}\phi(q)dq + \\ &\quad p_i\int_{V_i} \frac{1}{\|q-p_i\|}\left.\frac{df(x)}{dx}\right|_{\|q-p_i\|}\phi(q)dq \qquad (3)\end{aligned}$$

Next we define two properties analogous to mass-moments of rigid bodies. The mass of V_i is defined as

$$M_{V_i} = \int_{V_i} \frac{1}{\|q-p_i\|}\left.\frac{df(x)}{dx}\right|_{\|q-p_i\|}\phi(q)dq, \qquad (4)$$

and the centroid of V_i is defined as

$$C_{V_i} = \frac{1}{M_{V_i}}\int_{V_i} \frac{q}{\|q-p_i\|}\left.\frac{df(x)}{dx}\right|_{\|q-p_i\|}\phi(q)dq. \qquad (5)$$

Note that $f(x)$ strictly increasing and $\phi(q)$ strictly positive imply both $M_{V_i} > 0 \;\forall\; V_i \neq \{\emptyset\}$ and $C_{V_i} \in V_i \backslash \partial V_i$ (C_{V_i} is in the interior of V_i). Thus M_{V_i} and C_{V_i} have properties intrinsic to physical masses and centroids. Substituting these quantities into (3) gives

$$\frac{\partial \mathcal{H}_\mathcal{V}}{\partial p_i} = -M_{V_i}(C_{V_i} - p_i). \qquad (6)$$

Equation (6) implies that local minima of $\mathcal{H}_\mathcal{V}$, and therefore $\mathcal{H}(P, \mathcal{W})$, correspond to the configurations, P, such that $p_i = C_{V_i}\ \forall i$, that is, each agent is located at the centroid of its Voronoi region. We will denote the set of all such centroidal Voronoi configurations as P_C. Thus, the optimal coverage task is to drive the group of robots to a centroidal Voronoi configuration, $P \in P_C$.

III. Estimated Error Feedback Control

We will design a control law to take advantage of the surprisingly simple result in (6). The control law will drive the network to an *estimated* centroidal Voronoi configuration using sensory data available to the robots to form an on-line approximation of the centroids of their Voronoi regions.

Assume that the dynamics of each robot can be modeled by the first-order equation

$$\dot{p}_i = u_i, \qquad (7)$$

where u_i is the control input. This might mean simply that a low-level controller is in place to enforce first-order dynamics. Next, prescribe the linear proportional control law

$$u_i = k(\hat{C}_{V_i} - p_i), \qquad (8)$$

where $\hat{C}_{V_i}$ is an estimate of C_{V_i} based on the information available to robot i. To investigate the stability of such a control law we propose to use $\mathcal{H}_\mathcal{V}(P)$ as a Lyapunov-like function. Taking the time derivative of $\mathcal{H}_\mathcal{V}(P)$ along the trajectories of (7) gives

$$\dot{\mathcal{H}}_\mathcal{V} = \sum_i \frac{\partial \mathcal{H}_\mathcal{V}}{\partial p_i}^T \dot{p}_i,$$

and using (6) and (8) we get

$$\dot{\mathcal{H}}_\mathcal{V} = -k\sum_{i=1}^{n} M_{V_i} e_i^T \hat{e}_i, \qquad (9)$$

where $e_i = (C_{V_i} - p_i)$, and $\hat{e}_i = (\hat{C}_{V_i} - p_i)$. The actual error, e_i, is unknown. Notice, however, that if an estimate, $\hat{e}_i$, can be found such that the inner product of the two errors is positive, convergence of $\hat{e}_i$ to zero will be guaranteed. Geometrically, this means that the angle between $\hat{e}_i$ and e_i must remain less than $\pi/2$ *rad* for all time. We use this insight to design a centroid estimate, $\hat{C}_V$, using only sensed information local to robot i.

Henceforth we will deal with the specific case in which $f(x) = 1/2x^2$. This causes the factor

$$\frac{1}{\|q-p_i\|}\left.\frac{df(x)}{dx}\right|_{\|q-p_i\|}$$

to become unity, making the proceeding expressions more transparent. The method presented, however, is valid for any strictly increasing $f(x)$ with smooth first derivatives.

A. Control Using Linear Approximations

Consider a situation in which the values of the sensory function, $\phi(p_i)$, and its gradient, $\nabla\phi\ |_{p_i}$, are available continuously to robot i at its current position. In this case, the available information motivates a linear estimate of C_{V_i} over the known region V_i. We define the linear approximation to C_{V_i} calculated from an agent at position p_i as

$$\hat{C}_{V_i} = \frac{1}{\hat{M}_{V_i}}\int_{V_i^+} q\hat{\phi}_i(q)dq, \qquad (10)$$

where

$$\hat{M}_{V_i} = \int_{V_i^+} \hat{\phi}_i(q)dq, \qquad (11)$$

$$V_i^+ = V_i \cap \{q \mid \hat{\phi}_i(q) \geq 0\}, \text{ and} \qquad (12)$$

$$\hat{\phi}_i(q) = \phi(p_i) + \nabla\phi^T\ |_{p_i}\ (q - p_i). \qquad (13)$$

The above formulation follows naturally from the definition of C_{V_i} and M_{V_i} in (5) and (4). The integrals are taken over V_i^+ to avoid calculating values of $\hat{M}_{V_i} \leq 0$ and values of $\hat{C}_{V_i}$ outside of the region V_i.

We can prove the stability of the proposed controller in the case of linear $\phi(q)$. In this case the function $\phi(q)$ is fully parameterized by its value and gradient as measured at any point, therefore $\hat{\phi}_i(q) = \phi(q)$. This special case becomes mathematically equivalent to that treated in [5]. Then $\hat{C}_{V_i} = C_{V_i}$ and from (9) we have that

$$\dot{\mathcal{H}}_\mathcal{V} = -k\sum_{i=1}^{n} M_{V_i} e_i^T e_i.$$

As was noted previously, $M_{V_i} > 0$, therefore $\dot{\mathcal{H}}_{\mathcal{V}} \leq 0 \ \forall p_i$, and $\dot{\mathcal{H}}_{\mathcal{V}} = 0$ iff $p_i = C_{V_i} \ \forall i$. Additionally, P_C is the largest invariant set since, from (7) and (8), $\dot{p}_i = 0 \ \forall i \leftrightarrow P \in P_C$. Then by LaSalle's theorem $\lim_{t\to\infty} p_i = C_{V_i} \ \forall i$ which implies $P \to P_C$.

In the case of nonlinear $\phi(q)$, we observe that the control law with a linear estimation causes the robots to converge to the *estimated* centroid, $\hat{C}_{V_i}$, of their Voronoi region. We call such configurations near-optimal. It is difficult to bound the error between the estimated centroid and the actual centroid in the general N-dimensional case above. It is however possible to do so for 1-dimensional systems, where robots can move along an arbitrary curvilinear segment in three-space (e.g. a robot moving along a track.)

B. Efficient Computation of Integrals

We will use the convenient form of the centroid estimation above to derive an *analytical* solution to the centroid integral. The analytical expression will make the control law feasible for robot platforms with limited computational resources. This eliminates the need to descretize the Voronoi region and approximate the $\hat{M}_{V_i}$ and $\hat{C}_{V_i}$ integrals in a computationally-expensive numerical procedure.

Since the estimated $\hat{\phi}$ is polynomial in q, we can use the results from [3] to find the integrals for $\hat{M}_{V_i}$ and $\hat{C}_{V_i}$ as polynomials in the vertices of the Voronoi region V_i. First, from (10), (11), and (13) we can divide the integral expression for $\hat{C}_{V_i}$ and $\hat{M}_{V_i}$ into monomial components to get

$$\hat{C}_{V_i} = \frac{1}{\hat{M}_{V_i}} \left(\phi(p_i) - \nabla\phi^T \mid_{p_i} p_i\right) \int_{V_i^+} q dq + \frac{1}{\hat{M}_{V_i}} \int_{V_i^+} q^T q dq \nabla\phi \mid_{p_i}, \tag{14}$$

where

$$\hat{M}_{V_i} = \left(\phi(p_i) - \nabla\phi^T \mid_{p_i} p_i\right) \int_{V_i^+} dq + \nabla\phi^T \mid_{p_i} p_i \int_{V_i^+} q dq. \tag{15}$$

These expressions can be simplified further by introducing the constants $c_1 = \left(\phi(p_i) - \nabla\phi^T \mid_{p_i} p_i\right)$, and $\begin{bmatrix} c_1 & c_2 \end{bmatrix}^T = \nabla\phi \mid_{p_i}$, and by defining a general integral of a monomial over a polygon as

$$\mathcal{I}_{V_i^+}^{\alpha\beta} = \iint_{V_i^+} x^\alpha y^\beta dx dy, \tag{16}$$

where $q = \begin{bmatrix} x & y \end{bmatrix}^T$. Then we can write (14) and (15) as

$$\hat{C}_{V_i} = \frac{c_1}{\hat{M}_{V_i}} \begin{bmatrix} \mathcal{I}_{V_i^+}^{10} \\ \mathcal{I}_{V_i^+}^{01} \end{bmatrix} + \frac{1}{\hat{M}_{V_i}} \begin{bmatrix} \mathcal{I}_{V_i^+}^{20} & \mathcal{I}_{V_i^+}^{11} \\ \mathcal{I}_{V_i^+}^{11} & \mathcal{I}_{V_i^+}^{02} \end{bmatrix} \begin{bmatrix} c_2 \\ c_3 \end{bmatrix}, \tag{17}$$

where

$$\hat{M}_{V_i} = \mathcal{I}_{V_i^+}^{00} c_1 + \begin{bmatrix} \mathcal{I}_{V_i^+}^{10} & \mathcal{I}_{V_i^+}^{01} \end{bmatrix} \begin{bmatrix} c_2 \\ c_3 \end{bmatrix}. \tag{18}$$

The solution of the integral in (16) is given in analytical form by equation (4) from [3]. The cases specifically required for the computation of (17) are simplified and enumerated below:

$$\begin{aligned}
\mathcal{I}_{V_i^+}^{00} &= \frac{1}{2}\sum_{i=1}^{m}(y_{i+1}-y_i)(x_{i+1}+x_i), \\
\mathcal{I}_{V_i^+}^{01} &= \frac{1}{6}\sum_{i=1}^{m}[(x_i - x_{i+1})(y_{i+1}^2 + y_{i+1}y_i + y_i^2) + \\
&\quad 3(x_{i+1}y_{i+1}^2 - x_i y_i^2)], \\
\mathcal{I}_{V_i^+}^{10} &= \frac{1}{6}\sum_{i=1}^{m}(y_{i+1}-y_i)(x_{i+1}^2 + x_{i+1}x_i + x_i^2), \\
\mathcal{I}_{V_i^+}^{11} &= \frac{1}{24}\sum_{i=1}^{m}[(y_{i+1}-y_i)(2x_{i+1}x_i(y_{i+1}+y_i) + \\
&\quad x_{i+1}^2(3y_{i+1}+y_i) + x_i^2(y_{i+1}+3y_i))], \\
\mathcal{I}_{V_i^+}^{02} &= \frac{1}{12}\sum_{i=1}^{m}[(x_i - x_{i+1})(y_{i+1}^3 + y_{i+1}^2 y_i + \\
&\quad y_{i+1}y_i^2 + y_i^3) + 4(x_{i+1}y_{i+1}^3 - x_i y_i^3)], \\
\mathcal{I}_{V_i^+}^{20} &= \frac{1}{12}\sum_{i=1}^{m}(y_{i+1}-y_i)(x_{i+1}^3 + x_{i+1}^2 x_i + \\
&\quad x_{i+1}x_i^2 + x_i^3),
\end{aligned}$$

where m is the number of vertices, $\begin{bmatrix} x_i & y_i \end{bmatrix}^T$, of V_i^+, and where the index $m+1$ is interpreted as 1. The vertices must be ordered counter-clockwise around the perimeter of V_i^+.

The expression in (17) with the associated expressions for the integral terms can be computed directly to give the actual value of $\hat{C}_{V_i}$ and $\hat{M}_{V_i}$.

C. Practical Algorithms

A practical method for implementing the proposed control law on a network of robots is detailed in Algorithm 1. We assume that the robots have access to a procedure for obtaining their own Voronoi region. Several such algorithms exist, for example those given in [5], [6]. In our hardware implementation, we use the Delaunay computation from [7] and build Voronoi regions using knowledge of the Delaunay neighbors.

This algorithm is decentralized, fully distributed, and requires minimal communication between neighboring robots. It can be used on teams of large robots, on teams of small robots such as [8], or on mobile sensor network nodes with limited computation and storage capabilities such as the mobile Mica Motes described by [10].

IV. Numerical Simulations

A. Implementation

Simulations were carried out in a Matlab environment. The dynamics in (7) with the control law in (8) for a group of robots were modeled as a system of coupled differential equations. Voronoi computation was carried out in a centralized fashion using standard Matlab Voronoi functions. The centroid was calculated using the analytical solution to the centroid

Algorithm 1 The Coverage Control Algorithm

Require: Each robot can compute its local Voronoi region
Require: Each robot's sensor can measure $\phi(p_i)$ and $\nabla\phi\,|_{p_i}$
Require: Each robot can sense the location of the boundary of the space of interest, Q
  loop
    Measure coordinates of neighboring robots
    Compute local Voronoi region, V_i
    Measure $\phi(p_i)$ and $\nabla\phi\,|_{p_i}$
    Truncate V_i to get V_i^+
    Evaluate analytical expression in (17) to compute centroid approximation, $\hat{C}_{V_i}$
    Apply control input $u_i = k(\hat{C}_{V_i} - p_i)$
  end loop

integral given in (17). A custom, fixed-time-step numerical solver was used to integrate the equations of motion of the group of robots. The sensory function, $\phi(q)$, was built from Gaussians. Specifically, two cases were investigated. For the first case, the function was chosen as

$$\phi(q) = \frac{\gamma}{\sigma\sqrt{2\pi}}\left(e^{-\frac{(q-\mu_1)^2}{2\sigma^2}} + e^{-\frac{(q-\mu_2)^2}{2\sigma^2}}\right). \qquad (19)$$

A function with multiple maxima was used to illustrate the robustness of the control scheme to complicated sensing environments. For the second case, a single Gaussian was used of the form

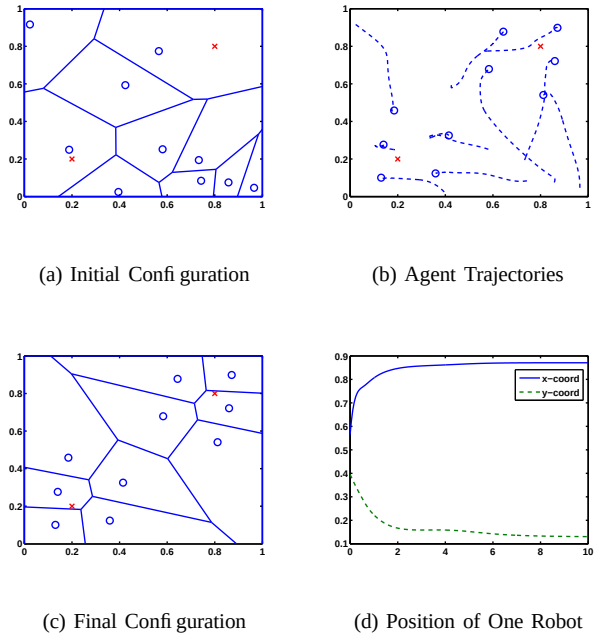
$$\phi(q) = \frac{\gamma}{\sigma\sqrt{2\pi}}e^{-\frac{(q-\mu_1)^2}{2\sigma^2}}. \qquad (20)$$

This $\phi(q)$ was chosen to gather statistical data about convergence properties over a number of runs. The parameters used for the Gaussian functions were $\gamma = 1$, $\sigma = 1/\sqrt{2}$, $\mu_1 = (.2, .2)$, and $\mu_2 = (.8, .8)$. The control gain was $k = 5$, the region Q was set to be a square of 1 meter on each side, and q was a Cartesian point $[\ x \quad y\ ]^T \in Q$.

B. Results

Figure 2 shows the results of a typical simulation run with the $\phi(q)$ given by (19). The initial configuration of the network is shown in Figure 2(a), the trajectories of the agents (dashed lines) in Figure 2(b), and the final configuration in Figure 2(c). The centers of the Gaussian functions, μ_1 and μ_2, are marked with x's. In Figure 2(d), the time evolution of the x and y coordinates of one robot are shown. The performance of the control scheme is clearly demonstrated in the simulation.

To assess convergence properties of the algorithm, statistical results were compiled over a number of simulation runs with random initial configurations using the $\phi(q)$ given by (20). Three batch runs were executed: one batch of 1000 simulation runs with 10 robots, one of 100 simulation runs with 100 robots, and one of 10 simulation runs with 1000 robots. Each simulation run was said to converge if the mean normed error of the group of robots was found to be less than 1×10^{-7}m. The mean and standard deviations of the convergence times are shown in Table I.



(a) Initial Configuration (b) Agent Trajectories

(c) Final Configuration (d) Position of One Robot

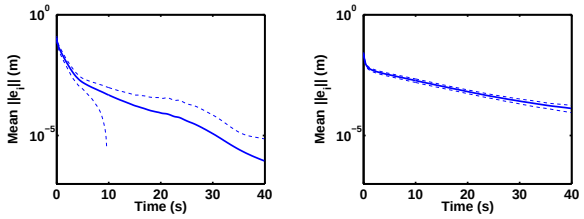
Fig. 2. The initial configuration of the network is shown in 2(a), the trajectories of the agents (dashed lines) in 2(b), and the final configuration in 2(c). The Gaussian centers of $\phi(q)$ are marked by x's. In 2(d), the time evolution of the x and y coordinates of one agent are shown.

	Mean (s)	Standard Deviation (s)
10 Robots 1000 Trials	9.56	3.18
100 Robots 100 Trials	25.10	1.23
1000 Robots 10 Trials	19.33	2.14

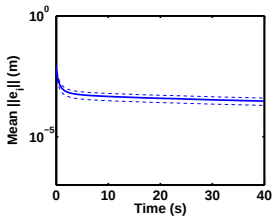
TABLE I
MEAN AND STANDARD DEVIATION OF CONVERGENCE TIMES

It is interesting to note that convergence time does not necessarily increase with the number of robots. This is due to the decentralized nature of the control law. The data suggests, in fact, that there is a certain network size that maximizes the convergence time, while larger or smaller values result in faster convergence. One can interpret this effect as the result of two opposing factors. With few agents, global movement of the network can occur rapidly because the motion of one agent quickly propagates to influence all of the other agents. This propagation effect becomes more sluggish as the network size increases. However, with a large number of agents, there is a high likelihood that the initial position of any one agent is close to its final position. Thus each agent has less distance to cover. The push-and-pull of these two factors creates the maximum that is evidenced in the data.

The mean normed errors are shown in Figure 3 for all simulation runs for each of the three batches. The plots are given on semi-log axes to emphasize the near-exponential convergence rate of the closed loop system. Again, the fact that convergence time does not depend strongly on the size of the network is clear from the plots. Notice also that the variance

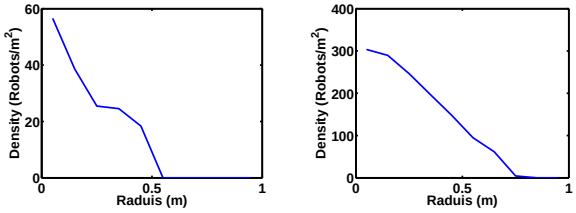


(a) 10 Robots, 1000 Trials (b) 100 Robots, 100 Trials

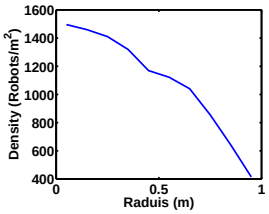


(c) 1000 Robots, 10 Trials

Fig. 3. The trajectories of the mean normed error of the robots are shown for 10, 100, and 1000 robots over 1000, 100, and 10 trials respectively. The plots are on log-log axes to show the near exponential error decay.



(a) 10 Robots, 1000 Trials (b) 100 Robots, 100 Trials



(c) 1000 Robots, 10 Trials

Fig. 4. The steady state density of robots as a function of distance from the Gaussian center are shown for the three batch runs.

appears to decrease with increasing network size. This is a statistical side-effect. Because the plots show mean normed error, a large network will be more likely to have a mean close to the ensemble mean than a small network.

The final distribution of the robots with respect to distance from the Gaussian center is shown for all three batches in Figure (4). The densities appear more jagged for smaller networks because there are fewer centroidal Voronoi configurations for smaller groups of robots. The densities are not precisely Gaussian, nor are they meant to be. But there are high concentrations of robots where the $\phi(q)$ is large–indicating an area of sensory interest–and low concentrations where $\phi(q)$ is small and there is little sensory interest.

V. Hardware Experiments

We implemented and tested the algorithm on groups of 45-50 robots. An incandescent and a halogen desk lamp were used simultaneously to create a sensory function $\phi(q)$. The robots used readings from light sensors and neighbor localization to carry out Algorithm 1.

A. The Swarm Hardware

Each "SwarmBot" (Figure 5) is autonomous and is equipped with bump sensors, light sensors, and an infra-red inter-robot communication and localization system called ISIS. The light sensors detect the sensory input. The ISIS inter-robot localization system provides local neighbor positions used to compute the Voronoi cells.

We limited the range of the localization system to about one meter to increase the depth of the network and minimize the number of inter-robot packet collisions. This algorithm

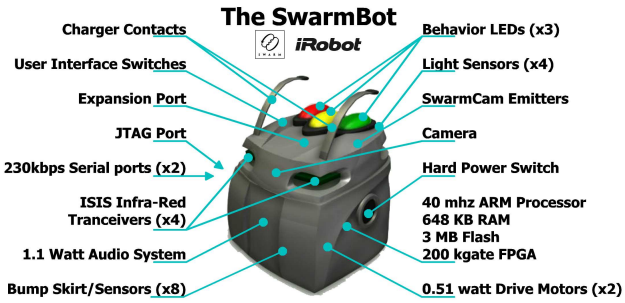


Fig. 5. The iRobot SwarmBot is designed for distributed algorithm development. Each SwarmBot has an infra-red localization and communication system called ISIS which enables nearby robots to communicate and determine the bearing, orientation, and range of their neighbors. An omni-directional bump skirt provides low-level obstacle avoidance. A 40 MHz 32-bit ARM Thumb microprocessor provides enough processing power for our algorithms.

uses no communications other than the localization messages, allowing us to test it with many robots in a small physical space.

B. Implementation of the Control Algorithm

The implementation of Algorithm 1 on the SwarmBot system required several modifications. The robots had a limited communications range of radius one meter, therefore the Delaunay neighbors computed by any robot were only those Delaunay neighbors within the one meter disk of the robot. These local constructions might not be the same as their global counterparts. We assumed that edges of greater than length one meter are uncommon in the triangulated graphs we consider and the effects of removing them have little impact on the

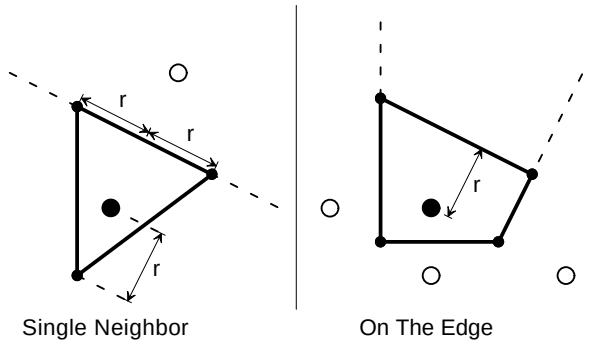


Fig. 6. The method used for closing the unbounded Voronoi region of a robot with 1 neighbor is shown on the left. The method for an unbounded Voronoi region of a robot with any two consecutive neighbors separated by more than π rad is shown on the right.

final result.[3]

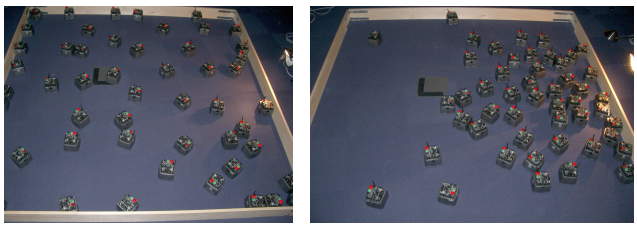
In addition, Algorithm 1 requires that each robot can detect and localize the boundaries of the region Q. This is necessary to prevent the Voronoi region of any robot from being unbounded. We did not implement boundary sensing on the SwarmBot. Instead, we developed three heuristic rules for truncating infinite Voronoi regions. Unbounded Voronoi regions occur in three distinct cases for robots on the boundary of the network. These are enumerated below along with the associated truncation technique.

1) *No Voronoi Neighbors:* The centroid is computed to be the robot's current position, thus the robot does not move until it acquires a neighbor.
2) *One Voronoi Neighbor:* A Voronoi region is constructed from an isosceles triangle of constant size whose base lies on the bisector between the two robots (see the left of Figure 6).
3) *Two Consecutive Voronoi Neighbors Separated by an angle $\geq \pi$:* A line is added perpendicular to the bisector of the infinite region at a constant distance from the robot (see the right of Figure 6). The two intersection points of this line with the unbounded Voronoi region are taken as vertices of the truncated Voronoi region.

Also, the robots' light sensors returned a bearing and distance. This was not enough information to compute $\nabla\phi\mid_{p_i}$ directly. To overcome this problem, we fixed the magnitude of $\nabla\phi\mid_{p_i}$ and determined its direction from the sensory stimulus. The signal strength $\phi(p_i)$ was measured directly.

Finally, the SwarmBot had a low-level controller in place which allowed the robot to move to a particular point relative to its current position. We used this position control as a substitute for velocity control. In particular, at each time instant, the input to the position controller was given as the estimated centroid value $\hat{C}_{V_i}$. The resulting control loop was identical to (8), but with an unknown, and unadjustable feedback gain k. This turned out not to be a difficulty since the fixed gain was well suited for the experiments.

[3]A Voronoi based control scheme with limited range communication was shown in [4] to have similar convergence properties as the case with unlimited communication range.



(a) Initial Configuration (b) Final Configuration

Fig. 8. The initial and final configurations for an experiment with 50 robots is shown.

The computational and memory requirements for this algorithm were small, and modification of the algorithm to run with integer calculations was straightforward. On the 40MHz ARM Thumb processors used in the SwarmBots, the memory usage for all the steps outlined in Algorithm 1, including the special cases discussed above, was 3949 bytes of code and 1284 bytes of RAM. With 8 neighbors, one cycle of the algorithm ran at 70ms, fast enough for real-time position updates.

C. Experiments

We documented the performance of the algorithm in a set of seven experiments. In each experiment, 45-50 robots were started randomly dispersed in a dark, $8' \times 8'$ workspace. One incandescent and one halogen desk lamp were turned on at the middle of one of the perimeter walls of the workspace. The robots were given two minutes to reach a final configuration, then their positions were photographed. Photographs of the final robot configurations are shown in Figure 7. Each of the seven experiments used a different sensor offset to induce varying degrees of clustering around the light source. If we define $\tilde{f}(x)$ as the natural response of the light sensors, we chose a DC offset, Φ to create a sensor function $f(x) = \Phi + \tilde{f}x$. For experiments 1–7 the Φ offsets were, 1600, 800, 400, 200, 100, 50, and 10 respectively.

A second set of experiments was carried out to quantify the final robot density as a function of distance from the light source. Three experiments were performed with a set-up identical to the one described above. Figure 8 shows the initial and final configurations for one of the three experiments. After two minutes, the distances from the light sources to each of the robots was measured. The density of robots calculates using these measurements is shown in Figure 9 with error bars. The density plots appear to match closely those found with numerical simulations.

VI. Conclusion

In this paper, a decentralized method for controlling coverage in mobile sensor networks was presented. The method is related to one proposed in [5], and introduces an innovation

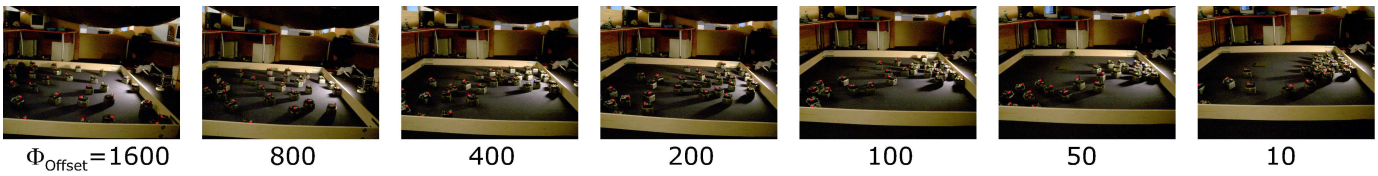


Fig. 7. The panels show the final positions of 40 robots using the proposed control law. The sensor offset, Φ, is labeled beneath each picture.

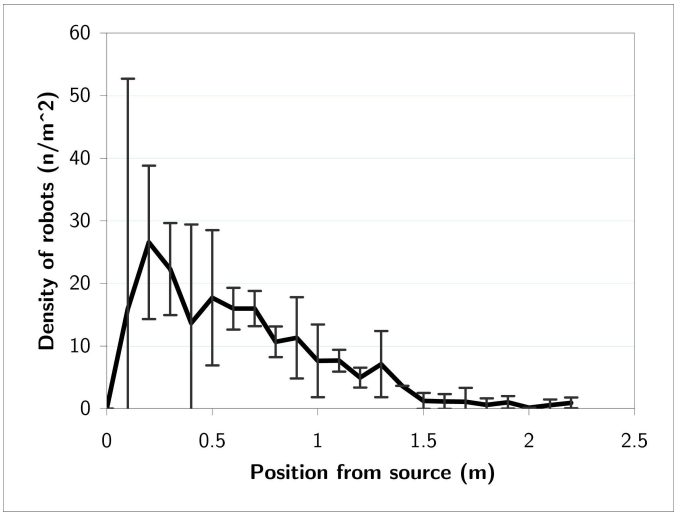


Fig. 9. The density of robots as a function of distance from the light source is shown with error bars for three experiments with 50 robots.

that allows for the network to adapt to unknown sensory environments. An analytical expression for centroid calculations was derived to make the algorithm computationally feasible on small robot platforms. The control scheme was demonstrated in numerical simulation and was shown to converge nearly exponentially, independent of the number of robots in the network. The control scheme was also implemented on a SwarmBot platform using integer calculations in a lean computing environment. The ability to implement the control scheme on such a platform emphasizes its practicality and minimalist nature. Robust performance was demonstrated in a number of experimental trials.

Potential extensions to this control method are numerous. For example, the approach in this work has used minimal communication among agents, only requiring localization to produce a Voronoi partition. If greater communication overhead is allowed, exciting possibilities for shared estimates of the sensory function become conceivable. For example, each robot might estimate the sensory function from a quadratic fit of its own and each of its Voronoi neighbors' measured values of $\phi(q)$. Theoretical, numerical, and experimental studies of this and other methods are ongoing.

Acknowledgment

This project was supported in part by the NSF, the MURI SWARM project, and Boeing. We are grateful for the support of all our sponsors.

References

[1] F. Bullo and J. Cortés. Adaptive and distributed coordination algorithms for mobile sensing networks. In V. Kumar, N. E. Leonard, and A. S. Morse, editors, *Cooperative Control. (Proceedings of the 2003 Block Island Workshop on Cooperative Control)*, volume 309 of *Lecture Notes in Control and Information Sciences*, pages 43–62. Springer Verlag, New York, 2005.

[2] Z. Butler and D. Rus. Controlling mobile sensors for monitoring events with coverage constraints. In *Proceedings of IEEE International Conference of Robotics and Automation*, pages 1563–1573, New Orleans, LA, April 2004.

[3] C. Cattani and A. Paoluzzi. Boundary integration over linear polyhedra. *Computer-Aided Design*, 22(2):130–135, 1990.

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

[5] J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, April 2004.

[6] Qun Li and Daniela Rus. Navigation protocols in sensor networks. *ACM Transactions on Sensor Networks*, 1(1):3–35, Aug. 2005.

[7] Xiang-Yang Li, Gruia Calinescu, and Peng-Jun Wang. Distributed construction of a planar spanner and routing for ad hoc wireless networks. In *Proceedings of the IEEE INFOCOM*, pages 1268–1277, New York, NY, June 2002.

[8] James McLurkin. Stupid robot tricks: A behavior-based distributed algorithm library for programming swarms of robots. Master's thesis, MIT, 2004.

[9] P. Ogren, E. Fiorelli, and N. E. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8):1292–1302, August 2004.

[10] Gabriel T. Sibley, Mohammad H. Rahimi, and Gaurav S. Sukhatme. Robomote: A tiny mobile robot platform for large-scale sensor networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.