From Theory to Practice: Distributed Coverage Control Experiments with Groups of Robots

Mac Schwager[1], James McLurkin[1], Jean-Jacques E. Slotine[2], and Daniela Rus[1]

[1] Computer Science and Artificial Intelligence Lab
MIT, Cambridge, MA 02139, USA
schwager@mit.edu,jamesm@csail.mit.edu,rus@csail.mit.edu

[2] Nonlinear Systems Lab
MIT, Cambridge, MA 02139, USA
jjs@mit.edu

Summary. A distributed algorithm is presented that causes a network of robots to spread out over an environment, while aggregating in areas of high sensory interest. The algorithm is a discrete-time interpretation of a controller previously introduced by the authors. The algorithmic implications of implementing this controller on a physical platform are discussed, and results are presented for 16 robots in two experiments. It is found that the algorithm performs well despite the presence of real-world complications.

1 Introduction

Robot group control is a fundamental problem for many robotics applications ranging from maintaining a desired formation, to achieving a desired network connectivity, to providing sensory coverage of an environment. We are interested in robot group control strategies that are (1) fully decentralized and distributed on the group, (2) adaptive to changes in the environment and the group, (3) provably convergent, and (4) experimentally feasible. In our previous work [13] we introduced a consensus-based controller for sensory coverage that meets the first three desiderata. In this paper we discuss the algorithmic implications of implementing this controller on a physical platform consisting of arbitrarily large groups of robots, and we present results of two experiments with 16 robots.

Specifically, we consider a distributed algorithm that causes a network of robots to spread out over an environment, while aggregating in areas of high sensory interest. It is known that a cost function representing the sensing cost of a robot network is locally minimized if each robot is positioned at the centroid of its Voronoi cell [6, 7]. It is not easy to position a robot at its Voronoi centroid because (1) the centroid moves as the robot moves, and (2) the robot cannot calculate its Voronoi centroid directly. Our algorithm uses consensus-based learning to estimate the centroid online. Then each robot "chases" its estimated centroid until it is eventually reached.

An overview of the algorithm is shown in Figure 1. Our algorithm could be used, for example, in search and rescue missions, environmental monitoring and clean-up, or automatic surveillance of rooms, buildings, or towns. The experimental results in this work demonstrate a significant step toward these practical applications.

There has been considerable theoretical and experimental work on multi-robot senor coverage problems in the past. One body of this work considers how to plan paths for groups of robots so that the sensor footprint of at least one robot visits every point in the environment. This body of work includes strategies with ant-like robots communicating through chemical trails [14], strategies that work on manifolds [2], and strategies that adapt known coverage algorithms for single robots to the multi-robot case [3, 8]. Voronoi tessellation have also been used to decompose the environment into easy to cover regions in [1, 9]. A thorough survey of this body of work can be found in [5].

In contrast, the objective of coverage control in our work is to disperse robots over an environment to do environmental monitoring or surveillance. This notion of coverage control has been explored with several approaches, for example [4, 6, 11]. We focus primarily on the framework introduced in [6] which poses the problem as an optimization of a cost function that has been well-studied in the facility placement literature [7]. However in [6] the robots are assumed to know before hand the distribution of sensory information in the environment. In our algorithm the robots are not assumed to know anything about the distribution of sensory information in the environment. Instead, the robots learn the distribution of sensory information while performing coverage. Our coverage algorithm can be thought of as proceeding in two complementary spaces, as described graphically in Figure 1. In position space, the robots perform coverage, while in a high-dimensional parameter space, the robots perform learning and consensus to collectively learn the distribution of sensory information in the environment.

This paper presents new experimental results with our coverage control algorithm using a group of 16 SwarmBots. We introduce the algorithm in a form that is practical for implementation on robot platforms with potentially limited computational resources. The algorithm is shown to operate in realistic situations in the presence of noise on sensor measurements and actuator outputs and with asynchronous execution among the robots in the group. We enumerate these complications in detail and discuss their apparent effect on the performance of the algorithm. Experimental results are analyzed with performance metrics to quantify the performance. In Section 2 we describe our algorithm and discuss complications relating to practical implementation. In Section 3 we give results of two experiments and show experimental snapshots. Conclusions and discussion are in Section 4.

2 Technical Approach

We model the robots as points moving in a plane. Specifically, consider a group of n robots in a convex, bounded area $Q \subset \mathbb{R}^2$. 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 $\{p_1, \ldots, p_n\}$

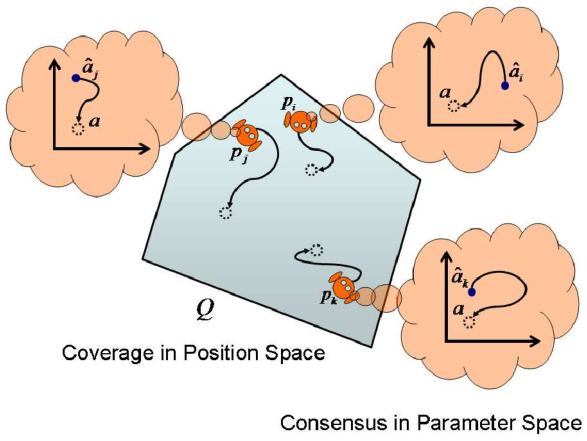


Fig. 1. A schematic of the overall control scheme is shown. The robots at p_i, p_j, and p_k move to cover the area Q. Simultaneously, each robot adapts a parameter vector ($\hat{a}_i, \hat{a}_j$, and $\hat{a}_k$) to make an estimate of the sensory environment. Neighboring robots average their parameter vectors at each step, causing all parameters to converge to a common vector.

is called the configuration of the network. Also, the set of robots that communicate to robot i is denoted $\mathcal{N}_i(t)$ and can change over time. Let $\{V_1, \ldots, V_n\}$ be the Voronoi partition of Q, for which the robot positions are the generator points. Specifically, $V_i = \{q \in Q \mid \|q - p_i\| \leq \|q - p_j\|, \forall j \neq i\}$.

Next, define the sensory function $\phi : Q \mapsto \mathbb{R}$, where $\phi(q) > 0 \; \forall q \in Q$. The sensory function $\phi(q)$ should be thought of as a weighting of importance over Q. We want to have many robots where $\phi(q)$ is large, and few where it is small. The function $\phi(q)$ is *not known* by the robots in the network, but the robots have sensors that can measure $\phi(p_i)$ at the robot's position p_i. The precise definition of the sensory function depends on the desired application. For example, for a human surveillance application in which robots use audio sensors, $\phi(q)$ may be chosen to be the intensity of the frequency range corresponding to the human voice. In an application in which a team of robots are used to clean up a chemical spill, an appropriate choice for the sensory function would be the concentration of the chemical as a function of position in the environment.

2.1 Sensory Function Approximation

Since the robots do not know $\phi(q)$, each one learns an approximation of $\phi(q)$ by expressing it as a linear combination of a set of known basis functions (or features) weighted by an unknown parameter vector. That is, robot i's approximation of the sensory function is given by

$$\hat{\phi}_i(q,t) = \mathcal{K}(q)^T \hat{a}_i(t), \tag{1}$$

where the vector of basis functions $\mathcal{K}(q) : Q \mapsto \mathbb{R}^m$ are fixed and are the same for all robots, but each robot has a potentially different parameter vector $\hat{a}_i(t) \in \mathbb{R}^m$ which changes in time. Figure 2 shows a graphical representation of this function approximation scheme. To learn the sensory function $\phi(q)$ the robots tune their parameter vector $\hat{a}_i(t)$ to make $\hat{\phi}_i(q,t)$ best match $\phi(q)$ given the measurements of their sensors. The way this tuning is done will be described in Section 2.2.

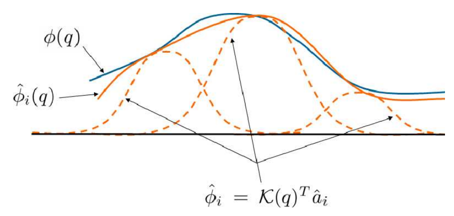


Fig. 2. The sensory function approximation is illustrated in this simplified 2-D schematic. The true sensory function is represented by ϕ (blue curve) and robot i's approximation of the sensory function is $\hat{\phi}_i$ (orange curve). The vector-valued function $\mathcal{K}(q)$ is shown as 3 Gaussians (dotted curves), and the parameter vector $\hat{a}_i$ denotes the weighting of each Gaussian.

Approximating a function in terms of parameters in the form of (1) is standard in many kinds of learning algorithms. In theory, the parameterization is not limiting since any function (with some smoothness requirements) over a bounded domain can be approximated arbitrarily well by a set of basis functions [12]. However, designing a suitable set of basis functions requires application-specific expertise. We use Gaussian basis functions in our experiments, but there is a variety of other basis function families to chose from including, wavelets, sigmoids, and splines. Gaussian basis functions have a computational advantage over non-local basis functions because they have nearly compact support. To compute the value of the network at a location $\hat{\phi}_i(q)$, or to tune the weights of the network $\hat{a}_i$ with new data, one has only to consider Gaussians in a region around the point of interest.

As described in Section 1, robots pursue the estimated centroid of their Voronoi region. The estimated centroid of the Voronoi region is its geometric center, weighted by the sensory function approximation. We calculate the discrete approximation of the centroid of V_i by dividing it up into a set of grid squares. Let the set of center points of the grid squares be $\bar{V}_i$ and each grid square has equal area Δq. Then the estimated centroid $\hat{C}_{V_i}$ of V_i, weighted by $\hat{\phi}(q)$, is given by

$$\hat{C}_{V_i}(t) = \frac{\sum_{q\in\bar{V}_i} q\hat{\phi}_i(q,t)\Delta q}{\sum_{q\in\bar{V}_i} \hat{\phi}_i(q,t)\Delta q}, \tag{2}$$

where $\hat{\phi}_i(q,t)$ is defined in (1).

2.2 Coverage Control Algorithm

The Coverage control algorithm has two components, corresponding to the two spaces described in Figure 1. In position space, the robots pursue their estimated centroids, given by

$$p_i(t+1) = C_{v_i}(t) \tag{3}$$

In parameter space, the robots collaboratively learn the function $\phi(q)$. They do this by iteratively integrating the values of $\phi(p_i)$ which they measure with their sensors into the quantity $\lambda_i(t)$. Simultaneously, they integrate the value of the basis function vector at their position $\mathscr{K}(p_i(t))$ into the quantity $\Lambda_i(t)$. Specifically,

$$\lambda_i(t+1) = \lambda_i(t) + \mathscr{K}(p_i(t))\phi(p_i(t)) \text{ and,} \tag{4}$$

$$\Lambda_i(t+1) = \Lambda_i(t) + \mathscr{K}(p_i(t))\mathscr{K}(p_i(t))^T. \tag{5}$$

The tuning of the parameter vector $\hat{a}_i(t)$ is greatly aided by forming a convex combination with the parameter vectors of robots in the neighbor set, which causes all robots' parameter vectors to approach a common value. This is what we call consensus learning. It has the effect of propagating every robot's sensor measurements around the network to be used by every other robot. Specifically, the robot tunes its parameter vector using

$$\begin{aligned}\hat{a}_{i_{\text{pre}}}(t) = \hat{a}_i(t) + \gamma\big(\lambda_i(t) - \Lambda_i(t)\hat{a}_i(t)\big) +& \\ \zeta \sum_{j\in\mathscr{N}_i(t)} \big(\hat{a}_j(t) - \hat{a}_i(t)\big).&\end{aligned} \tag{6}$$

where γ and ζ are positive gains. Then parameters are maintained above a predefined minimum positive value $a_{\min} \in \mathbb{R}$, $a_{\min} > 0$, using

$$\hat{a}_i(t+1) = \max(\hat{a}_{i_{\text{pre}}}(t), a_{\min}), \tag{7}$$

where the $\min(\cdot,\cdot)$ operates element-wise on the vector $\hat{a}_{i_{\text{pre}}}(t)$.

Both terms in (6) have an intuitive interpretation. The first term changes the parameter vector to decrease the difference between the current sensory function estimate, $\hat{\phi}(t)$, and all previous values of the sensory function measured with the robot's sensors, $\phi(p_i(\tau))$ for $\tau = 0, 1, \ldots, t$. The second term is the consensus coupling term, which causes all of the robots' parameters to reach a common value. We stress that a distributed implementation requires that each robot adapts its own parameter vector using local information available to it. If one were interested, instead, in designing a *centralized* adaptation law, one could simply use a common parameter vector that is adapted using the information from all robots. Our consensus-based coverage algorithm (as executed asynchronously by each robot) is written in Algorithm 1.

In summary, our coverage control algorithm integrates the sensor measurements and robot trajectory into $\lambda_i \in \mathbb{R}^m$ and $\Lambda_i \in \mathbb{R}^{m\times m}$, respectively. These are then used to tune the parameter vector $\hat{a}_i(t)$, which is also combined with the neighbors' parameter vectors. The parameter vector is used to calculate the sensory function estimate $\hat{\phi}_i(q,t)$, which is used to calculate the estimated Voronoi centroid $\hat{C}_{V_i}$, which the robot then moves toward.

The algorithm is a discrete-time interpretation of the control law from [13], which, under mild assumptions, was proved to cause robots to converge to the centroids of their Voronoi cells. By implementing the control algorithm on a group of

```
Algorithm 1 Consensus-Based Coverage
Require: Each robot knows its position $p_i(t)$
Require: Each robot can communicate with its Voronoi neighbors
Require: Each robot can compute its Voronoi cell, $V_i$
Require: Each robot can measure $\phi(p_i)$ with its sensors
  Initialize:
      $\Lambda_i(0) = 0,\ \lambda_i(0) = 0,\ \text{and}\ \hat{a}_i(0) = [a_{\min}, \dots, a_{\min}]^T$
  loop
    Update:
      $\lambda_i(t+1) = \lambda_i(t) + \mathscr{K}(p_i(t))\phi(p_i(t))$
      $\Lambda_i(t+1) = \Lambda_i(t) + \mathscr{K}(p_i(t))\mathscr{K}(p_i(t))^T$
      $\hat{a}_{i_{\text{pre}}}(t) = \hat{a}_i(t) + \gamma\big(\lambda_i(t) - \Lambda_i(t)\hat{a}_i(t)\big) + \zeta \sum_{j \in \mathscr{N}_i(t)} \big(\hat{a}_j(t) - \hat{a}_i(t)\big)$
    Project $\hat{a}_{i_{\text{pre}}}(t)$ to ensure parameters remain positive
      $\hat{a}_i(t+1) = \max(\hat{a}_{i_{\text{pre}}}(t), a_{\min})$
    Compute the robot's Voronoi region $V_i$
    Discretize $V_i$ into grid squares with area $\Delta q$ and center points $q \in \bar{V}_i$
    Compute the centroid estimate:
      $\hat{C}_{V_i}(t) = \frac{\sum_{q \in \bar{V}_i} q \hat{\phi}_i(q,t)\Delta q}{\sum_{q \in \bar{V}_i} \hat{\phi}_i(q,t)\Delta q}$,
      where $\hat{\phi}_i(q,t) = \mathscr{K}(q)^T \hat{a}_i(t)$
    Drive to the estimated centroid $p_i(t+1) = \hat{C}_{V_i}(t)$
  end loop
```

robots, we introduce a number of complications not considered in [13], as described in Table 1. The presence of noise in all measurement and actuation operations is a significant change from the noiseless scenario considered in [13]. Noise on the position measurements of neighbors in particular seemed to be a large source of error in the computation of the centroid of the Voronoi regions. We find that the algorithm performs well despite the presence of these real-world complications. The robustness of the algorithm can be attributed to its closed-loop structure, which constantly incorporates position updates and new sensor measurements to naturally correct mistakes. Also, the consensus-learning law tends to smooth the effects of noise on the sensory function measurements. This is because the parameter vectors are iteratively combined with neighbors' parameter vectors, so inaccuracies that might otherwise accumulate due to measurement errors are counteracted by measurement errors from neighboring robots.

Table 1. Algorithm 1 Vs. Controller from [13]

Algorithm 1	**Controller from [13]**
• Discrete-time difference equations	• Continuous-time differential equations
• Nonholonomic "unicycle" robot dynamics cause position errors and turning delays	• Holonomic "integrator" robot dynamics
• Asynchronous execution of instructions	• Synchronous evolution of equations
• Approximate Voronoi cells constructed from noisy measurements of neighbors within sensing range	• Exact Voronoi cells computed from exact positions of all Voronoi neighbors
• Discretized sums over the Voronoi cell	• Exact integrals over the Voronoi cell
• Noisy measurement of global position	• Exact knowledge of global position
• Noisy actuators	• Noiseless actuators
• Noisy measurement of sensory function	• Noiseless measurement of sensory function
• Basis function approximation cannot reconstruct exact sensory function	• Basis function approximation can reconstruct sensory function exactly with ideal parameter vector

3 Results and Experimental Snapshots

The algorithm was implemented using integer arithmetic on a network of 16 SwarmBots [10] (Figure 3). Each SwarmBot used an on-board IR system to sense relative neighbor positions (for computing its Voronoi cell) and to communicate its parameter vector for consensus learning. The robots moved in a square environment 2.44m$\times$2.44m. Each robot's global position was measured by an overhead camera and sent to it by radio. Each SwarmBot used a 40 MHz 32-bit ARM Thumb microprocessor, which provided enough processing power to execute our algorithm in real-time. There was no centralized or off-line processing.

We present the results of two experiments. In the first experiment in Section 3.1, the robots were given a noiseless measurement of a *simulated* sensory function $\phi(p_i)$. This allows us to compare the performance of the algorithm to a known ground truth. Since the function $\phi(q)$ is known, we also know the true position errors of the robots (the distances to their true centroids), as well as the true parameter errors. In the second experiment in Section 3.2, the robots use their on-board light sensors to sense light intensity in the environment as a sensory function. In this case we have no ground truth value for $\phi(q)$ so we can make no comparisons with ideal performance metrics. Instead we verify that the algorithm exhibits the behavior that one would expect given the scenario.

3.1 Simulated Sensory Function

The simulated sensory function, $\phi(q)$, was represented by two Gaussians, one in the lower right of the environment and one in the upper left. The set of basis functions of the function approximation was chosen to be 9 Gaussians arranged in a grid over the square environment. In particular, each of the nine components of $\mathscr{K}(q)$ was implemented as

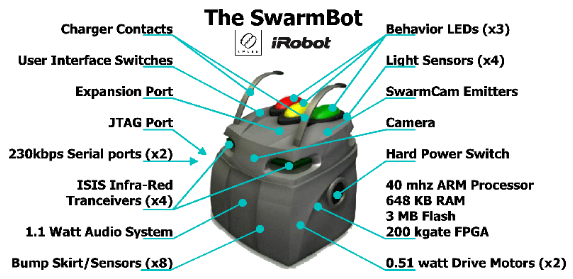


Fig. 3. In our experiments, we used the iRobot SwarmBot platform shown above.

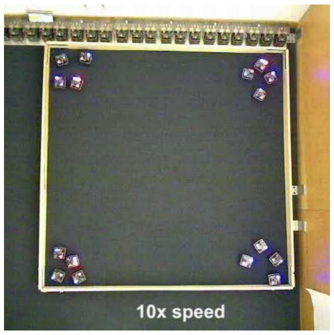
$$\frac{1}{2\pi\sigma^2}\exp\left\{-\frac{(q-\mu_j)^2}{2\sigma_j^2}\right\}, \tag{8}$$

where $\sigma_j = .37m$. The 2.44m$\times$2.44m square was divided into an even 3×3 grid and each μ_j was chosen so that one of the 9 Gaussians was centered at the middle of each grid square. The parameters for the simulated sensory function were chosen as $a = [200 \quad a_{\min} \quad \cdots \quad a_{\min} \quad 200]^T$, with $a_{\min} = 1$ so that only the upper left and lower right Gaussians contributed significantly to the value of $\phi(q)$, producing a bimodal distribution.

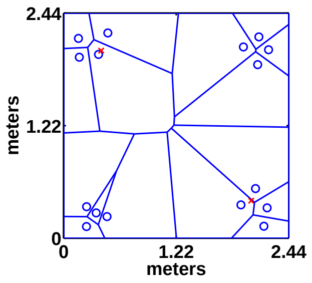
Figure 4 shows the positions of 16 robots over the course of an experiment. The algorithm caused the robots to group around the Gaussian peaks. The robots had no prior knowledge of the number or location of the peaks. Figure 5(a) shows the distance to the centroid, averaged over all the robots. The distance to the true centroid decreased over time to a steady value. The distance to the estimated centroid decreased to a value close to the pre-set dead zone of 5cm. The significant noise in the distance to the estimated centroid comes from the fact that the robots used their measured neighbor positions to compute their Voronoi cells. The IR sensing system used to measure the neighbor positions had a considerable amount of uncertainty. This caused the Voronoi cells to change rapidly, which in turn caused the centroid estimates to be noisy. The fact that the true distance to the centroid decreased steadily is evidence that the algorithm is robust to these significant sources of error. Figure 5(b) shows that the normed parameter error, averaged over all of the robots, decreased over time, indicating that the robots were learning the sensory function. Figure 5(c) shows $\sum_{i=1}^{n}\hat{a}_i(t)^T\sum_{j\in\mathcal{N}_i}(\hat{a}_i(t)-\hat{a}_j(t))$, representing the disagreement among the parameter vectors of different robots. The disagreement started at zero because all parameters were intialized with the same value of $a_{\min}$. The disagreement initially grew, then decreased as the robots' parameters reached a consensus.

3.2 Measured Sensory Function

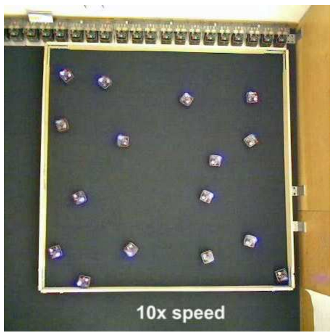
An experiment was also carried out using light intensity over the environment as the sensory function. Two incandescent office lights were placed at the lower left corner of the environment, and the robots used on-board light sensors to measure the light intensity. The same 3×3 grid of basis functions as in the first experiment was used. In



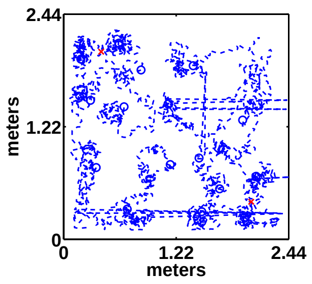
(a) Initial Snapshot



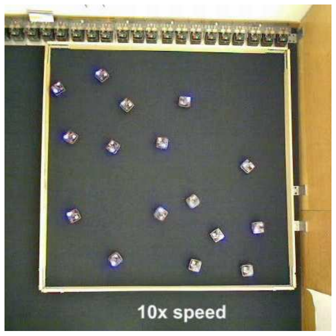
(b) Initial Config.



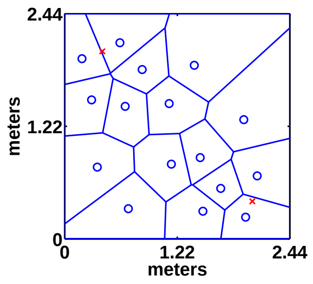
(c) Middle Snapshot



(d) Trajectories



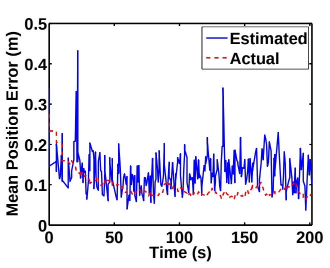
(e) Final Snapshot



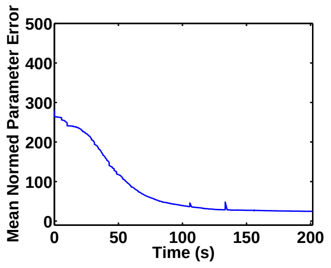
(f) Final Config.

Fig. 4. Results for the algorithm are shown in video snapshots in the left column (4(a), 4(c), and 4(e)). The positions collected from the overhead camera for the same experiment are plotted in the right column (4(b), 4(d), and 4(f)). The Gaussian centers of $\phi(q)$ are marked by red x's.

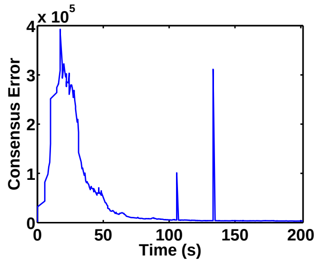
this experiment there was no ground truth against which to compare the performance of the algorithm since we did not know the "true" light intensity function over in the



(a) Mean Position Error



(b) Mean Parameter Error



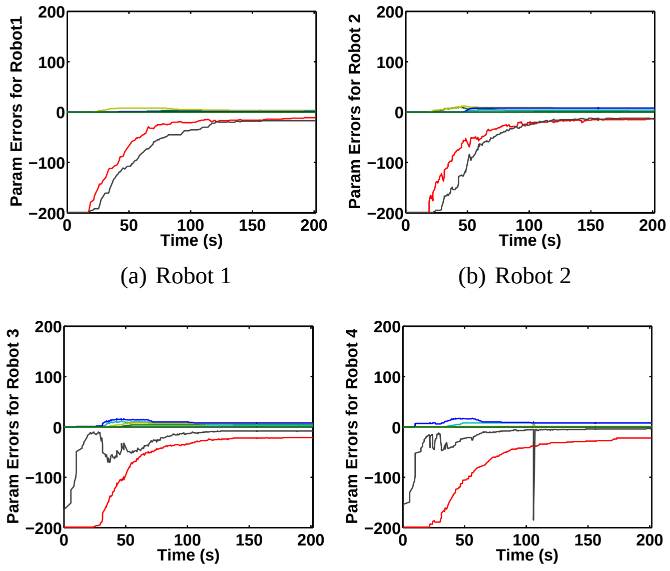
(c) Consensus Error

Fig. 5. The distance to the actual centroid, and the distance to the estimated centroid, averaged over all the robots in the network are shown in 5(a). The normed parameter error averaged over all robots is shown in 5(b). The plot in 5(c) shows a quantity representing the disagreement of parameters among robots.

environment. We instead show that the algorithm caused the network to do what one would expect given the *qualitative* light intensity distribution.

Firstly, we examine the configuration of the robots. Figure 7 shows snapshots of the experiment taken from the overhead camera. Notice that the robots collected in higher density around the light sources while still covering the environment. Fig. 8(a) shows that the distance to the robots' estimated centroids decreased, albeit with a significant amount of noise due to uncertainty in the neighbor position estimates, as in the previous experiment. Figure 8(a) also shows the distance to the estimated centroid filtered so that the decreasing trend becomes more evident. Figure 8(b) shows that, as in the previous experiment, disagreement between robot parameters initially grew, then decreased as the robots tended toward consensus.

Secondly, we examine the function approximation that the robots learned in the experiment. The robots learned a function with a large weight near the position of the light sources, as shown in Fig. 10. Again in Fig. 9 the largest parameter for each of the four robots shown is that corresponding to the basis function centered nearest the position of the light sources. The other parameters adjust to find the best fit of the data within the parameterized space of functions. There are two distinct sources of possible function approximation error that arise from using actual sensor measurements. Firstly, the robots' light sensors have noise which inherently effects



(a) Robot 1 (b) Robot 2

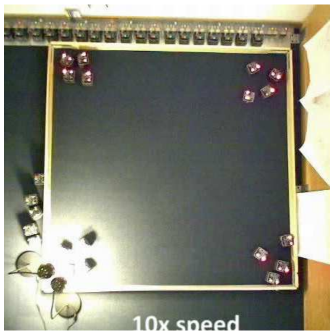
(c) Robot 3 (d) Robot 4

Fig. 6. The parameter errors for each of the 9 parameters in the vector $\hat{a}_i(t)$ are shown for four of the robots in the network. The errors for parameters 3 and 7 started large and decayed to nearly zero for all robots, indicating that the robots learned the function as they moved around the environment.

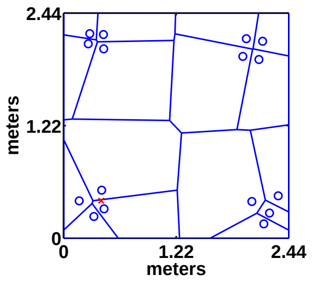
their ability to learn the underlying true light distribution. Secondly, it is unlikely that any linear combination of the simple 3×3 grid of Gaussians could represent the true light intensity over the environment. That is to say, the robots were forced to find the *best fit* to a function which was too complex to be represented with their function approximation scheme. However, the function learning procedure seems to be robust to both of these sources of error, as evidenced by the plots in Fig. 10 and 9.

4 Conclusions

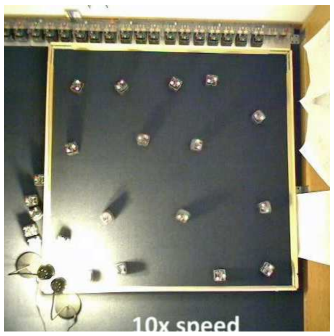
In this paper, we modified a theoretically-proven controller for multi-robot coverage controller to enabled it to perform in real-time on a minimalist platform. We explored the assumptions of the theoretical controller and described an implementation which requires mapping continuous computations to discrete approximations. The controller was adapted to the hardware platform available, and was shown to perform robustly despite the presence of sensor and actualtor noise, and other real-world complications. We presented the results of two experiments with 16 robots. In the first experiment, the robots were given simulated sensory function measurements so that we could compare the results with a known ground truth. The results showed that the algorithm performed as expected. In the second experiment, the robots used



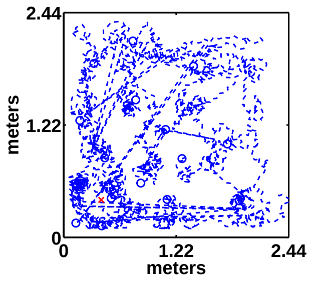
(a) Initial Snapshot



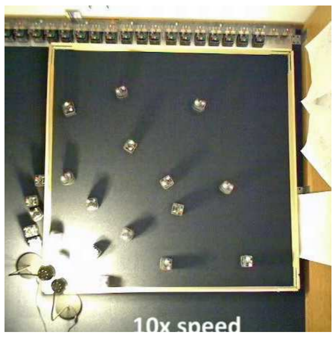
(b) Initial Config.



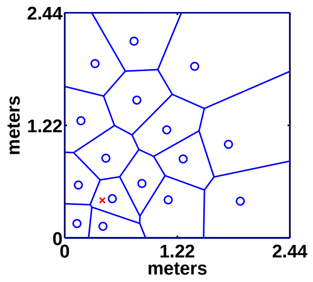
(c) Middle Snapshot



(d) Trajectories



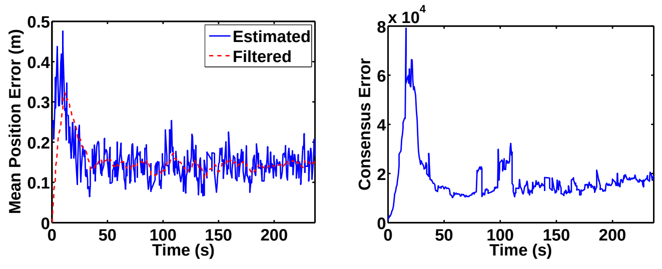
(e) Final Snapshot



(f) Final Config.

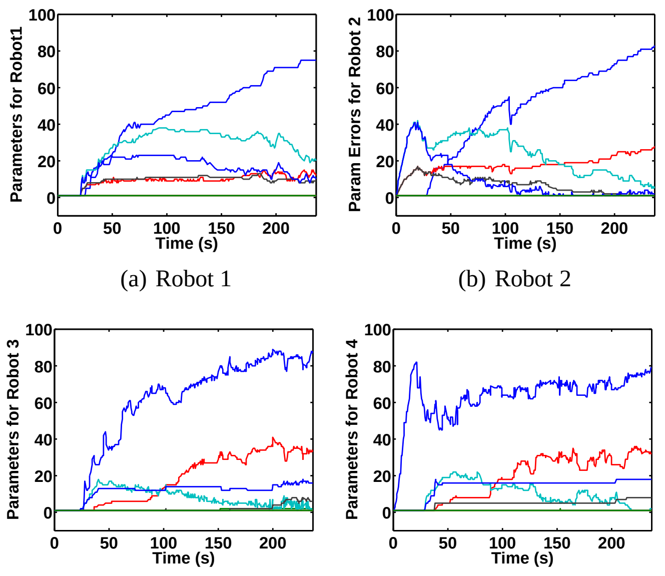
Fig. 7. Results for the algorithm are shown in video snapshots in the left column (7(a), 7(c), and 7(e)). The positions collected from the overhead camera for the same experiment are plotted in the right column (7(b), 7(d), and 7(f)). The robots used the light intensity measured with on board light sensors as the sensory function.

measurements from light sensors as a sensory function. In this experiment there was no known ground truth with which to compare, but the algorithm appeared to behave correctly. We hope these results represent a significant step toward the use of



(a) Mean Position Error (b) Consensus Error

Fig. 8. The distance to the estimated centroid, averaged over all the robots in the network is shown in 8(a). Since measured light intensity was used, the distance to the "true" centroid could not be calculated as in Fig. 5. Instead, we plot a filtered version of the estimated centroid which shows the decreasing trend. The plot in 8(b) shows a quantity representing the disagreement of parameters among robots.



(a) Robot 1 (b) Robot 2

(c) Robot 3 (d) Robot 4

Fig. 9. The parameter values for each of the 9 parameters in the vector $\hat{a}_i(t)$ are shown for four of the robots in the network. Since measured light intensity was used, there is no notion of ideal parameters, and therefore we cannot show parameter errors as in Fig. 6. The light was positioned closest to the center of the basis function in the lower left of the environment, and the parameter for that basis function grew to be the largest, as one would expect. The other parameters also grew to fit the measured light intensity as well as possible.

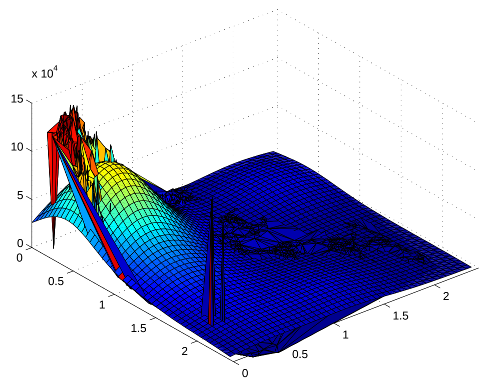


Fig. 10. The basis function approximation of the light intensity (smooth surface) over the area for one robot is shown superimposed over a triangular interpolation of the light intensity measurements of all the robots (jagged surface). The approximation is as close as possible to the light measurements given the basis function parameterization.

multi-robot coverage control algorithms in practical monitoring and surveillance applications in the future.

5 ACKNOWLEDGMENTS

This work was supported in part by the MURI SWARMS project grant number W911NF-05-1-0219, NSF grant numbers IIS-0513755, IIS-0426838, CNS-0520305, CNS-0707601, and EFRI-0735953, and a Boeing Strategic University Initiative grant. Computing hardware was donated by Intel. This work was done in the Distributed Robotics Laboratory at MIT.

References

1. E. Acar and H. Choset. Complete sensor-based coverage with extended-range detectors: A hierarchical decomposition in terms of critical points and voronoi diagrams. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1305 – 1311, October 2001.
2. P. Atkar, H. Choset, and A. Rizzi. Towards optimal coverage of 2-dimensional surfaces embedded in R3: choice of start curve. In *Proceedings of International Conference on Intelligent Robots and Systems*, volume 4, pages 3581–3587, October 2003.
3. Z. J. Butler, A. A. Rizzi, and R. L. Hollis. Complete distributed coverage of rectilinear environments. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, Hanover, NH, March 2000.

4. Z. J. 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.
5. H. Choset. Coverage for robotics—A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31:113–126, 2001.
6. 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.
7. Z. Drezner. *Facility Location: A Survey of Applications and Methods*. Springer Series in Operations Research. Springer-Verlag, New York, 1995.
8. D.T. Latimer IV, S. Srinivasa, V.L. Shue, S. Sonne adnd H. Choset, and A. Hurst. Towards sensor based coverage with robot teams. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 961 – 967, May 2002.
9. D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida. Cooperative sweeping by multiple mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, Minneapolis, Minnesota, 1996.
10. J. McLurkin. Stupid robot tricks: A behavior-based distributed algorithm library for programming swarms of robots. Master's thesis, MIT, 2004.
11. 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.
12. R. Sanner and J.J.E. Slotine. Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, 3(6), 1992.
13. M. Schwager, J. J. E. Slotine, and D. Rus. Consensus learning for distributed coverage control. In *Proceedings of International Conference on Robotics and Automation*, Pasadena, CA, May 2008.
14. I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918–933, 1999.