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

Mac Schwager¹, James McLurkin¹, Jean-Jacques E. Slotine², and Daniela Rus¹

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. In our previous work [7] we introduced a distributed, theoretically-proven controller for a group of robots to provide sensor coverage over an environment. In this paper we describe the algorithmic and systems challenges we solved to implement our theoretical solution to coverage control on a physical robot swarm with computationally lean nodes. We present results of experiments with 16 robots.

Our algorithm 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 [3, 4]. 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 on-line. Then each robot “chases” its estimated centroid until it is eventually reached. 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 is considerable existing work on multi-robot coverage control. One approach considers planning paths for robots so that every point in the environment is
visited by the sensor footprint of at least one robot. A thorough survey of this approach can be found in [2]. 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 [1, 3]. We focus primarily on the framework introduced in [3] which poses the problem as an optimization of a cost function that has been well-studied in the facility placement literature [4]. However in [3] the robots are assumed to know beforehand the distribution of sensory information in the environment. In our algorithm the robots do not know the distribution of sensory information in the environment. Instead, the robots learn the distribution of sensory information while performing coverage. Our algorithm can be thought of as proceeding in two complementary spaces, as described graphically in Figure 1(a). 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.

![Fig. 1.](image)

This paper presents experimental results with our coverage control algorithm using a group of 16 SwarmBots. In Section 2 we introduce the algorithm in a form that is practical for implementation on robot platforms with limited computational resources. We also enumerate the differences between the practical algorithm and the idealized controller that was presented in [7]. In Section 3 we give results of two experiments and show experimental snapshots. The algorithm is shown to operate in realistic situations in the presence of noise on sensor measurements and actuator outputs. Conclusions and discussion are in Section 4.
2 Technical Approach

We model the robots as points moving in a plane. Specifically, we have \( n \) 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 \( \{p_1, \ldots, p_n\} \) 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, we define the sensory function \( \phi : Q \rightarrow \mathbb{R} \), 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) and an unknown parameter vector. That is, robot \( i \)'s approximation of the sensory function is given by

\[
\hat{\phi}_i(q,t) = \mathcal{K}(q)^T \hat{a}_i(t),
\]

where the vector of basis functions \( \mathcal{K}(q) : Q \rightarrow \mathbb{R}^m \) are fixed and are the same for all robots, but each robot has a potentially different parameter vector \( \hat{a}_i \in \mathbb{R}^m \) which changes in time. Figure 1(b) shows a graphical representation of this function approximation scheme. To learn the sensory function \( \phi(q) \) the robots tune their parameter vector 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.

Approximating a function with the parameterization in (1) is standard in many kinds of learning algorithms. 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 [6]. 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 choose from including, wavelets, sigmoids, and splines.

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 \( \Delta q \). Then the estimated centroid \( \hat{C}_{V_i} \) of \( V_i \), weighted by \( \hat{\phi}_i(q,t) \), is given by

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

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(a). In position space, the robots pursue their estimated centroids, given by

\[
p_i(t + 1) = \hat{C}_{V_i}(t)
\]

In parameter space, the robots collaboratively learn the function \( \phi(q) \). They do this by iteratively integrating the values of \( \phi(p_i) \) into the quantity \( \lambda_i(t) \). They also integrate the value of the basis function vector at their position \( \mathcal{H}(p_i(t)) \) into the quantity \( \Lambda_i(t) \). Specifically,

\[
\lambda_i(t + 1) = \lambda_i(t) + \mathcal{H}(p_i(t)) \phi(p_i(t)) \quad \text{and,}
\]

\[
\Lambda_i(t + 1) = \Lambda_i(t) + \mathcal{H}(p_i(t)) \mathcal{H}(p_i(t))^T.
\]

Each robot then tunes its parameter vector using

\[
\hat{a}_{i,\text{pre}}(t) = \hat{a}_i(t) + \gamma(\lambda_i(t) - \Lambda_i(t) \hat{a}_i(t)) + \zeta \sum_{j \in N_i(t)} (\hat{a}_j(t) - \hat{a}_i(t)).
\]

where \( \gamma \) and \( \zeta \) are positive gains. The term \( \lambda_i(t) - \Lambda_i(t) \hat{a}_i(t) \) changes the parameters to follow the negative gradient of the Least Squares cost function. The term \( \sum_{j \in N_i(t)} (\hat{a}_j(t) - \hat{a}_i(t)) \) has the effect of propagating every robot’s parameters around the network to be used by every other robot, and ultimately causes all robots’ parameter vectors to approach a common value. Finally, parameters are maintained above a predefined minimum positive value \( a_{\text{min}} \in \mathbb{R}, a_{\text{min}} > 0 \) using

\[
\hat{a}_i(t + 1) = \max(\hat{a}_{i,\text{pre}}(t), a_{\text{min}}),
\]

where the \( \max(\cdot, \cdot), \) operates element-wise on the vector \( \hat{a}_{i,\text{pre}}(t) \). 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
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 $\phi_i(p_i)$ with its sensors

Initialize: $\lambda_i(0) = 0, \dot{\lambda}_i(0) = 0,$ and $\hat{a}_i(0) = [a_{\min}, \ldots, a_{\min}]^T$

loop

Update:

$\lambda_i(t + 1) = \lambda_i(t) + \mathcal{X}(p_i(t))\phi_i(p_i(t))$
$\Lambda_i(t + 1) = \Lambda_i(t) + \mathcal{X}(p_i(t))\mathcal{X}(p_i(t))^T$
$\dot{a}_{i,\text{pre}}(t) = \dot{a}_i(t) + \gamma(\lambda_i(t) - \lambda_i(t))\dot{a}_i(t)) + \zeta \sum_{j \in N_i(t)} (\dot{a}_j(t) - \dot{a}_i(t))$

Project $\dot{a}_{i,\text{pre}}(t)$ to ensure parameters remain positive: $\dot{a}_i(t + 1) = \max(\dot{a}_{i,\text{pre}}(t), a_{\min})$

Compute the robot’s Voronoi region $V_i$
Discretize $V_i$ into grid squares with area $\Delta q$ and center points $q \in V_i$
Compute the centroid estimate:

$\hat{C}_V_i(t) = \sum_{q \in V_i} q\hat{\phi}_i(q, t)\Delta q$, where $\hat{\phi}_i(q, t) = \mathcal{X}(q)^T\dot{a}_i(t)$

Drive to the estimated centroid: $p_i(t + 1) = \hat{C}_V_i(t)$

end loop

$\hat{\phi}_i(q, t)$, 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 [7], 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 robots, a number of complications are introduced that were not considered in [7], 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 [7]. 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 [7]

<table>
<thead>
<tr>
<th>Algorithm 1</th>
<th>Controller from [7]</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Discrete-time difference equations</td>
<td>• Continuous-time differential equations</td>
</tr>
<tr>
<td>• Nonholonomic “unicycle” robot dynamics</td>
<td>• Holonomic “integrator” robot dynamics</td>
</tr>
<tr>
<td>• Asynchronous execution of instructions</td>
<td>• Synchronous evolution of equations</td>
</tr>
<tr>
<td>• Approximate Voronoi cells constructed from noisy measurements of neighbors within sensing range</td>
<td>• Exact Voronoi cells computed from exact positions of all Voronoi neighbors</td>
</tr>
<tr>
<td>• Discretized sums over the Voronoi cell</td>
<td>• Exact integrals over the Voronoi cell</td>
</tr>
<tr>
<td>• Noisy measurement of global position</td>
<td>• Exact knowledge of global position</td>
</tr>
<tr>
<td>• Noisy actuators</td>
<td>• Noiseless actuators</td>
</tr>
<tr>
<td>• Noisy measurement of sensory function</td>
<td>• Noiseless measurement of sensory function</td>
</tr>
<tr>
<td>• Basis function approximation cannot reconstruct exact sensory function</td>
<td>• Basis function approximation can reconstruct sensory function exactly with ideal parameter vector</td>
</tr>
</tbody>
</table>

3 Results and Experimental Snapshots

The algorithm was implemented in integer arithmetic on a network of 16 SwarmBots [5] (Figure 6(a)). Each SwarmBot used an on-board IR system to sense relative neighbor positions (for computing its Voronoi cell) and to communicate its parameter vector to its neighbors. 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 40MHz 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.

The system reliably performed numerous experiments and demonstrations. Here we present detailed 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 allowed 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 used 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 $\phi(q)$. 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 was chosen to be 9 Gaussians arranged in a grid over the square environment. In particular, each of the nine components of $\mathcal{K}(q)$ was implemented as
$1/(2\pi\sigma^2)\exp\left\{-\|q - \mu_j\|^2/(2\sigma_j^2)\right\}$, where $\sigma_j = .37m$. The $2.44m \times 2.44m$ square was divided into an even $3 \times 3$ grid and each $\mu_j$ 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 $\phi(q)$.

Figure 2 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 3(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 noise in the IR system used to measure the neighbor positions. This caused the Voronoi cells to change rapidly, which in turn caused the centroid estimates to be noisy. Despite this noise, the true distance to the centroid decreased steadily, indicating that the algorithm is robust to these significant sources of error. Figure 3(b) shows that the normed parameter error, averaged over all of the robots, decreased over time. Figure 3(c) shows $\sum_{i=1}^n \hat{a}_i(t)/\sum_{j \in N_i}(\hat{a}_i(t) - \hat{a}_j(t))$, representing the disagreement among the parameter vectors of different robots. The disagreement started at zero because all parameters were initialized 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 $3 \times 3$ grid of basis functions as in the first experiment was used. In 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 environment. We instead show that the algorithm caused the network to do what one would expect given the qualitative light intensity distribution.

Figure 4 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. 5(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 5(a) also shows the distance to the estimated centroid filtered so that the decreasing trend becomes more evident. Also, Fig. 6(b) shows that the robots learned a function with a large weight near the position of the light sources. The weights on the 9 Gaussians adjusted to find the best fit of the data. Figure 5(b) shows that, as in the previous experiment, disagreement between robot parameters initially grew, then decreased as the robots tended toward consensus. The parameters never actually reach consensus because of noise and calibration differences among the different robots’ light sensors.
Fig. 2. Results for the algorithm are shown in video snapshots in the left column (2(a), 2(b), and 2(c)). The positions collected from the overhead camera for the same experiment are plotted in the right column (2(d), 2(e), and 2(f)). The Gaussian centers of $\phi(q)$ are marked by red x’s.

Fig. 3. The distance to the actual centroid, and the distance to the estimated centroid, averaged over all the robots are shown in 3(a). The normed parameter error averaged over all robots is shown in 3(b). The plot in 3(c) shows a quantity representing the disagreement of parameters among robots.

4 Conclusions

In this paper, we implemented a control algorithm for multi-robot coverage on a minimalist robot platform. The controller was adapted to the hardware platform available, and was shown to perform robustly despite the presence of sensor and actuator noise, and other real-world complications. We presented the results of two
Fig. 4. Results for the algorithm are shown in video snapshots in the left column (4(a), 4(b), and 4(c)). The positions collected from the overhead camera for the same experiment are plotted in the right column (4(d), 4(e), and 4(f)). The robots used the light intensity measured with on board light sensors as the sensory function.

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

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. In the second experiment, the robots used measurements from light sensors as a sensory function. We hope these results represent a significant step toward the use of multi-robot coverage control algorithms in practical monitoring and surveillance applications in the future.
The iRobot SwarmBot platform is shown in 6(a). The basis function approximation of the light intensity (smooth surface) over the area for one robot is shown in 6(b) superimposed over a triangular interpolation of the light intensity measurements of all the robots (jagged surface).

Fig. 6. The iRobot SwarmBot platform is shown in 6(a). The basis function approximation of the light intensity (smooth surface) over the area for one robot is shown in 6(b) superimposed over a triangular interpolation of the light intensity measurements of all the robots (jagged surface).

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.

References