Simulation and Experimental Evaluation of Complete Sensor-based Coverage in Rectilinear Environments

Zack J. Butler[*], Alfred A. Rizzi and Ralph L. Hollis
Robotics Institute
Carnegie Mellon University
zackb@cs.dartmouth.edu, {arizzi,rhollis}@ri.cmu.edu

Abstract: Although sensor-based coverage is a skill which is applicable to a variety of robot tasks, its implementation has so far been limited, mostly by the physical limitations of traditional mobile robots. This paper presents sensor-based coverage algorithms both for a single robot and for a team of independent robots which have been designed to allow for easy integration on to real robots. The specific robots in question are planar robots called couriers, components of the minifactory, an automated assembly system. The couriers have excellent position sensing, which enables them to perform coverage, but have no explicit range or contact sensors to detect boundaries, which adds to the complexity of the coverage algorithm. A set of experiments from simulation is presented to show the overall efficiency of the single-robot and cooperative coverage processes in a variety of environments. A second set of experiments performed on a real robot demonstrates the ability to reliably perform sensor-based coverage and also illuminates the effects of specific choices in the type of control used.

1. Introduction

Sensor-based coverage is the problem of directing a robot operating in an initially unknown environment to explore each and every point of the environment. The applications are numerous, including demining, floor cleaning, and similar tasks. While several algorithms have been proposed, demonstrations of theoretical correctness and successful applications have led to different approaches. On the theoretical side, several works have shown how to reach every point in an unknown environment in a provably correct way [1, 2, 3, 4], but few of these have been applied to a real robot, and with limited success [4]. Much of this difficulty can be attributed to the challenge of mobile robot localization, compounded by the need to operate over a long time and large space in order to perform coverage of interesting environments. In contrast, one successful application of sensor-based coverage has been the development of autonomous lawn mowers that use pseudo-random motions. This approach requires no onboard mapping or odometry, but gives no theoretical guarantee of a complete traversal of the environment, and is typically quite inefficient [5].

[*]Currently with the Dept. of Computer Science, Dartmouth College, Hanover, NH

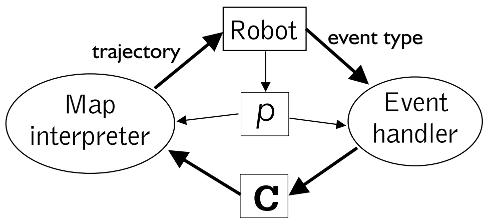


Figure 1. A schematic of the components of CC_R.

In the *minifactory*, an automated assembly system designed for rapid development and deployment [6], autonomous self-calibration is a critical step — once a factory has been assembled from its component modules (each of which is an independent robot or modular piece of infrastructure), the overall structure of the factory must be verified and the precise relative locations of all modules must be determined. Sensor-based coverage by the factory's *couriers* represents one way in which this task can be performed. Couriers are small robots based on planar motors that have reliable position sensing (due to a novel precision AC magnetic position sensor [7]) and motion capability in $\mathbb{R}^2$, although they can only detect the boundaries of their workspace by noting an inability to move in a particular direction. The couriers' position sensing abilities and the system's highly structured (rectilinear) environment provide a domain in which complete sensor-based coverage can be reliably performed.

This paper documents the successful application of new sensor-based coverage algorithms in simulation and on a minifactory courier. We discuss key features of (and additions to) the proven algorithm and present experimental results verifying that with appropriate attention to detail in terms of the algorithmic inputs and outputs as well as the incorporation of appropriate models for position error, a provably correct coverage algorithm can be successfully implemented on a real robot.

2. Coverage algorithms

Because of the unusual nature of the couriers' sensing and environment, a new sensor-based coverage algorithm was required. The algorithm developed, CC_R (*C*ontact-based *C*overage of *R*ectilinear environments), was also designed specifically to allow for the addition of cooperation as well as straightforward implementation on the couriers. CC_R, shown in schematic form in Fig. 1, achieves complete coverage of any rectilinear environment by incrementally building an exact cellular decomposition of the environment $\mathbf{C}$ in a reactive way. In each cycle of the algorithm, the *map interpreter* (half of CC_R) uses an ordered list of rules to evaluate $\mathbf{C}$ and the current position p and chooses a single straight-line trajectory with which to continue coverage. The robot executes the trajectory without interference from CC_R until a specified maximum distance has been traveled or a collision is experienced. At this point the *event handler* updates $\mathbf{C}$ based on the new event and the position at which it occurred. The basic behavior of CC_R is to cover each cell with a *seed-sowing* path, as

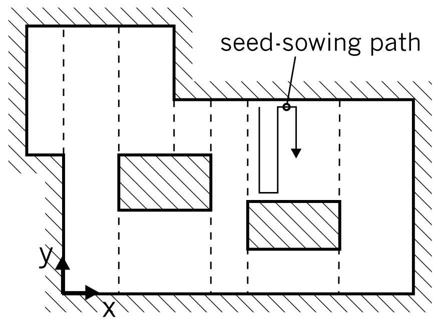


Figure 2. An example of a cellular decomposition that would be constructed by CC_R, along with a portion of a seed-sowing path used to perform coverage.

shown in Fig. 2. It starts with the assumption that the environment can be represented by a single rectangular cell, and commands the robot to perform seed-sowing until a corner is discovered in the environment. When this occurs, it will construct additional cells, localize the corner, and continue seed-sowing in one cell. This process continues until the boundary of $\mathbf{C}$ is known and closed and its interior is covered. A proof that CC_R will produce complete coverage of any finite rectilinear environment was presented in [8], although the proof assumes perfect position sensing.

In addition, an algorithm for distributed sensor-based coverage, DC_R, was developed under which each robot independently runs a slightly modified version of CC_R to perform coverage [9]. Cooperation is induced by adding an algorithmic component which alters $\mathbf{C}$ to reflect data obtained from other robots. This function, the *overseer*, operates in parallel with (and independent of) the event handler, so that the map interpreter can examine $\mathbf{C}$ and plan coverage without knowing anything about the cooperation process. A final additional function works to determine the robots' relative locations in their environment, as this information is assumed to be unknown when coverage begins. This algorithmic structure allows CC_R (and its proof) to be retained virtually without alteration while allowing for increased efficiency in terms of total time required by the team. Starting with the proof of CC_R, a proof was developed that shows that any number of cooperating robots running DC_R will collectively produce complete coverage of their environment [8].

3. Algorithm deployment

Although the coverage algorithms outlined above were developed with real-world application in mind (for example, the use of straight-line trajectory outputs simplifies deployment), there are necessary additions that are not fundamental to the proof of correctness. One important point to consider is localization ability — while the couriers have high precision position sensing, especially compared to most mobile robots, it is still not perfect, nor is the environment (in terms of being strictly rectilinear). Therefore the development of the cell decomposition and the way it is used to generate trajectories for the robot must allow for non-cumulative error in the robot's position. For example, the rules of the map interpreter implicitly assume that the cells of the decomposition do not overlap, and so as the event handler creates and updates cells this property must be maintained.

Under DC_R, these localization issues still apply for each robot, but in addition, when data is shared between robots, each of which have uncertainty in their maps, each robot must build a map that is consistent and representative of the underlying environment. In addition, since DC_R necessarily involves multiple robots in a common workspace, and the robots do not know their relative initial locations, inter-robot collisions are inevitable and must be handled in the context of coverage. The current simulations use a simple reactive avoidance strategy, wherein after two robots collide, they will decide which one should attempt to move out of the other's way. This works fairly well for two robots, but leads to frequent deadlock in large teams and in confined spaces.

Another practical issue is the ability to follow walls and detect gaps in them. Since the couriers have only contact sensing, CC_R was written (and proven) based on the ability of the robot to perform *sliding* motions, in which the robot maintains contact with a boundary while moving along it. The robot must be able to detect a loss of contact with the boundary as well as contact in the direction of motion. If sliding motions cannot be executed, an alternative is to approximate them by interleaving small free-space motions along the boundary with short motions to contact the boundary. These interleaved motions can be produced at the control level (from a sliding trajectory specification) or at the algorithmic level (by slightly altering the rules of the map interpreter that generate the sliding motions). In either case, a proof has been developed that shows that the robot will still produce complete coverage in virtually all rectilinear environments [8].

In addition to these algorithmic issues, when implementing coverage on the courier, there is a choice to be made about the type of control used. The courier's position sensor allows for micron-level precision and accuracy in the range of tens of microns throughout its workspace, enabling a variety of closed-loop control policies (with widely variable parameters) in addition to the open-loop microstepping commonly used to control planar motors. The most important choice is whether to attempt the sliding motions described above. The use of these motions inherently requires smooth boundaries and control based on good force sensing (or estimation). While the distance traveled by the robot to perform coverage is similar whether or not sliding is used, sliding motions are much more efficient in terms of time required, as shown in Sec. 5. A previously developed *dynamic force* controller [10] was used which allows for straight-line motion (both in contact and in free space). This controller uses an estimator to determine the disturbance force on the courier from position data, and attempts to apply a given desired force in each of x and y. Damping is then added in each direction to implicitly set a maximum velocity. By using the same control law for motion and maintaining contact, instability due to controller switching can be avoided. In addition, since the courier has no contact sensors, the controllers also use the force estimate to determine if contact

Environment size	$5w \times 5w$	$10w \times 10w$	$20w \times 20w$
Average cf	2.483	1.710	1.367
Std. deviation	0.1300	0.0636	0.0219

Table 1. Performance of CC_R in square environments of various sizes, where w is the width of the robot.

Environment	Random	Fig. 3a	Fig. 3b
Average cf	2.307	1.986	3.557
Std. deviation	0.329	0.086	0.144

Table 2. Performance of CC_R in various complex environments.

has occurred, and are able to immediately return this result to the higher level interface code.

4. Simulation experiments

Implementations of CC_R and DC_R were developed and operated in simulations that incorporated the position error models and optional sliding motions described above. The simulation was developed using a *world modeler* function in place of real physics, where the "Robot" block appears in Fig. 1. Since both CC_R and DC_R interact with the world through simple interfaces (straight-line trajectory output and a single position reading and one-bit contact sensor inputs), integrating such a function with the coverage algorithms was straightforward. These implementations were then run in a wide variety of simulated environments to determine typical efficiency for a single robot as well as the efficiency gain for robots in a cooperative team. To describe the efficiency of the algorithm as performed experimentally (both in simulation and on the courier), we define the *coverage factor* metric as the average number of times the robot passes over each point in the environment. This is easily calculated as:

$$cf = \frac{d \times w}{\text{Area}(\mathbf{C})},$$

where d is the total distance traveled and w the robot width.

Ideally, the coverage factor for pure seed-sowing would be exactly 1, but two factors make this impossible to achieve in practice. First, unless the robot starts exactly an integer multiple of w away from the edge of the cell, it will finish seed-sowing with a pass that does not add a full robot-width of covered area (because it has only contact sensing, it cannot detect the edge of the cell until it has reached it). Also, in order to ensure detection of all gaps in the top and bottom of each cell, CC_R requires the robot to cover these edges twice. To empirically determine the magnitude of these effects, CC_R was run from 50 different starting locations in each of three empty square environments. The results of these experiments are given in Table 1. These experiments show that these inefficiencies have decreasing effect as the environment gets larger, which is as expected, since they add distance proportional to the perimeter of the cell, which is then divided by the area of the cell to calculate cf.

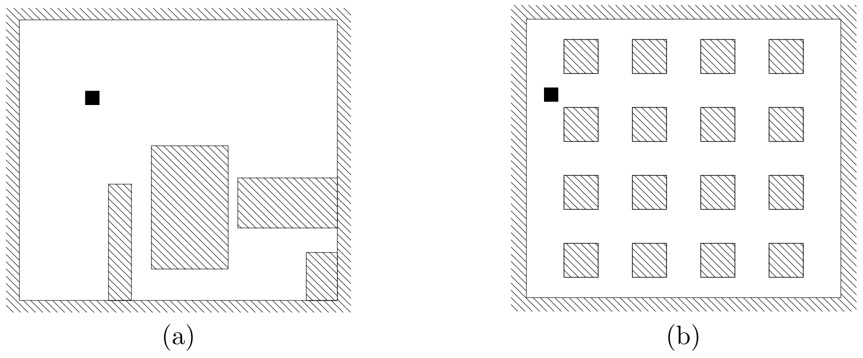


Figure 3. Environments used to test CC_R. The black square in each represents the size of the robot.

CC_R was then tested in a variety of randomly generated environments, with results reported in the first column of Table 2. These environments were generated by populating an open square of dimension $\sim 20w \times 20w$ with between three and eight rectangular obstacles, each with a random height and width between $w/20$ and $10w$. In addition, the two environments shown in Fig. 3 were chosen for repeated tests (50 runs in each), the results of which are also reported in Table 2. The coverage factor for CC_R ranged from under 2.0 for reasonably open environments, such as that of Fig. 3a, to nearly 4.0 for very cluttered spaces, such as Fig. 3b (which was manually designed to be adversarial to CC_R). It is worth noting that these data compare favorably to previously reported values for other (simulated) algorithms, and very favorably to the ratio of about 10 for the randomized technique used by the Friendly Robotics lawn mower[1].

To measure the efficiency gain of cooperating robots running DC_R, teams of sizes from two to ten robots were run in the environment of Fig. 3a. The coverage factor for each robot was then compared to that achieved by a single robot in the same environment. Under DC_R, each robot will develop a complete cell decomposition of the environment, so that the coverage factor for each will simply be proportional to the total distance it has traveled, which will hopefully be smaller for each robot than if it was working alone. This was in fact the case for these experiments, as shown in Table 3. The coverage factor decreased about 30% for each robot in a two-robot team, and almost 50% for each of three robots (this was also true for other similar environments). In addition, these experiments showed that the work was divided fairly evenly among the robots. This was measured by noting the largest coverage factor among the team for each run, and as can be seen in Table 3, in general no robot had to do more than 10% more work than the average robot.

For larger teams, in order to avoid frequent deadlock, the world modeler was set up such that collisions between the simulated robots would not occur (i.e. the robots simply traveled through each other). With this allowance, the

[1]Using data from the manufacturer's data sheet [11], $cf \approx \frac{0.5[\mathrm{m/s}] \times 0.56[\mathrm{m}]}{1000[\mathrm{ft}^2/\mathrm{hr}]} = 10.84$

	Two robots		Three robots		Five robots	Ten robots
	$\parallel$ orient.	$\perp$ orient.	w/ coll	w/o coll		
# trials	15	15	10	10	10	10
Average cf	1.309	1.294	1.134	1.093	0.922	0.698
Avg. max. cf	1.408	1.365	1.268	1.205	1.096	0.790

Table 3. Performance of DC_R in the environment of Fig. 3a. Note that all runs with 2 robots include include inter-robot collisions while all runs with 5 and 10 robots do not.

	Single robot	Two robots		Three robots
		$\parallel$ orient.	$\perp$ orient.	
Number of trials	50	10	11	10
Average cf	3.557	1.936	1.981	1.450
Avg. maximum cf	—	1.947	1.985	1.547

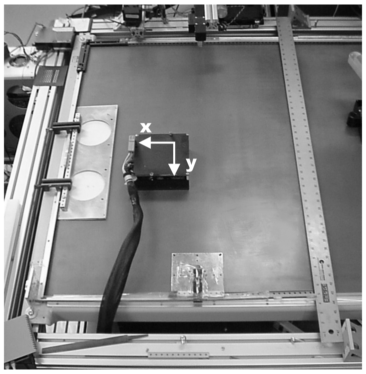
Table 4. Performance of DC_R in the environment of Fig. 3b.

efficiency continued to improve with increasing team size in teams up to ten robots. It is likely that with collisions in place (which would require a better collision avoidance strategy for success), this increase would not be achievable, as the robots would spend considerable time avoiding each other. However, experiments with and without collisions for three robots show only a very slight loss of efficiency with the addition of collisions.

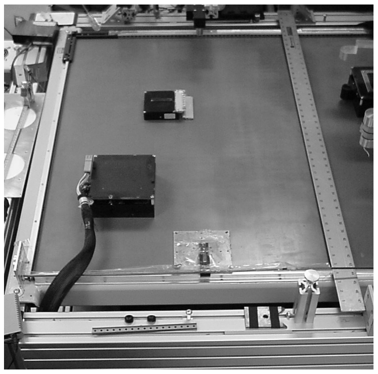
A set of experiments in the adversarial environment of Fig. 3b was also performed, and shows that the constricting nature of this environment was actually beneficial to DC_R. Since this environment will always be decomposed into many small cells, the cooperation between the robots (which happens via transfer of completed cells) could be done more frequently. This allowed the robots to spend less time working in the same areas, leading to a decrease in coverage factor of approximately 45% for each of two robots and nearly 60% for three robots.

5. Courier experiments

As mentioned earlier, the use of straight-line trajectories as outputs from CC_R and the acceptance of small position errors allowed for a straightforward transition from simulation to the courier. The same algorithmic code was used as in simulation by simply replacing the world modeler with calls to the underlying courier motion control system, which in turn used the trajectory specification to create appropriately parameterized controllers. CC_R was tested in three environments: two were those shown in Fig. 4, and the third was a simple rectangle of similar size (approximately 70$\times$100 cm, compared to a courier width of 15 cm). Ten runs were performed in each of two orientations and varying starting positions in each environment to determine the efficiency and qualitative capabilities of the algorithm. The results of these experiments are given in Table 5. These data confirm that the efficiency obtained in simulation is



(a)



(b)

Figure 4. Environments used for CC_R testing, consisting of half of a commercial platen surface with additional obstacles. The tethered courier performing CC_R is included for scale.

Environment	Empty platen		Fig. 4a		Fig. 4b	
Orientation	std.	rot.	std.	rot.	std	rot.
Number of Runs	10	10	10	10	10	10
Average cf	1.69	1.72	2.99	2.68	2.92	3.05
Std. deviation	0.13	0.08	0.27	0.26	0.23	0.30

Table 5. Performance of CC_R on the courier in the environments of Fig. 4. "Standard" orientation is as shown in Fig. 4 (note overlaid coordinate axes) and "rotated" orientation is 90° counter-clockwise.

comparable to that seen in the real system — note that the environments in Fig. 4 are fairly constricted relative to that in Fig. 3a and most other simulated environments, accounting for the somewhat higher coverage factors.

The major failure modes resulted from the courier's tether producing an occasional large disturbance force and the boundaries not being amenable to sliding motions, both of which were overcome with careful engineering. One point of particular interest is the environment shown in Fig. 4b, in which the free space is not simply connected, requiring the robot to successfully attach the cells of the decomposition around the obstacle (in the presence of position uncertainty) to complete coverage. This was successfully done under a variety of conditions, although small additions to CC_R were required.

Finally, although the coverage factor indicates the distance traveled to perform coverage, when instantiated on a real system, the time taken for coverage is also of interest. Therefore, experiments were performed using different types of control and different maximum velocities v_{max} to determine empirically how the type of control affects the overall performance. Each set of control parameters was run from two starting locations in each of two environments, making four coverage "tasks". The simplest type of control used was open-loop trajectory following based on microstepping of the planar motor, using the position sensor only to detect collision (by noticing a significant difference

Control type	O.L.	C.L.	C.L.	C.L.	C.L.
Sliding?	No	No	No	Yes	Yes
v_{max} [mm/s]	70	70	250	70	250
Empty platen (p_1)	310	281	230	115	46
Empty platen (p_2)	318	295	234	140	53
Fig. 4a (p_1)	409	399	291	224	91
Fig. 4a (p_2)	365	341	250	201	79

Table 6. Elapsed time (in seconds) for CC_R under various open-loop (O.L.) and closed-loop (C.L.) control strategies.

between the commanded position and the actual position). While this is the easiest to implement, and is the standard mode of operation for planar motors, it is not capable of recovering from large disturbances such as those caused by strong impact. (This type of operation is fairly insensitive to small disturbances, however, making it suitable for certain types of operations.) Therefore, the maximum speed at which coverage could reliably be completed under this technique was approximately 70 mm/s. The amount of time required for the four coverage tasks is shown in the first column of Table 6. For some of these tasks, as well as some of the closed-loop experiments described below, the experiment was run several times, to confirm that variations from one run to the next due to random disturbances were slight (on the order of 1-3 s) compared to the overall time required.

To measure the effect of the sliding motions, since these operate under closed-loop control, it was first necessary to test the interleaved motions under closed-loop control. As can be seen in Table 6, this resulted in a slight improvement over the open loop technique for the same v_{max}, probably due to the collision detection being more responsive under closed-loop control. Then, since the controller added the capability for recovery from collisions at higher speeds (reliably up to about 250 mm/s), the interleaved motions were tested at a v_{max} of 250 mm/s, and the sliding control was run for the same four tasks at the same two velocities. It was found that without the sliding motions, increasing v_{max} had little effect, as the majority of the elapsed time was spent executing the short interleaved motions along boundaries during which v_{max} was not reached. However, with sliding enabled with small v_{max}, the elapsed time was about half as much as without sliding, since the exploration of the boundaries did not require stopping. In addition, for higher v_{max}, the sliding motions could realize even more advantage during the boundary explorations, and used as little as one quarter of the time required for the interleaved motions with the same maximum velocity.

6. Conclusions

This work has demonstrated that with an appropriately designed coverage algorithm along with a well-engineered robot system, it is possible to reliably execute provably complete sensor-based coverage in the real world. The algorithms interface to the robot through simple channels, allowing them to plug in to a simulation or a robot system in a straightforward way. Experiments both in simulation and on a robot have returned efficiencies that are comparable to previous theoretical results and much better than current commercial hardware implementations.

The major avenue for future work is to implement the cooperative algorithm DC_R on a set of couriers. The biggest challenge there is to implement a strategy for the tethered couriers to share a workspace (in which they initially have no knowledge of each other's location) without tangling their tethers while still making progress toward coverage.

Acknowledgements

The courier experiments would not have been possible without the work of Arthur Quaid, who developed the courier control software and interface code for it. This work was supported in part by NSF grants DMI-9523156 and DMI-9527190.

References

[1] A. Pirzadeh and W. Snyder, "A unified solution to coverage and search in explored and unexplored terrains using indirect control," in *Proc. of IEEE Int'l. Conf. on Robotics and Automation*, pp. 2113–2119, April 1990.

[2] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain covering algorithm for an AUV," *Autonomous Robots*, vol. 3, pp. 91–119, 1996.

[3] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "MAC versus PC: Determinism and randomness as complementary approaches to robotic exploration of continuous domains," *Int'l Journal of Robotics Research*, vol. 19, pp. 12–31, January 2000.

[4] E. Acar and H. Choset, "Critical point sensing in unknown environments for mapping," in *Proc. of IEEE Int'l Conf. on Robotics and Automation*, April 2000.

[5] Friendly Robotics, "RoboSim: RL500 simulator." Available at http://www.friendlyrobotics.com/sim/RoboSim.exe.

[6] R. L. Hollis and J. Gowdy, "Miniature factories for precision assembly," in *Int'l Workshop on Microfactories*, (Tsukuba, Japan), pp. 9–14, 1998.

[7] Z. J. Butler, A. A. Rizzi, and R. L. Hollis, "Integrated precision 3-DOF position sensor for planar linear motors," in *Proc. of IEEE Int'l. Conf. on Robotics and Automation*, May 1998.

[8] Z. J. Butler, *Distributed Coverage of Rectilinear Environments.* PhD thesis, Carnegie Mellon, September 2000.

[9] Z. J. Butler, A. A. Rizzi, and R. L. Hollis, "Distributed coverage of rectilinear environments," in *Proc. of the Workshop on the Algorithmic Foundations of Robotics*, (Hanover, NH), March 2000.

[10] A. Quaid, *A Planar Robot for High-Performance Manipulation.* PhD thesis, Carnegie Mellon, July 2000.

[11] Friendly Robotics, *RL500 Owner Operating Manual.* Available at http://www.friendlyrobotics.com/um/RL500_manual.pdf.