Contact sensor-based Coverage of Rectilinear Environments

Zack J. Butler
The Robotics Institute, Carnegie Mellon University
zackb@ri.cmu.edu

Abstract

A variety of mobile robotics tasks require complete coverage of an initially unknown environment, either as the entire task or as a way to generate a complete map for use in further missions. This is a problem known as sensor-based coverage, in which the robot's sensing is used to plan a path that reaches every point in the environment. Sensor-based coverage algorithms have been developed previously for robots with remote sensing, however, it is possible to cover certain environments using only contact sensing. This paper presents a geometrically complete algorithm CC_R that performs sensor-based coverage in rectilinear worlds using only contact sensing. The outline of a completeness proof of CC_R is also presented, which shows that it produces coverage of any of a large class of rectilinear environments. Implementation of CC_R in simulation is discussed, as well as the results of testing in a variety of world geometries. Finally, potential extensions of the algorithm are discussed, including its use by robots working in a team.

1 Introduction

For robotic missions as diverse as mine detection and floor cleaning, the overall task can be specified as a complete exploration, or *coverage*, of the environment. In general, coverage can be defined as bringing a sensor or effector to every point in an environment, and the *coverage problem* as the generation of a path leading to coverage such as the one shown in Fig. 1a. Coverage can also be used by any mobile robot to generate a complete map of the environment for future tasks. For known planar environments, the coverage problem has been solved, motivated by tasks such as pocket cutting by milling machines[1]. For tasks where the environment cannot be known ahead of time, the sensor or effector needs some computation along with it to simultaneously determine the geometry of the environment and a coverage path within it. This machine then becomes a robot performing *sensor-based coverage*. While the implementation of sensor-based coverage is dependent on the particular class of environment and robot, algorithms have been written that apply to a wide range of systems. However, no single general algorithm has yet been proposed.

In a rather unusual application of sensor-based coverage, the *minifactory*, a modular automated assembly system under development in the Microdynamic Systems Laboratory, will use a version of sensor-based coverage for calibration. In a minifactory, a small example of which is shown in Fig. 2, small robots based on planar linear motors (called *couriers*) travel over table-like *platen* surfaces and are used to transfer products through the factory and participate in the assembly of these products by cooperating with overhead robots. One of the major goals of the minifactory is to greatly reduce design and setup time through the use of independent modular robots, a common programming language, and a high-fidelity simulation environment [2]. To this end, the factory is designed and the robots in it programmed in simulation, and once the actual factory is created, the simulation is automatically updated to match the as-built factory. The individual robot programs are then modified so that the robots move correctly and can coordinate with sufficient precision. To perform this update process, each courier will run a sensor-based coverage algorithm, using contact sensing to determine the layout of the platens. As it covers its workspace, it will also use an upward-pointing optical sensor to locate LED beacons placed on overhead robots. Using sensor-based coverage for this process will ensure the detection and precise localization of all overhead robots in the factory as well as the verification of the overall shape and topology of the factory.

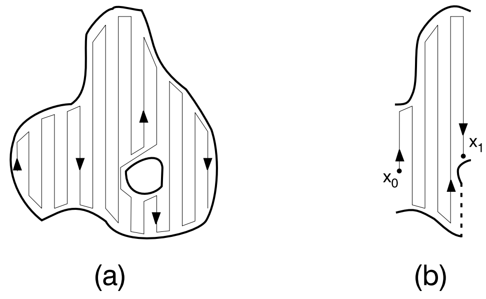

Figure 1: Planned and sensor-based coverage: (a) a planned coverage path of the environment; (b) a possible path taken by a robot performing sensor-based coverage which started at x_0 and has just encountered an obstacle at x_1.

In general, most sensor-based coverage algorithms start with a basic scheme for covering a simple type

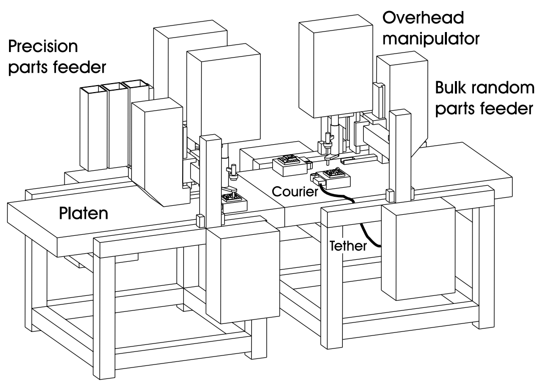

Figure 2: A section of a small minifactory with very simple topology.

of geometry (such as convex or simply connected regions). Coverage then begins by assuming the environment to have this geometry, proceeding until a feature that contradicts the assumption is discovered. At this point the algorithm continues coverage on one side of the feature and remembers the location of the other side as a place to continue coverage from in the future, as shown by the dashed line in Fig. 1b. For example, the method developed by Hert et al [3] puts these line segments in a stack, recursively adding them (when discovering obstacles such as in Fig. 1b) and removing them (when selecting such a segment to cover from) as it progresses. This algorithm allows the robot to completely cover a planar area without building an explicit map. The method of Choset et al [4, 5] instead builds a cell-based map of the environment that is sparse but contains sufficient information for navigation through the environment as well as direction of coverage. In contrast to these, the algorithm proposed by Pirzadeh [6] does not use a specific coverage strategy but simply builds a grid-based map of the environment and moves toward unexplored cells until all cells (each the size of the robot) have been visited or shown to be unreachable.

The cell-based algorithms have provided the basis of the algorithm presented here. In these algorithms, the environment is divided into a set of disjoint cells as shown in Fig. 3, each of which can be easily covered. For planned coverage, the known environment is divided up into cells by passing a vertical 1-D slice through it. Each x location where the number of connected components of the slice changes is called a critical point, and the area between critical points defines a cell. Each cell then has a continuous *floor* and *ceiling* and can be covered with a series of vertical strips from one side to the other. To implement sensor-based coverage within this framework, it is assumed initially that the environment consists of a single cell and coverage of the cell is begun by traveling in vertical strips. When the robot discovers a discontinuity in the floor or ceiling of the cell, a new cell is instantiated on the other side of the discontinuity. Each cell is then covered the same way, with more cells created as necessary, until the boundary of all cells is known and all cells have been covered, at which point coverage is complete.

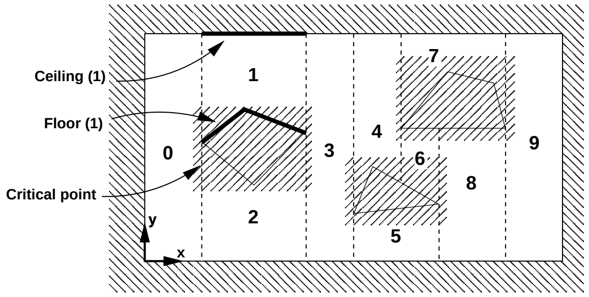

Figure 3: Cell decomposition (due to the method in [4]) of an environment with polygonal obstacles.

2 Problem specifics

To implement sensor-based coverage for couriers, an algorithm will be required that uses contact sensing. So, as a first step toward creating a comprehensive algorithm, and motivated by the geometry of the minifactory, the problem approached here is to create an algorithm that can cover rectilinear environments using only contact sensing. This is in contrast to previous sensor-based coverage algorithms, which have applied to robots that use range sensors such as sonar to determine the locations of obstacles in the environment. Range sensors are quite common in mobile robotics, and coverage algorithms for robots that use them have been shown to be complete for large classes of planar environments. However, for some (and perhaps all) planar environments, remote sensors are not necessary for sensor based coverage. Contact sensors are inexpensive, quite robust with respect to false positives and negatives, and can still provide sufficient information for coverage. In the minifactory, couriers use *implicit* contact sensing to discover the boundaries of their environment — when they attempt to move but cannot, they assume the presence of an impeding obstacle.

Using contact sensing to perform coverage complicates the algorithm, since the sensor's extent is by definition the same size as the robot's extent, and the robot must therefore be treated as having a finite size within its workspace.[1] One complication this introduces is that when the robot has a finite size within a cell decomposition, it is no longer always completely within a single cell, leading to a number of special cases in the implementation of an algorithm. Also, robots with implicit contact sensing require two collisions with a corner (once on each edge) before its location can be determined. The algorithm must therefore allow this location to be uncertain after the initial collision and direct the robot to the second collision, capabilities not required if range sensors are used.

On the other hand, covering a rectilinear environment rather than a more general polygonal one simpli-

[1] Alternately, the coverage can be performed in the robot's configuration space, resulting in a single-point sensor with unusual properties and similar challenges to the algorithm.

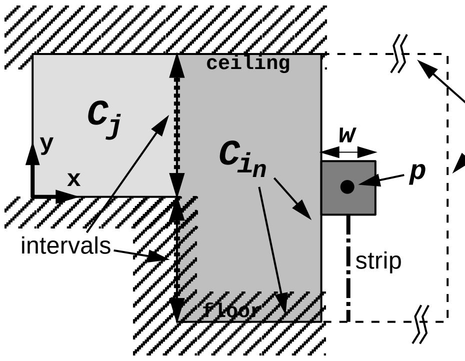

Figure 4: Cell C_i and its data structures; cell C_j has been added for clarity.

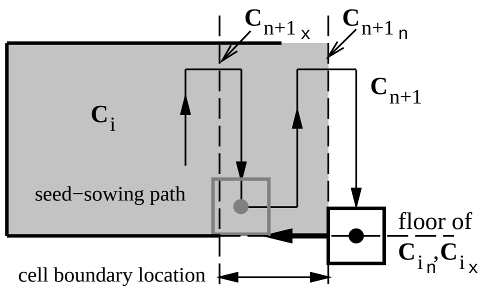

Figure 5: While seed-sowing in C_i, the robot travels outside of C_{i_x}, so a new cell (C_{n+1}) must be added to C and the boundary between them localized.

fies the problem somewhat. For example, cells of a rectilinear environment (when swept parallel to one axis) are themselves simply rectangles, since their floors and ceilings are defined by walls perpendicular to the sweep, and can therefore be represented with simple data structures. However, when creating the current algorithm, care was taken to avoid precluding its extension to more general environments. Also, the rectilinear environment invalidates some standard assumptions of the cell decomposition methods for polygonal environments. Specifically, the "general position" assumption for these methods is that a single obstacle vertex defines each critical point, and that no two critical points occur at the same x value. In a rectilinear environment, the critical points will be due to walls of finite length parallel to the direction of the coverage strips. In addition, we do not preclude having two or more of these walls at the same x location, although the current algorithm was written for environments in which all cells are at least as wide as the robot.

3 Description of CC_R

Inspired by the cell decomposition algorithms described above and taking into account the issues presented in Sec. 2, a new sensor-based coverage algorithm has been developed. Using this algorithm (denoted CC_R for contact-based coverage of rectilinear environments), a square robot with implicit contact sensing (and essentially perfect position sensing) can cover environments with rectilinear boundary and obstacles. It is a completely reactive algorithm based on incremental construction of a cell decomposition of the environment. The robot's exploration is directed only by the state of the cell decomposition C, which is continuously updated as coverage progresses, and the robot's current position p — no time-based history is maintained. The exploration continues until there is no part of C left unknown, at which point the environment has been completely covered. The completeness property of CC_R for most environments is proven in detail in [7], and the proof is outlined below in Sec. 4.

The cell decomposition C consists of an unordered list of cells $C_0 \ldots C_n$ and an unordered list of placeholders $H_0 \ldots H_m$. The data structures that comprise a cell are shown in Fig. 4. The area of a cell C_i is represented by a pair of rectangles: C_{i_n} is its minimum known extent and C_{i_x} its maximum possible extent. Its known side edges each consist of a set of *intervals*, each of which corresponds to another cell or an entire wall segment. The interval representation allows for multiple cells to adjoin one side of a single cell, and also serves to indicate when an edge has been explored - namely, when the intervals span the line from the floor to the ceiling. A placeholder is simply a line segment that lies adjacent to the side edge of a cell and serves as a geometric pointer to an area that has not yet been explored.

CC_R executes based on *events*, where an event is defined as any occasion when the robot has collided with a wall or the current trajectory has completed. When an event occurs, the *event handler* (half of CC_R) makes changes to C based on the new event. The *map interpreter* then uses C and p to determine the robot's current cell C_c and then examines C_c (and p) to determine a new direction of travel and the maximum distance that can be traveled in that direction before a change of direction would be necessary. Thus, if this new trajectory completes without collision, CC_R will run at the right time to replan the coverage path. The trajectory is then executed by the robot without interference from CC_R until another event takes place.

The choice of travel direction at any point is determined by an ordered list of rules. In general, the desired behavior is that of "seed-sowing", namely that the robot should travel in parallel strips of alternating direction slightly closer together than the width of the robot. Seed-sowing is known to be an efficient coverage method in terms of overall path length. In CC_R, for seed-sowing, the robot is directed to a point just outside the edge of the covered portion of the cell and touching the floor or ceiling of the cell, then moved in the y direction until it reaches the opposite edge of the cell. Repeating these two steps results in a seed-sowing path. As the robot performs seed-sowing, however, it will eventually come across unexpected obstacles or corners in the boundary and the seed-sowing will be interrupted. Therefore, other rules must exist to continue coverage correctly in these special cases.

For example, consider the case shown in Fig. 5. The robot was performing seed-sowing in cell C_i when

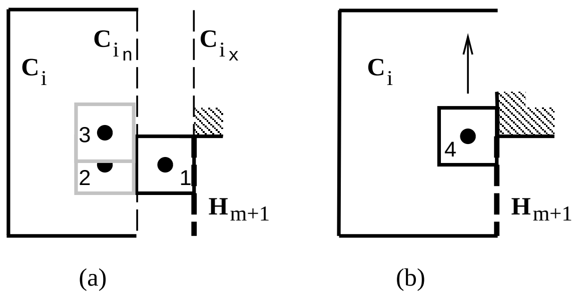

Figure 6: (a) After unexpected contact at location 1, the robot must move around the corner and (b) localize the edge of C_i at location 4 before continuing.

it reached the end of a trajectory and found itself outside C_{i_x}. At this point, C_{n+1} is instantiated by the event handler with minimum and maximum left edges as shown in Fig. 5. The map interpreter then notices the uncertainty of this left edge (note that C_{n+1} is now the current cell) and directs the robot to move sideways to determine its location. The other common special case is illustrated in Fig. 6, in which the robot has come in contact with an obstacle wall. For this case, the event handler first sets the maximum right edge of C_i to be at the right edge of the robot and instantiates a new placeholder H_{m+1}. The map interpreter then notices that the right edge of C_i has finite uncertainty, similar to the left edge of C_{n+1} in the previous example. Rather than moving toward this uncertainty, however, it must direct the robot completely within C_{i_n} (to location 2 in Fig. 6a), then up to location 3, where there must be a wall to its right, before moving back to the right to localize the new edge. The robot then completes the seed-sowing strip that was interrupted by the collision while simultaneously exploring the remainder of this new edge.

In addition to these cases, another situation can arise in which seed-sowing should not be continued in the current cell. When a cell is first entered, the side from which it has been entered should be fully explored before seed-sowing continues away from that edge. This will avoid forcing the robot to return to the edge to explore it once it has reached the other side of the cell. It also puts the cell in a well-defined state (i.e. with one completely known edge) after being created, making it easier to prove that it will always be completed correctly, as shown below.

The rules for all the different cases are tested in order by the map interpreter, with the first applicable rule determining the next trajectory. For this reason, the special cases are tested first, followed by the rule that generates seed-sowing. These rules are as follows, where C_c is the cell containing p:

1. If C_c has a side edge with finite uncertainty, move inside C_c and next to a wall, then toward the edge
2. If a side edge of C_c is not completely explored, move adjacent to the unknown point of the edge closest to p and try to make contact
3. If C_c has an unknown ceiling, travel in $+x$
4. If C_c has an unknown floor, travel in $-x$
5. Otherwise, if C_c is not completely covered, perform seed-sowing (as described above)

It should be noted that rules 1 and 2 are actually tested twice, once for the right side edge and again for the left, and that seed-sowing may occur either to the left or right of the previously covered portion of C_c. Additionally, depending on the location of the robot, various cases within each rule may be triggered to determine travel direction. For example, rule 1 tests to determine if the robot is completely within C_c, and if so, whether it is adjacent to a wall, to select the appropriate trajectory.

Finally, it may be the case that the current cell has been completely covered and all its edges have been explored. This cell is then said to be *complete*. In this case, the map interpreter must direct the robot to an uncovered area (either an incomplete cell or a placeholder). For reasons explained in the proof below, if there is an incomplete cell in C, it is dealt with before any placeholders. If there is not one, the choice of which placeholder to go to next is arbitrary as far as the completeness of the algorithm is concerned. However, for greater efficiency, the map interpreter first looks for placeholders adjacent to the current cell and chooses the nearest of those. If there are none of these, the lowest numbered placeholder is chosen (arbitrarily) as a destination. Then, for all of these cases, a path is planned to the cell or placeholder by creating a graph from the adjacency relationships of the cells and using depth-first search within that graph. To complete the list of rules:

6. If an incomplete cell exists, plan a path to it and travel to the first cell along that path.
7. If C_c has a placeholder H_d adjacent to it, instantiate a cell just beyond H_d with width equal to that of the robot and plan a (straight line) path to the interval corresponding to the new cell.
8. Plan a path to the lowest numbered placeholder and travel to the first cell along that path.

4 Proof/Results

Because the task of sensor-based coverage is to reach every point in an environment, it is important to show analytically that an algorithm will achieve this for all potential environments. Therefore, to show the completeness of CC_R, a proof has been written that verifies that for a certain large class of rectilinear environments and for all initial positions of the robot within the environment E, CC_R will direct the robot to every point in E. Environments for which this proof are valid are all those in which the canonical cell decomposition contains only cells wider than the width of the robot. Completeness is shown as follows:

i. From any initial position, at most two cells are created, both *well-opened* [2]

[2] A cell is well-opened when the location of its floor and ceiling are known and one side edge is completely explored.

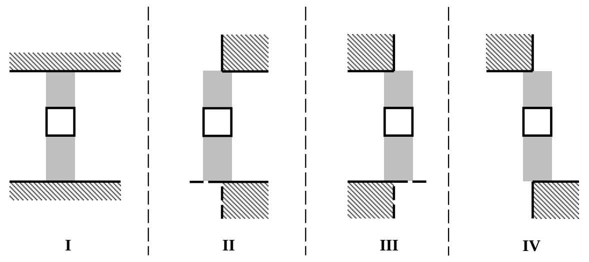

Figure 7: The four types of initial conditions, in which the shaded area is that covered by the first strip of seed-sowing.

ii. Any well-opened cell will be completed (including complete coverage) when visited
iii. Every cell completion results in at most one well-opened cell and any number of placeholders
iv. Any placeholder can be turned into a well-opened cell
v. Every incomplete cell will be visited and every placeholder removed

Each of these items is shown in detail in [7], using the rules of CC_R over the potential range of environment geometries to determine all possible behaviors. A summary of the arguments for each item is presented here. For the initial conditions, there are four distinct cases, as shown in Fig. 7. Note that in all but Case I, the height of this strip (which will start cell C_0) does not correspond to the cell the robot's center is in, and in Case IV, it does not correspond to any cell in the final decomposition. Case I is the general case, in which seed-sowing will continue to the right, and although the cell is not strictly well-opened, it will behave as if it is. Case II also continues to the right, leaving C_0 larger on its left than it should be. This will also work correctly provided this extra area is explored from within before it is approached from above or below, which is assured by rule 6. In case III, a new cell is instantiated during the second strip, leaving C_0 arbitrarily thin. However, the new cell C_1 will be immediately turned into a well-opened cell, and C_0 can be re-entered and coverage resumed regardless of its width. Finally, case IV begins as case III, but upon return to a thin C_0, yet another cell is created during the first seed-sowing strip. This process results in C_0 having zero width, but this does not affect the correctness of C or the ability to plan paths within it.

To prove property ii, it is first shown that rule 2 of CC_R successfully explores any partially known edge. It is then shown that in the absence of overlapping incomplete cells, seed-sowing continues in a well-opened cell until the final unknown edge is discovered, either as shown in Fig. 5 or Fig. 6 or during a horizontal motion of seed-sowing. To determine the location of this edge, the two former cases are always handled as described in Sec. 3, and the latter case is even simpler: when the edge is discovered, its position is immediately known. Then, in all cases, the new edge is partially known and is immediately explored, completing the

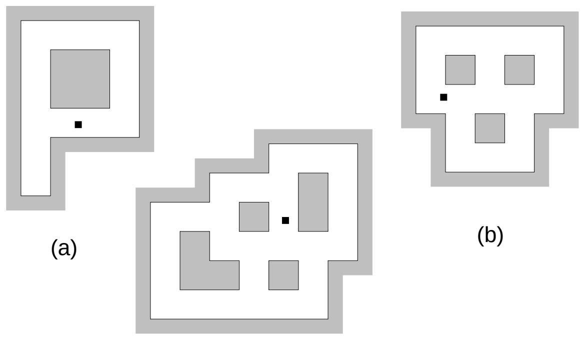

Figure 9: Some of the worlds in which CC_R was tested: (a) "P" world (b) "Skull" world (c) "Large" world. The black square in each is the size of the robot.

knowledge of the cell edges. This also finishes coverage of the cell, thereby completing the cell. This exploration is then shown to result only in the creation of placeholders (no incomplete cells), and so these completions will indeed result in at most one incomplete cell (created only in the case shown in Fig. 5), proving property iii. This property is then used to show that there will never be more than two incomplete cells in C, and that these two will never overlap. This last fact means that the restriction placed on the proof of property ii above is in fact not restrictive at all.

Once the ability to complete any well-opened cell is shown as well as the ability to turn any initial condition into well-opened cells, it is proven that any placeholder can also be turned into a well-opened cell. This is simply done by showing that for the small cell instantiated from a placeholder by rule 7, the robot will enter it and immediately explore the near side edge including discovering the floor and ceiling, making a well-opened cell. Finally, the assurance that all placeholders and cells will be visited (and therefore all area will be covered) is given by rule 8, which will look for and be able to plan a path to *any* cell or placeholder in C, and so CC_R does not terminate until coverage is complete.

In addition to the proof, to verify and test the performance of CC_R, a simulation was programmed within the minifactory simulation environment. This environment allowed for a variety of test environments to be implemented as well as easy 3-D visualization of robot motion under CC_R. In addition to CC_R itself, for this work, software was written that displayed a representation of the robot's cell decomposition as coverage progressed. A screen shot of this software in operation is shown in Fig. 8.

Once encoded, CC_R was run in a variety of environments such as the ones shown in Fig. 9. Each run began at a random position (including all types of initial conditions) and one of four orientations and the robot proceeded to cover the environment. After several iterations of debugging CC_R and its implementation, the final version was able to successfully cover each of 5 worlds over 25 times without a single failure.

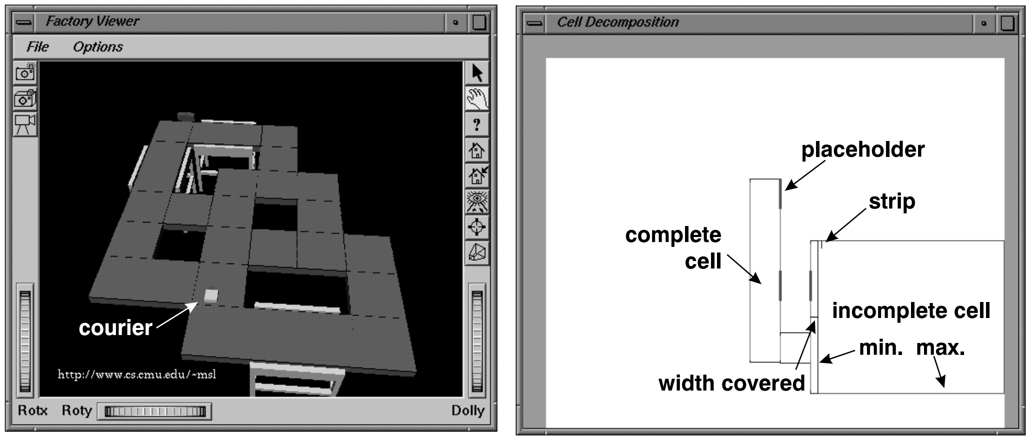

Figure 8: A screen shot of the simulation of CC_R. The left window is a rendering of the environment which is being covered, and the right window is a graphical representation of the current cell decomposition (labels have been added by hand; in the simulation, different colors are used for the different data structures).

5 Conclusions/Future work

Sensor-based coverage is a problem with a number of applications in robotics. The algorithm presented here, CC_R, has been shown both analytically and empirically to solve this problem for a previously unconsidered type of robot system. Additionally, work is underway to implement CC_R on the actual minifactory hardware, a task which is simplified by having trajectory outputs from the algorithm.

CC_R also has great potential to be extended. One such extension is to a larger class of robots and environments. Current research is underway on an algorithm CC_P which will have similar structure to CC_R but apply to circular robots with a ring of one-bit contact sensors operating in a planar environment with polygonal obstacles. This type of system is more comparable to those handled by other coverage algorithms and closer to common mobile robotics tasks than the rectilinear world currently handled. In addition, CC_P will have similar data structures to CC_R that will allow it to handle a slightly wider class of environments than other coverage algorithms.

Additionally, in order for CC_P to be applied to a wide range of mobile robots, it will need to be able to handle robots with some dead reckoning error in their position sensing. This could be as simple as putting uncertainties on wall locations based on models of sensor error. However, it might also be reasonable to change the basic coverage scheme or the way in which edges are localized depending on the particular types of errors likely in the positioning.

Finally, research is beginning that will take CC_R and apply it to a team of robots. In this work, which will eventually be applied to the minifactory to make the calibration process more efficient, each robot performs a slightly modified version of CC_R while passing information back and forth about their common environment. Additional functions will be added in parallel with CC_R to interactively change the robot's cell decomposition based on knowledge gained through communication with other robots. This will have the effect of reducing the amount of area covered by each robot in the team without having to fundamentally change CC_R, allowing the completeness of CC_R for each individual robot to be retained.

Acknowledgements

This work was supported in part by NSF grant DMI-9523156. The author was supported in part by an NSF Graduate Research Fellowship. The author would like to thank Jay Gowdy, who wrote the minifactory simulation environment, and Al Rizzi and Howie Choset for thoughtful discussions about CC_R and its completeness proof.

References

[1] M. Held, *On the Computational Geometry of Pocket Machining.* Springer-Verlag, Berlin, 1991.

[2] A. A. Rizzi, J. Gowdy, and R. L. Hollis, "Agile assembly architecture: An agent-based approach to modular precision assembly systems," in *Proc. of IEEE Int'l. Conf. on Robotics and Automation*, pp. 1511–1516, April 1997.

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

[4] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon decomposition," in *Intl. Conf. on Field and Service Robotics*, 1997.

[5] E. Acar and H. Choset, "Sensor based coverage of unknown environments." submitted to 1999 Intl. Conf. on Robotics and Automation.

[6] 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.

[7] Z. J. Butler, "CC_R: A complete algorithm for contact-sensor based coverage of rectilinear environments," Tech. Rep. CMU-RI-TR-98-27, Robotics Institute, Carnegie Mellon Univ., 1998.