Towards Sensor Based Coverage with Robot Teams

DeWitt Latimer IV, Siddhartha Srinivasa, Vincent Lee-Shue
Samuel Sonne, Howie Choset, Aaron Hurst
Carnegie Mellon University, Pittsburgh, PA 15217

Abstract

We introduce a new algorithm to cover an unknown space with a homogeneous team of circular mobile robots. Our approach is based on a single robot coverage algorithm, a boustrophedon approach, which divides the target two-dimensional space into regions called cells, each of which can be covered with simple back and forth motions. Single robot coverage is then achieved by ensuring that the robot visits each cell. The new multi-robot coverage algorithm uses the same planar cell-based approach as the single robot approach, but also prescribes the methods by which multiple robots cover a cell, teams are allocated among cells, and sub-teams of robots share information in a minimalistic manner. The advantage of this method is that planning occurs in a two dimensional configuration space for a team of n robots, bypassing the need to plan in a $2n$ dimensional configuration space. The approach is semi-decentralized: robot teams cover the space independent of each other, but robots within a team communicate state and share information.

Keywords : Multi-robot, Distributed Coverage, Sensor Based Planning

1 Introduction

Coverage planning for a mobile robot deals with the problem of ensuring that the robot's footprint passes over all reachable points in its target environment. In this paper, we describe an algorithm that directs a team of robots to cover an unknown space solely relying on sensor data acquired on-line. Previous sensor-based work on coverage planning relied on randomized and heuristic algorithms that equipped the robot with a set of behaviors (like *avoid obstacle*, *forage* and *follow wall*)[3] which cooperate to attempt to cover a target region. Unfortunately, such approaches do not posses any guarantees that the target space can be exhaustively covered. Therefore, we have developed a multi-robot coverage strategy that is *complete,* one that possesses provable guarantees which ensures that the robot team passes over all points in a target space. Complete approaches have the advantage of removing any doubt that the robot has successfully covered its environment.

The use of multiple robots can expedite the coverage mission and thus improve our measure of *efficiency* of the operation, which we measure in terms of area covered in unit time. Guaranteeing the most efficient coverage is impossible because the robots have no prior knowledge of the workspace - it is always possible to deploy antagonistic obstacles and lead them astray. However, we prescribe our algorithm to minimize *repeat coverage* which we define as the robot passing over already covered space.

The approach in this paper is based on prior single robot coverage methods. To obtain provable completeness, most complete single robot coverage planners, either explicitly or implicitly, use a *cellular decomposition* of the environment to achieve coverage. A cellular decomposition breaks down the target region into *cells* such that coverage within each cell is simple. Provably complete coverage is then attained by ensuring that the robot visits each cell.

Essentially, this paper presents an adaptation of the single robot cellular decomposition approach for multiple robots. We describe a a semi-decentralized sensor based approach to multi-robot coverage which simultaneously covers an unknown space and construct a cellular decomposition, which in turn is used to guarantee complete coverage. A robot team moves in formation to coverage cell. As cells are created and/or completed, team then splits up into smaller teams, each of which continues coverage. Each robot has a knowledge of its position and heading with respect to a global coordinate frame. Communication is restricted to members of a team — members communicate to maintain formation and to update their knowledge of the world. We also provide heuristics for different robot teams to merge when they encounter each other.

2 Relation to Current Work

Exact cellular decompositions represent the free configuration space by dividing it into non-overlapping cells such that adjacent cells share a common boundary, the interior of each cell intersects no other cell, and the union of all the cells covers the free space.

The boustrophedon decomposition[8] is an exact cellular decomposition where each cell can be covered with simple back and forth motions. The cells are defined by sweeping a slice[6, 7] (a one-dimensional line) through the configuration space and noting where the connectivity of the slice changes in the free configuration space. These connectivity changes occur at critical points. A method that uses simple sonar range sensors to detect critical points was introduced in[1]. Using this method, the robot can simultaneously cover an unknown space while looking for critical points to ensure complete coverage.

Butler *et. al.*[4] achieve complete coverage of unknown rectilinear environments using a square robot with intrinsic contact sensing. They perform an online decomposition where each cell, in the shape of a rectangle, is formed such that it can be covered completely by back and forth motions performed parallel to one of the walls of the environment.

Butler *et al* have also developed a cooperative sensor-based coverage algorithm DC_R[5] based on the single robot CC_R algorithm. The basic concept of DC_R is that cooperation and coverage are algorithmically decoupled. This means that a coverage algorithm for a single robot can be extended to a cooperative setting. To produce cooperative coverage, CC_{RM} is enhanced with an *overseer* which takes incoming data from other robots and integrates it into the cellular decomposition. It can be shown that the overseer indeed performs this operation in such a way that coverage can continue under the direction of CC_R without CC_R even knowing that cooperation occurred.

3 Distributed Coverage using the Boustrophedon Decomposition

To formulate the multi-robot coverage problem, we borrow the following terms from the single robot coverage algorithm - *slice*, *cell*, *sweep direction*, *critical point* and an *adjacency graph*. The slice and cell are self-evident. The sweep direction, as its name suggests, is the direction the slice is swept. The adjacency graph encodes the topology of the decomposition, where nodes represent the cells and edges connect nodes corresponding to adjacent cells in the workspace, which we denote Ws.

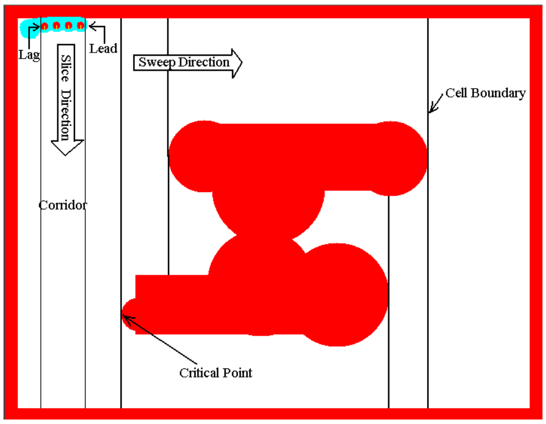

Figure 1: Terminology

We have additional terms – a *corridor*, a *frontier boundary*, *lead* and *lag* robots. For a single robot, a *corridor* is the area in free configuration space that is covered by the robot while following a slice. For multiple robots, the corridor is the area covered when the robot team moves in formation parallel to a slice. A frontier boundary, described in more detail later, sub-divides a cell to allow for more efficient coverage within a cell. The *lead* as the robot in the team that is furthest along the sweep direction. The robot at the opposite end of the team is defined as the *lag* robot. Fig.1 gives a graphical representation of the terms (except for frontier boundary).

The coverage algorithm is described as follows: the robots start together and traverse the space in a formation covering the first cell of the decomposition. The team covers this cell, as well as all other cells, one corridor at a time. If a robot in the team encounters a critical point, the team divides to cover separate cells. This procedure repeats until other critical point are detected, causing more subdivisions.

Each member of a single team shares a common adjacency graph of covered and unexplored cells associated with that particular team. This graph is updated whenever a critical point is detected or when separate teams encounter each other within the same cell. In this case, the teams merge and combine adjacency graphs. Upon completion of a cell. We terminate the coverage operation when all the teams have explored all of the cells in their adjacency graphs. The reminder of this section will detail each step of our coverage algorithm.

3.1 Covering the Interior of a Cell

Our goal is to cover a cell with a team of n-robots, covering one corridor at a time with a valid formation. A valid formation requires that each robot in the team moves along the slice, no two robots can overlap their paths, and the union of the paths fills the corridor. This allows for a variety of formations, including all robots moving in a horizontal line or an echelon formation, as depicted in Figure 1. Observe that repeat coverage cannot occur while covering most of the corridor in a valid formation.

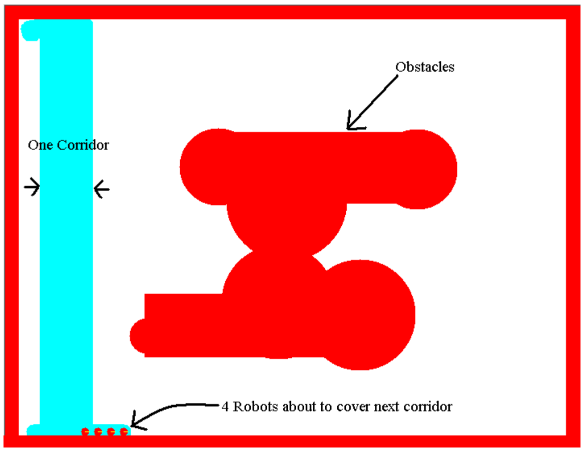

Figure 2: Covering a corridor

A robot stops when it encounters either an obstacle or another robot from a different team. We presume we can distinguish between the two cases. Once all the robots in the formation have stopped motion in the corridor, either the team moves to the next corridor or teams are merged if robots from different teams encounter each other, which is described in §3.5.

Note that when the team finishes covering the corridor it will have broken its formation in order to form fit the bottom of the corridor. At this point, the robot team moves in single file (i.e., robot i follows robot $i+1$) until the team has collectively moved one robot-team-width to the beginning of the next corridor, unless the team detects a critical point. See Fig.2. Note that the length of the path followed team is greater than or equal to robot-team-width (equal if the walls are flat and perpendicular to the slice).

Since critical points occur on the boundary of the environment, the robot team may detect a critical point during this "move over" operation, in which case the robot team does not move over one team-width. This is just one of many wall-following behaviors where a team can detect a critical point, as described in the next subsection.

3.2 Critical Point Sensing

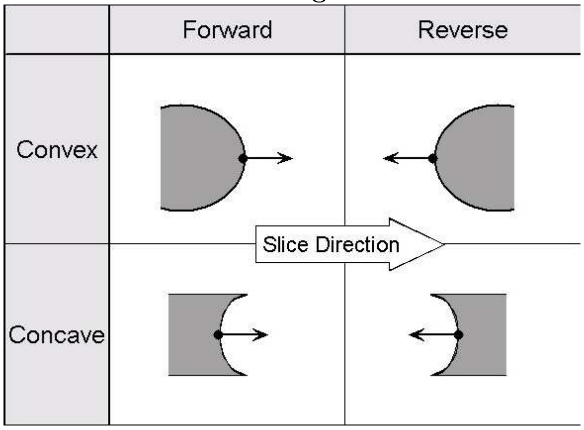

Figure 3: Critical point nomenclature

The explicit detection of critical points occurs when a robot is following the boundary of an obstacle (i.e. some wall following maneuver). Before continuing with the algorithm description, we enumerate the types of critical points that can be encountered. Let x be a critical point on obstacle C_i. Let $B(x)$ be a small ball centered at x. If $B(x) \bigcap C_i$ is convex, then x is a *convex critical point*; if $B(x) \bigcap Ws$ is convex, then x is a *concave critical point*. Concave critical points only indicate the termination of a cell whereas convex critical points not only indicate the border of a cell, but also the beginning of other cells. Critical points may also be defined as *forward critical points*, whose normal vectors point along the sweep direction, and *reverse critical points*, whose normal vectors points opposite to the sweep direction[2]. Fig.3 shows the four possible cases of critical points.

There are two main classes of where teams can encounter critical points: while covering the corridor and while moving to the next corridor. Lets first consider the case where the team passed by a critical point while covering the corridor. At the end of the corridor, each robot in the team is next to a wall (most likely the same wall if there are no critical points). At this point, the team still does not know that it has passed by a critical point or not. To search for these critical points, each member of the team initially has to move in the reverse slice direction along the wall to the location of its *predecessor* in the formation. The predecessor to a robot is the robot immediately adjacent to it in the reverse sweep direction. Note that the lag robot's predecessor is not in the team, but rather the lead robot from the previous corridor. In a sense, the lag robot has a virtual predecessor.

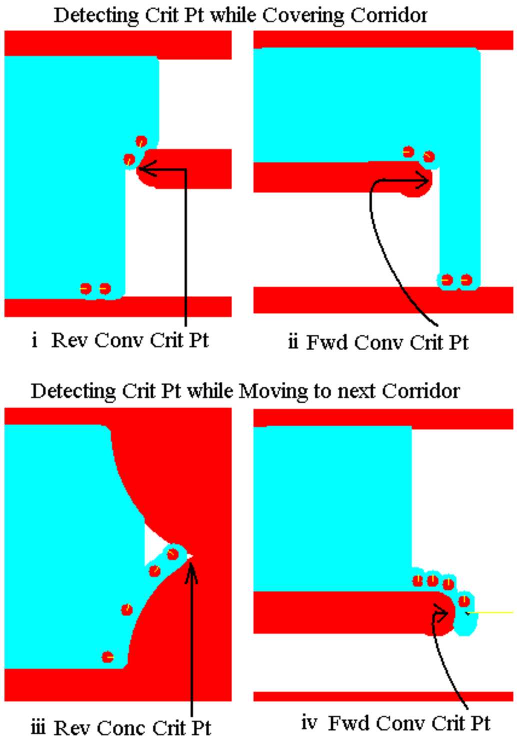

Figure 4: Critical Point Detection

Since we are introducing this additional motion, the motion to the next corridor operation is a bit more complicated than the motion described in the previous section, even if there are no critical points. If there were no critical points, the team would simply move along the boundary in single-file in the reverse sweep direction one robot-width to the previous corridor, and then move one robot-width plus one corridor-width to the next corridor. Note that we should be clear by what we mean by one-robot width – this means that the robot *could* move a path length greater than or equal to one robot-width but moves in the sweep direction by one robot-width, i.e., the distance between the start and goal slice is one robot-width.

However, we are now considering the case that the team has passed by either a reverse or forward critical point while covering the corridor. While undergoing the reverse motion, if a robot detects a reverse convex critical point (Fig.4(i)), it records the location of the critical point for the team, stops wall following, and then moves in the slice direction to the "other" boundary to its predecessor's original location. In actuality, the robot could end up following multiple boundaries to reach its predecessor's location. In this case, the boundary follow/move along the slice motion is repeated, detecting multiple critical points, until the robot achieves its predecessor original location. These actions guarantee that the robot team will find a reverse convex critical point, if one exists.

This reverse motion action will also hunt for a forward convex critical point, as well. In this case, the robot moves back by one robot-width along the boundary in the reverse sweep direction. Again, to be clear, the path length followed may be more than one robot width but the robot moves one of its widths in the sweep direction (horizontal in our examples). If after moving one robot width in the reverse sweep direction, the robot does not reach the location of its predecessor (Fig.4(ii)), it stops wall following and moves up to the "other" boundary to locate its predecessor. Note that when the robot encounters the boundary, it then starts moving in the forward sweep direction along the boundary until it finds its predecessor (finding just one critical point) or moves forward and backward one robot width (finding additional critical points). In the latter case, the robot moves along the slice again until it encounters an obstacle, repeating this process until the robot finds its predecessor. This procedure will detect one or more forward convex critical points.

Note that special consideration must be taken for the lag robot to not miss a critical point. The lag robot has to reverse wall-follow to the position where the lead robot from the previous corridor began its sweep when covering the previous corridor. This would have been the predecessor's position if the lag robot had a predecessor.

Now, we consider the case when critical points are detected while following the boundary from one corridor to the next. This second case has two sub-cases itself: the lead robot encounters a concave reverse critical point (Fig.4(iii)) and the lead robot encounters a forward convex critical point (Fig.4(iv)). Detection here is straightforward because the robots are already on the boundary of the environment.

3.3 Team Division

A team may divide into sub-teams depending upon which type of critical point it detects. The simplest example is when the team is covering a corridor and detects the reverse convex critical point while doing the post-reverse wall following motion (Fig.4(i)). In this case, the current cell has been completed and two new cells are to be covered; the sub-teams each move to the cell to which they are closest. In fact, one of the sub-teams will already be in one of the new cells. The adjacency graph is trivially updated here as well.

Then next two cases deal with forward convex critical points from Fig.4(ii) and (Fig.4(iv). In both of these examples, the team has just covered the cell above the obstacle and now there are two more cells to be covered: one below the obstacle and one to the right. Instead of dividing the teams accordingly to each cell, we divide the cell to the right into two sub-cells along a virtual frontier [5] emanating from the critical point. In this case, the team then covers the top sub-cell of the right-most cell and leaves pointers to the bottom sub-cell and the bottom cell to be covered later. The idea here is that it is possible that another team will come along to cover the bottom cells, so space is being left open to avoid possible repeat coverage.

More specifically, a virtual frontier is perpendicular to the cell boundary extended from the critical point. The virtual frontier terminates when the team discovers an obstacle intersecting the virtual frontier. Until then, the virtual frontier is assumed to continue till the end of the world. All the robots in the team treat the virtual frontier as a wall limiting coverage to one side of the frontier. The heuristic or belief here is that another team will be passing through this cell so space is being reserved for them so as to minimize repeat coverage. An example of a group creating and moving in a cell with a virtual frontier is in Fig.5.

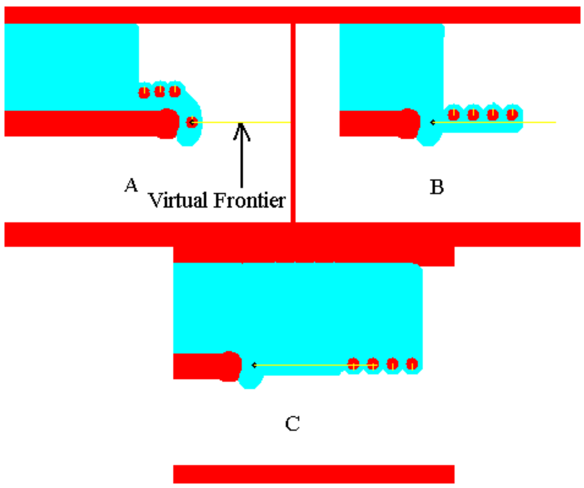

Figure 5: Creation of a Virtual Frontier: (A) Robots encountering a convex forward critical point, (B) Following the virtual frontier, and (C) Stopping at the next corridor of the sub-cell at the virtual frontier.

The final case Fig.4(iii) occurs at a concave reverse critical point. This case is trivial; the team naturally and simply divides at the corridor boundary because there is no more space for robots to fit!

3.4 Moving to Uncovered Cells

When a cell is completed, the team must move to an uncovered cell in its adjacency graph. The only reasonable route to an unexplored cell is through known covered space. It is worth noting that if a team covering a cell encounters another group that is in transition through the cell, then the covering team can conclude that the cell it is covering is already covered and then join the traveling team. In choosing a new cell to cover, we use some heuristics using a priority stack based on the history of a team's splits. These heuristics are the subject of current work right now, but something as simple as picking the closest cell or the cell that maximizes the likelihood of re-encountering a team has been investigated.

3.5 Team Rejoining

The meeting of two robot teams, each of which is covering the space separately, is almost certain to occur in a space with obstacles. When teams meet while sweeping, they combine their adjacency graphs, complete the corridor they were each initially covering, and merge. If the two teams were covering in opposite sweep directions, they have together completed the entire current cell. If the two teams were covering in the same sweep direction, the merged team continues covering the same cell.

If both teams are moving through covered space to an unexplored cell, they can pick either destination after merging. If a sweeping team meets a team moving to an unexplored cell, the teams merge and choose a new cell to cover. This is because, as stated in §3.4, the only route for a team to move to an unexplored cell is through known covered space. Hence the team moving has already covered the cell the sweeping team is currently covering.

4 Simulation Example

Figure 6 shows a team of four robots covering an unknown space. In this figure, the team discovers a reverse convex critical point and experiences its first division.

The team discovers its second reverse convex critical point 7 and experiences its second divion. There are now three separate teams covering the space.

In Figure 8, the teams are still split into three sub-teams. The lone robot in the top cell is re-covering the cell because it has not learned that the two-robot team has covered it. Meanwhile the two-robot team has passed by the second forward convex critical point, but has not yet discovered it. Meanwhile, bottom single robot team has covered a bottom cell and has discovered a forward convex critical point, and thus introduced the virtual frontier. Alas, this single robot will meet the two-robot team. When this happens, they "dance" around each other so that they can complete the corridors (not leave any space uncovered). Next, the two robot team does its reverse boundary-following motion to complete coverage of the corridor and hunt for the critical point (at this point, however, the two-robot team does not know there is a critical point. Next, the two-robot team moves up and finds the critical point, at which point it completes coverage of the top cell, updates its adjacency graph, and then forgets that it met the other robot (future work will optimize this to allow for this meeting) because the single robot rendezvous occurred "after" the critical point. The single robot then continues to cover the bottom sub-cell below the virtual frontier.

Finally, Figure 9 indicates the final state of the robots after coverage is complete. Note that, after covering the space to thr right of the starting position, the team went back to cover the remaining corridor to the left of the team's starting postion.

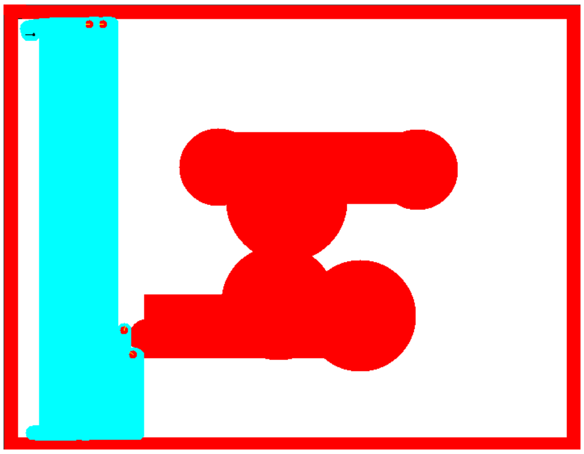

Figure 6: Robot team discovers first critical point and divides

5 Conclusions

This paper presents a new algorithm for complete coverage with a team of multiple robots. The use of multiple robots clearly expedites the coverage operation because many robots can cover in parallel. The

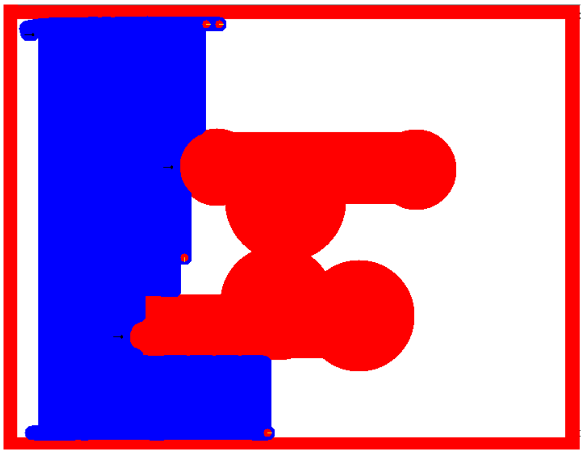

Figure 7: The team discovers its second critical point and again divides

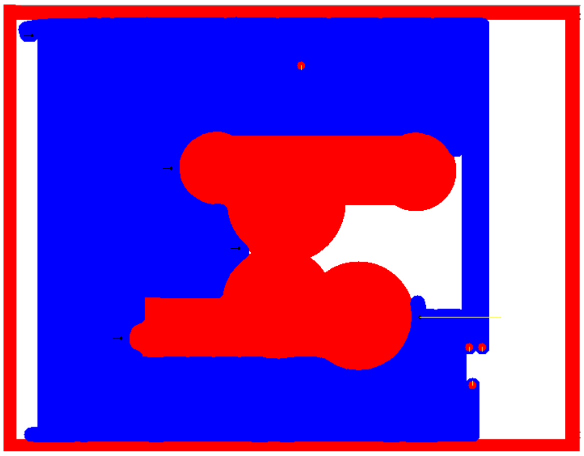

Figure 8: Robot teams rejoin

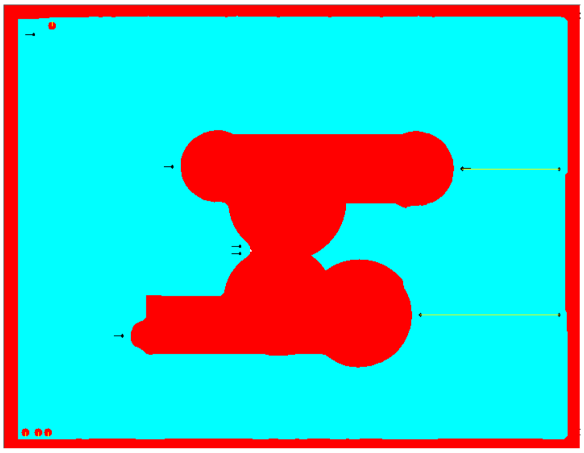

Figure 9: The entire space is covered

central issue which we address in this paper is when to merge and divide teams. Our algorithm uses critical points of a slice that is swept through the robots' work space. Here, we were borrowing ideas from previous work in single robot coverage that uses a cellular decomposition to divide the robots' free space into regions called *cells* such that simple back-and-forth farming motions are sufficient to cover a cell. Once the single robot visits each cell, coverage is complete because coverage within a cell is trivial.

In terms of coverage, we make two contributions: how to cover a cell and how to detect critical points. We first extend the notion of how to cover a cell by generalizing the notion of back-and-forth motions to motion in a valid formation. A valid formation, in a sense, is robots moving back-and-forth, like the single robot, but in parallel. They can all move maintaining a horizontal formation, like a rake, or move in an echelon formation, if desired. The second contribution is distributed critical point detection, which coordinates the robots to simultaneously look for critical points more quickly than the single robot version of the algorithm.

Ultimately, our goal was to make coverage more efficient in terms of decreasing the time to cover per unit time. We can also measure efficiency in terms of area that is re-covered by robots. The more repeat coverage, obviously the less efficient the approach is. Unfortunately, we cannot prove rigorous efficiency improvements because no matter what algorithm we determine, we can also construct an antagonistic environment that breaks it. However, we should note that repeat coverage tends to occur when the robot team is traveling from cell to cell or along the boundary of a cell. This occurs on a one-dimensional subset of the free space whereas a bulk of the coverage occurs on a two-dimensional set. Therefore, we believe that efficiencies that can be gained with multiple robots with the *optimal* approach, if one existed even for known environments, will only be second order.

We believe, however, that this work highlights issues that go beyond coverage and that is answering the question of when a robot team should divide or merge. In this work, we use a topological change, i.e., the existence of critical points to make this decision. Future work will look at other topological structures and how they effect other tasks

In the near-term, we believe that several extensions of this method can be envisioned. The heuristics that we have chosen are based on common sense and have not been validated against other potential heuristics. Future work should investigate other interesting heuristics. This method utilizes ubiquitous communication between all robots within a group. However, extensions can be envisioned to handle line of sight or other limited communication factors. These extensions would involve extra states where communication needs to occur. Additional methods for non-contact based coverage can be devised. Inherently this method conceptualizes the robot's sensor range to be equal to the robot's physical footprint. Increases in efficiency or robustness can be achieved through different optimizations of overlapping sensor ranges.

6 Acknowledgements

Siddhartha Srinivasa was supported in part by the National Science Foundation through grant IIS-9820180 to Michael Erdmann.

References

[1] E. Acar and H. Choset. Critical point sensing in unknown environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2000.

[2] E. Acar and H. Choset. Robust sensor-based coverage of unstructured environments. In *International Conference on Intelligent Robots and Systems*. IEEE/RSJ, 2001.

[3] T. Balch and R. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):292–352, 1998.

[4] Z. Butler. cc_R : A complete algorithm for contact-sensor based coverage of rectilinear environments. Technical Report CMU-RI-TR-98-27, The Robotics Institute, Carnegie Mellon University, October 1998.

[5] Z. Butler, A. Rizzi, and R. Hollis. Distributed coverage of rectilinear environments. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, January 2001.

[6] J. Canny. Constructing roadmaps of semi-algebraic sets i : Completeness. *Artificial Intelligence*, 37:203–222, 1988.

[7] J. Canny and M. Lin. An opportunistic global path planner. *Algorithmica*, 10:102–120, 1993.

[8] H. Choset and P. Pignon. Coverage path planning : the boustrophedon decomposition. In *Proceedings of the International Conference on Field and Service Robotics*, Canberra, Australia, December 1997.