2018

Multi-Robot Complete Coverage Using Directional Constraints

Stefanus Malan
Nova Southeastern University, sg_malan@yahoo.com

Multi-Robot Complete Coverage Using Directional Constraints

by

Stefanus G. Malan

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computer Information Systems

College of Engineering and Computing
Nova Southeastern University

2018

We hereby certify that this dissertation, submitted by Stefanus Malan, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Sumitra Mukherjee, Ph.D.
Chairperson of Dissertation Committee

11/26/2018
Date

Francisco J. Mitropoulos, Ph.D.
Dissertation Committee Member

11/20/2018
Date

Michael J. Laszlo, Ph.D.
Dissertation Committee Member

11/26/2018
Date

Approved:

Meline Kevorkian, Ed.D.
Interim Dean, College of Engineering and Computing

11/26/18
Date

College of Engineering and Computing
Nova Southeastern University

2018

An Abstract of a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Multi-Robot Complete Coverage Using Directional Constraints

by
Stefanus G. Malan
November 2018

Complete coverage relies on a path planning algorithm that will move one or more robots, including the actuator, sensor, or body of the robot, over the entire environment. Complete coverage of an unknown environment is used in applications like automated vacuum cleaning, carpet cleaning, lawn mowing, chemical or radioactive spill detection and cleanup, and humanitarian de-mining.

The environment is typically decomposed into smaller areas and then assigned to individual robots to cover. The robots typically use the Boustrophedon motion to cover the cells. The location and size of obstacles in the environment are unknown beforehand. An online algorithm using sensor-based coverage with unlimited communication is typically used to plan the path for the robots.

For certain applications, like robotic lawn mowing, a pattern might be desirable over a random irregular pattern for the coverage operation. Assigning directional constraints to the cells can help achieve the desired pattern if the path planning part of the algorithm takes the directional constraints into account.

The goal of this dissertation is to adapt the distributed coverage algorithm with unrestricted communication developed by Rekleitis et al. (2008) so that it can be used to solve the complete coverage problem with directional constraints in unknown environments while minimizing repeat coverage. It is a sensor-based approach that constructs a cellular decomposition while covering the unknown environment.

The new algorithm takes directional constraints into account during the path planning phase. An implementation of the algorithm was evaluated in simulation software and the results from these experiments were compared against experiments conducted by Rekleitis et al. (2008) and with an implementation of their distributed coverage algorithm.

The results of this study confirm that directional constraints can be added to the complete coverage algorithm using multiple robots without any significant impact on performance. The high-level goals of complete coverage were still achieved. The work was evenly distributed between the robots to reduce the time required to cover the cells.

Acknowledgements

First and foremost, I would like to thank God Almighty for giving us life, the ability to learn, and the gifts of creativity.

I am so grateful for the support of my wife, Corrie, during our more than twenty years of marriage and her willingness to give up so much to make it possible for me to pursue my life goals. I am also grateful to my children, Chaney and Michelle, who are an inspiration for working so hard towards their own academic achievements and goals. This dissertation would never have succeeded without all your sacrifice and support.

A special thank you to my advisor and committee chair, Dr. Mukherjee. He is always willing to listen, help, and guide. The idea for this research started during the course work and Dr. Mukherjee helped shaped the direction of the work by giving valuable feedback from the start. His incredible knowledge and ability to recall papers are so valuable. Even though he is scary smart, he is always friendly, approachable, and helpful.

I am indebted to my dissertation committee for their willingness to support my research, their time, and guidance. A big thank you to my committee members Dr. Laszlo and Dr. Mitropoulos. I appreciate every interaction, lesson, and feedback that was provided during the course work and research. Their guidance throughout this journey will always be remembered.

I would like to thank Maggie Prince for volunteering her professional skills and making time to proof read all the documents. Not only did she provide valuable feedback, she always explained why a change is needed. I am fortunate to have such a great friend and teacher in my life.

To all the faculty and staff at NSU, thank you. Without them, it would not be possible for students to learn and grow.

Finally, I would like to thank my family, friends, fellow students, and colleagues for all their continued support and encouragement throughout this journey.

Thank you!

Table of Contents

List of Tables

Tables

List of Figures

Figures

Chapter 1

Introduction

Background

Many real-life applications require one or more robots to cover an unknown environment. The task of path planning for complete coverage is required by several robotic applications. The importance of complete coverage of an unknown environment is highlighted by their use in applications like automated vacuum cleaning, carpet cleaning, lawn mowing, chemical or radioactive spill detection and cleanup, and humanitarian de-mining (Rekleitis, Lee-Shue, New, & Choset, 2004).

Complete coverage guarantees that all of the accessible area of the environment is covered. The coverage path should minimize the time required to cover the area while avoiding obstacles (Huang, 2001). The goal is to plan a path that guides one or more robots to pass an end-effector (a sensor or actuator, but in most cases, the body of the mobile robot) over the entire area to achieve complete coverage while minimizing repeat coverage and avoiding obstacles (Rekleitis, New, Rankin, & Choset, 2008).

Several single-robot coverage methods exist that guarantee complete coverage of an unknown environment. Most of these single-robot coverage planners use cellular decomposition. With exact cellular decomposition, the environment is divided into non-overlapping cells. Complete coverage is achieved by ensuring that the robot visits every cell. The robot moves in a simple back-and-forth motion to cover the cells with this strategy (Rekleitis et al., 2008).

Rekleitis et al. (2008) introduce two algorithms for the complete coverage planning problem using a team of mobile robots when unlimited communication is available. The coverage task is divided into two stages: the exploration and partial coverage, and the collaborative coverage of the uncovered cells.

For some applications, the goal is to cover the environment using different patterns (Choset, Acar, Rizzi, & Luntz, 2000). Examples where patterns are used to cover the environment include spray painting and lawn mowing. A pattern-based approach is important when robotic lawnmowers are required to mow a lawn in such a way that a pattern is visible when the task is complete.

Scag Power Equipment (2010) lists some of the common stripe patterns. The basic stripe pattern is shown in Figure 1. This pattern is achieved by mowing alternate adjacent paths in the opposing direction. This provides the most contrasting stripe effect.

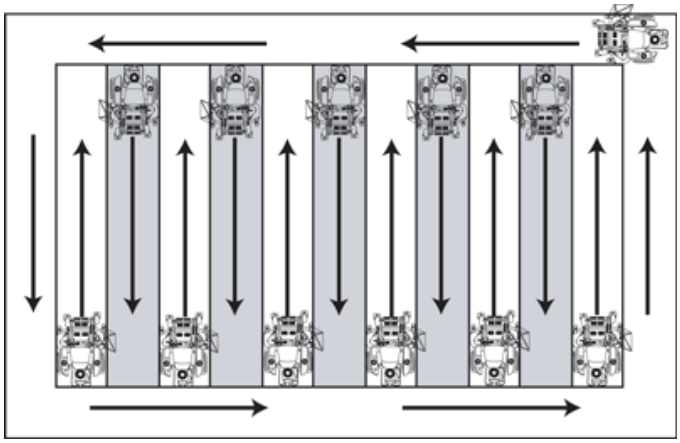

Figure 1. Basic lawn stripe pattern. *Retrieved from http://www.scag.com/images/stripingart-basic.gif*

When a specific pattern is used to mow a lawn, it is usually for visual appeal or to increase the life of the grass. Lawn stripes are made by bending the grass blades in different directions and not by cutting at different heights or using different breeds of

grass. Grass blades that are bent away from the viewpoint appear lighter in color. The reason for this is that more light is reflected off the wide part of the blade. On the other hand, blades bent toward the viewpoint appear darker since the tip of the blades has a smaller area to reflect light. The grass blades are bent with a roller that is usually attached to the back of the lawnmower. With the roller bending the grass in the same direction that the robot travels, the travel direction becomes important (Hameed, Sorrenson, Bochtis, & Green, 2011).

When the task to cover an environment with a specific pattern is assigned to multiple robots, directional constraints are introduced to the path-planning process. Not only should an optimal coverage algorithm create a path that minimizes the time required to execute the plan, it should also enforce directional constraints to cover the environment with the selected pattern.

Problem Statement

Complete coverage of an unknown environment is usually achieved by decomposing the environment into cells and then assigning the cells to individual robots to cover. The robots proceed to cover the cells with a simple back and forth movement until all the cells are covered (Rekleitis et al., 2008). There is not a complete coverage algorithm that imposes directional constraints for the movement of the robots based on a specified pattern.

To generate a specific pattern on lawns with robotic lawnmowers, the route planning algorithm will have to determine a path that satisfies certain directional constraints. For example, alternating the travel direction for each cell (up/down,

left/right, north/south, etc.) provides the most contrasting stripe effect (Hameed et al., 2011).

The distributed coverage algorithms developed by Rekleitis et al. (2008) that use multiple robots with unlimited communication cannot be directly applied to the lawnmower problem that imposes directional constraints.

Dissertation Goal

The main goal of the study is to adapt the distributed coverage algorithm developed by Rekleitis et al. (2008) so that it can be used to solve the complete coverage problem with directional constraints in unknown environments while minimizing repeat coverage. The proposed algorithm will allow robotic lawnmowers to plan a path that will not only guarantee complete coverage but will also mow the lawn so that the specified pattern is visible after the task is complete.

The proposed solution will be evaluated experimentally using benchmark problem instances. Predefined tests with a specific environment size and specified obstacles will be used. The results from using the distributed coverage algorithm by Rekleitis et al. (2008) and the proposed algorithm to complete the tests will be compared.

According to Mannadiar and Rekleitis (2010), the algorithm is correct if each edge of the graph is covered, that is, all available free space has been covered. The algorithm is optimal if all free space is covered exactly once. Repeat coverage may be required if the directional constraint guides the robot to a dead-end.

The target environment in which the robots operate in can be of any shape. The shape of the environment does not make a difference in the approach. However,

consistent with previous studies, a rectangle with known dimensions is used to simplify the representation.

Research Questions

To address the problem statement and achieve the dissertation goal, the following questions will be used to guide the study:

1. Can repeat coverage be minimized when the directional constraint guides the robot to a dead-end?
2. Is the algorithm fast and efficient enough so that the robots can perform the path planning on-line while the coverage task is in progress?

Relevance and Significance

Improved algorithms are needed for the complete coverage path planning problem in an unknown space (Acar and Choset, 2000; Oh, Park, and Choi, 2001; Butler, 1998; Solanas and Garcia, 2004; Rekleitis et al., 2008). Several algorithms exist that work with a team of robots (Rekleitis et al., 2008; Batalin and Sukhatme, 2002; Dias and Stentz, 2001; Yong, Zhang, and Zhang, 2008; Wang and Syrmos, 2009). Most of these algorithms use exact cellular decompositions to identify the cells to be covered by individual robots, but they fail to provide the ability to meet requirements on direction of coverage.

Barriers and Issues

The goal of the multi-robot coverage task is to assign work to each robot in such a manner that the time required to complete the coverage task is minimized. All the robots should finish covering the area assigned to them roughly at the same time. The time to

complete the coverage task is the time it takes the last robot to complete coverage of the last uncovered area. According to Rekleitis et al. (2008), determining the optimal solution that minimizes the travel time is an NP-hard problem. An optimal coverage algorithm would minimize the time required to execute the coverage path (Huang, 2001). Adding another constraint to the complete coverage problem will make the planning algorithm more complex.

Assumptions, Limitations and Delimitations

Assumption similar to Acar and Choset (2000) and Rekleitis et al. (2008) are made. These assumptions include:

1. The interior of the area to be covered is unknown and that it is partially occupied by obstacles.
2. The robots are identical and their orientation and exact location is known at any given time.
3. The end-effector is assumed to be of the same size as the footprint of the robot.
4. The robots travel at the same speed so that the same distance is covered in one unit of time.
5. The robots can sense obstacles that lie in their path. Sensors are used to detect obstacles, nearby robots, and position as they move through the environment.

6. The robots communicate with each other in order to avoid collisions, coordinate with each other to reduce repeat coverage, and minimize coverage time.

7. Unrestricted communication exists between the robots.

Definition of Terms

The following list of terms is key terms in the field of complete coverage. They are used throughout this document.

Auctioneer

The robot calling an auction is called the auctioneer (Rekleitis et al. (2008).

Boustrophedon

The first time the word "boustrophedon" was used in the English language was in 1699. It comes from ancient Greek and it literally means "the way of the ox" (Choset & Pignon, 1998, p. 203).

Critical Points

The point of an obstacle that is detected is known as a critical point. Critical points represent topologically meaningful events in the environment (Acar & Choset, 2000).

Complete Coverage

A coverage algorithm is complete when the robot pass over every point in the environment when the planned path is followed (Choset & Pignon, 1998).

Monotone Polygon

A polygon P is monotone if there exists a straight line ℓ that can partition the boundary into two chains that are monotone with respect to ℓ. Every line that is orthogonal to ℓ should intersect P at most twice. Certain computational problems are easier using monotone polygons than arbitrary simple polygons (Preparata & Supowit, 1981).

Repeat Coverage

> Repeat coverage is defined as "any robot covering previous covered space" (Rekleitis, Lee-Shue, New, & Choset, 2004, p. 3462).

Summary

Most algorithms for complete coverage use exact cellular decompositions to identify the cells to be covered by individual robots. An improved algorithm is needed for the complete coverage path-planning problem in an unknown environment that will include directional constraints. This research propose a new path-planning algorithm for multi-robot complete coverage that will take directional constraints into consideration.

Chapter 2

Review of the Literature

Introduction

This chapter represents a literature review relevant to complete coverage algorithms and establishes the background for this research. It begins with an overview of coverage and then review several different approaches that can be used to address the complete coverage problem. It also looks at the differences between single-robot and multi-robot path planning algorithms.

Overview of Coverage

A robot's motion can be classified as either point-to-point or area sweeping depending on its motion objective. Point-to-point is defined as a motion to perform a task while moving from the start to the goal position, while area sweeping is defined as a motion to traverse the environment in such a way so that the robot's sweeping device (sensor or actuator) will cover the entire area (Min & Yin, 1998).

"Coverage Path Planning (CPP) is the task of determining a path that passes over all points of an area or volume of interest while avoiding obstacles" (Galceran & Carreras, 2013, p. 1258). This is the main task for several robotic applications such as vacuum cleaning robots, painter robots, autonomous underwater vehicles, lawn mowers, automated harvesters, or window cleaners.

Several coverage approaches and algorithms are discussed in the "Coverage Algorithms" section. The distributed multi-robot coverage algorithm by Rekleitis et al.

(2008) is discussed in the "Multi-robot Coverage" section. In the "Approach" section, the proposed approach that build on the distribute coverage algorithm by Rekleitis et al. (2008) is discussed.

Coverage Algorithms

Coverage algorithms should allow robots to pass a sensor or actuator over a given area in an efficient way while guaranteeing coverage of the entire area. The coverage path should minimize the time required to cover the area (Huang, 2001).

Choset (2001) classifies the algorithms for coverage path planning into four categories: heuristic approaches, approximate cellular decompositions, semi-approximate, and exact cellular decompositions. The guarantee that the region will be completely covered is one of the major accomplishments of several recent works in coverage.

Random Path Planning

There are scenarios in which random movement is a valid approach to solve the problem. The idea behind this approach is that if the robot moves randomly for long enough, the area will be covered. The advantage to this approach is that no complex sensors or expensive computation resources are needed (Galceran & Carreras, 2013).

Robots without localization capabilities that use randomized search strategies can be built more cheaply than robots with more precise positioning systems. In some situations, it may be effective to use robots with randomized search algorithms. However, random search does not guarantee complete coverage. In order to generate the

path for more than one robot, accurate localization capabilities are required (Choset, 2001).

Grid-based Decomposition

A fine resolution occupancy grid can be used to represent the decomposed area. Each cell in the grid will be the size of the robot. When the centroid of a grid cell is reached the cell is deemed to be covered. However, partially occupied grid cells and the coverage paths of multiple robots are two problems that are difficult to coordinate and optimize. Rekleitis et al. (2008) believe that it is not efficient to decompose the coverage task using such a fine grid.

Line-sweep Decomposition

The sweep direction is the same for all subregions when line-sweep decomposition is used. A simple back and forth motion perpendicular to the sweep direction is used to cover each subregion. This coverage path is a detailed sequence of motion commands for the robot to perform the task. Finding the optimal sweep direction can reduce the time required to cover the area. One way to accomplish this is to minimize the number of turns. For every turn the robot must slow down, turn, and then accelerate. (Huang, 2001).

Cellular Decomposition

Robots equipped with sensors can sense obstacles that lie in their path. The point of an obstacle that is detected, also known as a critical point, is used to subdivide the cells (Acar & Choset, 2000). More cells are formed as obstacles are detected in the environment. Choset (2000) defines cellular decomposition and coverage as breaking the

free space into cells at the critical points and then covering the cells with back-and-forth boustrophedon motions.

Approximate Cellular Decomposition

Many path-planning algorithms use cellular decomposition to achieve complete coverage. Cellular decomposition algorithms break down the target region into cells so that the coverage of each cell is simple. By visiting each cell of the decomposition, complete coverage can be achieved. A fine-grid is used for approximate cellular decomposition. The target region is divided into cells that have the same size and shape. The size of the cells is typically the same size as the robot's footprint or effector (Choset, 2001).

Exact Cellular Decomposition

A great number of complete coverage algorithms use exact cellular decomposition. Many of the cell coverage algorithms use "simple back-and-forth motions" (Rekleitis, New, Rankin, & Choset, 2008, p. 113).

"An exact cellular decomposition is the set of non-intersecting regions, each termed a cell, whose union fills the target environment" (Choset, 2001, p 118). The exact cellular decomposition approach is an enhancement of trapezoidal decomposition. Each cell in the trapezoidal decomposition is a triangle or trapezoid instead of the equal square cells. The cells are typically covered with a simple back-and-forth motion. By doing this, the coverage path planning is reduced to the planning for motions between cells (Choset, 2001).

The path-planning algorithm presented by Oksanen and Visala (2009) will prohibit certain driving directions when direction limitations are set. Regions with restricted driving directions are handled as obstacles or interior polygons. Driving directions that fall in the range of restricted angles are prohibited. One of their simulations show that total driving time increased by only 0.2% when directional constraints are taken into account. Some of the drawbacks from their algorithms include:

1. only straight driving lines can be used,
2. it will always find suboptimal solutions, and
3. the algorithms are not geared for multiple robots.

Boustrophedon Cellular Decomposition

Boustrophedon means the way of the ox and boustrophedon motion the back and forth ox-like motions (Choset & Pignon, 1998). When an ox plows a field, it drags a plow across the full length of the field in a straight line, turns around, and then plows a new straight line adjacent to the previous one. The ox is guaranteed to cover the entire field by repeating the procedure (Choset, 2000).

The Boustrophedon cellular decomposition is a type of exact cellular decomposition approach. It is designed to minimize the number of lengthwise motions compared to the trapezoidal decomposition approach (see Figure 2 and Figure 3). This is achieved by merging two cells into one so that fewer passes are required to cover the new monotone polygon (Choset, 2000).

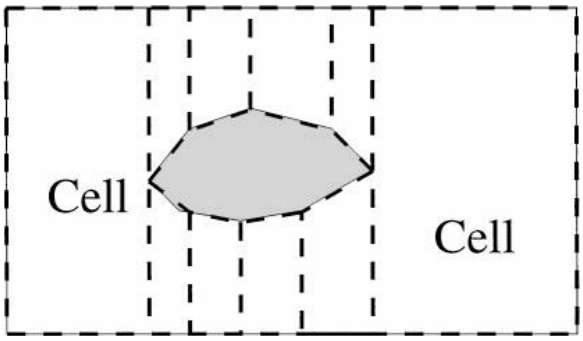

Figure 2. Trapezoidal decomposition. *Adapted from " Coverage of known spaces: The boustrophedon cellular decomposition" by H. Choset, 2000,* Autonomous Robots, 9(3), *p. 249. Copyright 2000 by Kluwer Academic Publishers*

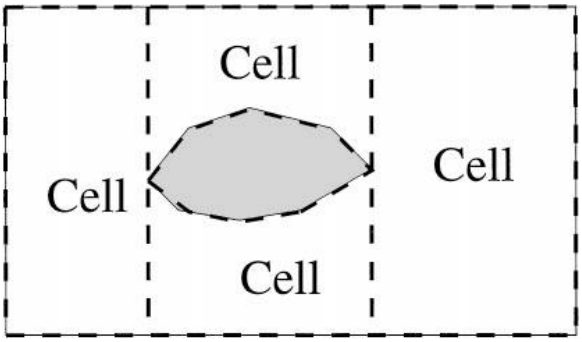

Figure 3. Boustrophedon. *Adapted from " Coverage of known spaces: The boustrophedon cellular decomposition" by H. Choset, 2000,* Autonomous Robots, 9(3), *p. 249. Copyright 2000 by Kluwer Academic Publishers.*

Spanning Tree Coverage

Senthilkumar and Bharadwaj (2012) presented an on-line algorithm that is based on the Spanning Tree Coverage (STC) technique for multiple robot coverage that is designed to cover the terrain in a complete and efficient manner. It is based on approximate cellular decomposition to achieve complete coverage.

Genetic Algorithm

Kapanoglu, Alikalfa, Ozkan, and Parlaktuna (2012) proposed a single-stage, pattern-based genetic algorithm for multi-robot sensor-based coverage. The object of

their algorithm is to minimize completion time. The routing and partitioning of the coverage area is performed concurrently among the robots. This approach is superior to the two-stage hierarchical genetic algorithm where the first stage finds a single route that minimizes repeat coverage and then partitions the route based on actual travel time costs and assigns it to the robots.

Single-Robot Coverage

Most single robot coverage planners use cellular decomposition. With exact cellular decomposition, the environment is divided into non-overlapping cells with the so-called sweeping line strategy, or Boustrophedon path. The robot moves parallel to the given sweeping inclination in a back and forth motion to cover the environment with this strategy. Inclination refers to the orientation of the scan line (Yao, 2006). Complete coverage is achieved by ensuring that the robot visits every cell (Rekleitis et al., 2008). Several existing algorithms use the traveling-salesman algorithm to determine the sequence in which the robot should visit the cells (Huang, 2001).

Multi-Robot Coverage

The algorithmic solutions by Rekleitis et al. (2008) for the complete-coverage-path-planning problem use a team of mobile robots. The coverage algorithm for a single robot is extended for the multi-robot algorithm. This is achieved by algorithmically decoupling the exploration and coverage phases. An overseer algorithm was added to the single robot algorithm to produce cooperative coverage. The overseer integrates data from other robots into the cellular decomposition. The algorithm uses simple back-and-forth motions to cover the cells assigned to the robots.

The use of multiple robots for coverage is motivated by efficiency and robustness. The task can be completed faster with multiple robots by dividing the environment between them. When a robot fails, multi-robot algorithms may still succeed. The reason why this type of algorithm may still succeed, is that the robot's peers might still cover the cell or cells assigned to the failed robot (Hazon, Kaminka, 2008).

Using multiple robots not only decreases the time required, but increases efficiency and robustness. Multiple robots also increase the complexity of the algorithms used. When unlimited communication between robots is available, the robots can cover different areas of the environment while constantly updating each other on their progress (Rekleitis et al., 2008).

Distributed Coverage Algorithm

The distributed multi-robot coverage algorithm by Rekleitis et al. (2008) performs complete coverage of an unknown environment with exact cellular decomposition using a group of robots. The environment is divided into smaller areas so that the workload is evenly distributed between robots when looking at the total distance travelled by each robot. The number of robots is used as the denominator to divide the environment into smaller areas. Each robot is assigned an area to cover. The robots are deployed at regular intervals along one side of the environment, as depicted in Figure 4.

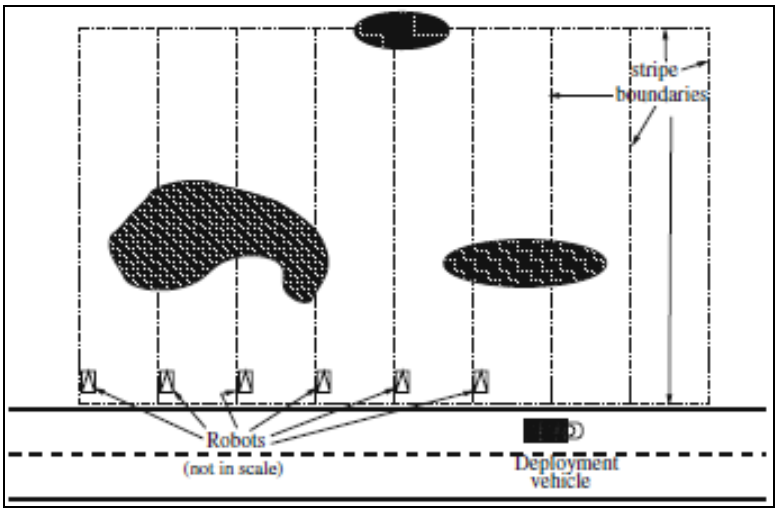

Figure 4. Robots are distributed. *Adapted from "Efficient boustrophedon multi-robot coverage: an algorithmic approach" by I. Rekleitis, A. New, E. Rankin, and H. Choset, 2008,* Annals of Mathematics and Artificial Intelligence, 52(2-4), *p. 112. Copyright 2009 by Springer Science + Business Media.*

Each robot will try to trace the outline of the assigned area to determine the layout of the free space. The connectivity is known but not necessarily the inside of the area. Areas may be divided into two or more. A Reeb graph is used to map the connectivity of the space during the exploration. The robots share the information with the other robots (Rekleitis et al., 2008).

The Reeb graph is constructed with the nodes representing the intersections of the sweeping line with the boundary and objects in the environment in the same way as the typical complete coverage problem. The robots use this graph to determine the optimal path to cover the area. To calculate the order in which the cells are going to be covered, either the Chinese Postal Problem or Eulerian tour can be used (Mannadiar & Rekleitis, 2010).

Some robots may detect that there exists an unreachable space in their assigned area while moving around in it. These areas will be assigned to other robots to explore.

The robots will then cover the cells they own using a single-robot coverage algorithm. Robots that finish their task can offer its service to the other robots. A market-based methodology is used to assign areas and resources. When a robot discovers a cell that needs to be covered, the robots call an auction with an initial estimate to complete the task. Robots that are free can bid for the task. When the auction ends, the task is assigned to the robot with the lowest bid. This prevents two robots from starting to cover the same cell. When all the robots complete their cell coverage and no uncovered cells exists in the Reeb graph, they return to their starting positions and declare the environment covered (Rekleitis et al., 2008).

Time estimates play a big role in determining the size of the area that is assigned to each robot. The first robot that encounters an obstacle makes at least one complete circuit around it (Huang, 2001). This action adds additional time it takes the robot to cover the area assigned to it. The time required to complete some of the required actions can be determined ahead of time but the time it takes to circle around the detected obstacles will only be known afterward due to the random nature of the obstacles.

Finer Granularity Distributed Coverage Algorithm

This algorithm decomposes the coverage problem into finer resolution tasks. Grid-based decomposition is used to decompose the area. The cells are defined differently from the first algorithm. The width of the cells is twice the width of the robots' footprint as shown in Figure 5. An adjacency graph is used to represent the cells and is of finer resolution than the Reeb graph. Each node in the graph represents a cell and the edges represent the connectivity between the cells. The coordinates of the four corners of the cell and an indicator whether the cell is covered or not are stored at each node.

The robots are distributed the same as before (along one side of the environment) but each robot is just assigned a single cell instead of an area. After covering a cell, the robot will update its internal adjacency graph, broadcast the information to the other robots, select the closest uncovered cell, notify the other robots of its selection, and then move to the selected cell (Rekleitis et al., 2008).

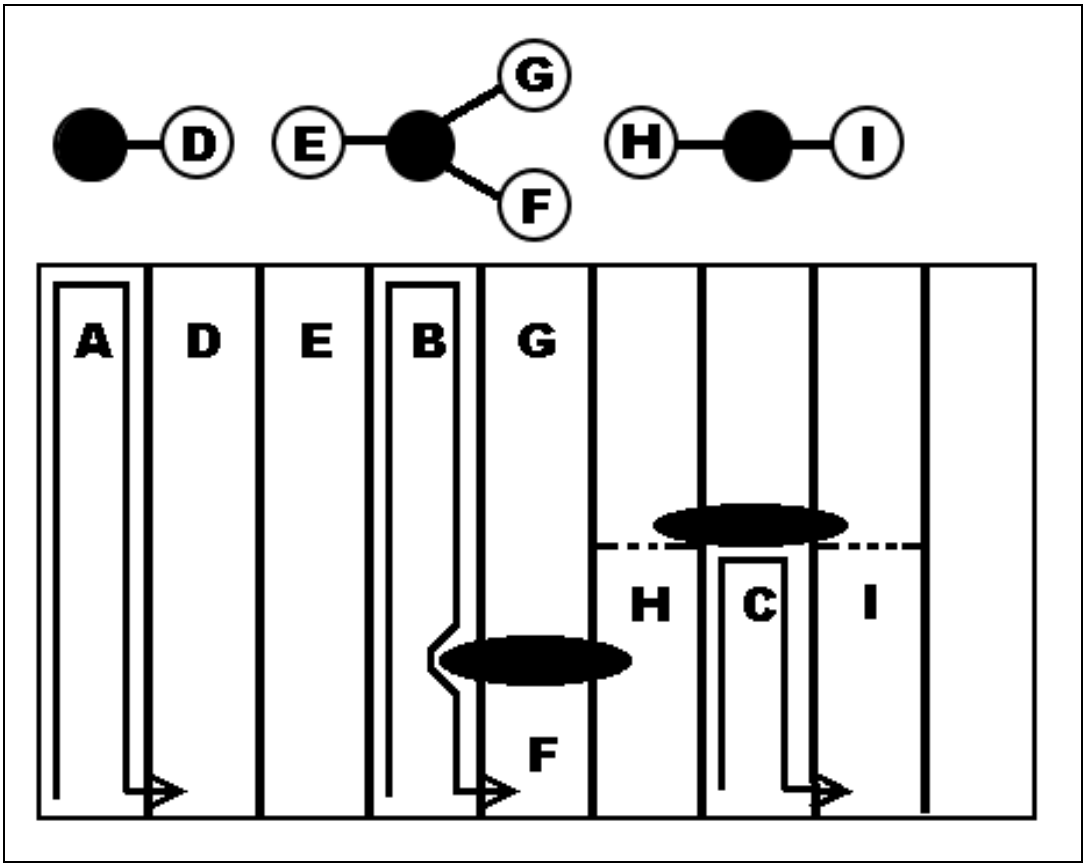

Figure 5. A cellular decomposition with fixed size cell width. *Adapted from "Efficient boustrophedon multi-robot coverage: an algorithmic approach" by I. Rekleitis, A. New, E. Rankin, and H. Choset, 2008,* Annals of Mathematics and Artificial Intelligence, 52(2-4), *p. 128. Copyright 2009 by Springer Science + Business Media.*

Summary

This chapter gave a comprehensive review of relevant literature pertaining to path planning for several different coverage problems and how it can be used for complete coverage. In the next chapter, the methodology used in this research is presented as well as the algorithm created for the research.

Chapter 3

Methodology

Introduction

This chapter explains the methodologies that were used in this research. The proposed algorithm for multi-robot complete coverage using directional constraints is described in more detail in the following sections. Details about the experimental design, the specific code implementation of the algorithm, and the validation process is also provided.

Overview of Research Methodology

This research is an empirical study using simulation. Numeric data was collected from the different experiments. The data was subsequently analyzed in order to answer the research questions.

This study extends the distributed coverage algorithm of Rekleitis et al. (2008). The new algorithm assigns a direction to each cell based on the selected pattern. The goal is to cover all the accessible area but use the directional constraints during the path planning process. A secondary goal is to distribute the workload between the robots so that the work is completed roughly at the same time by all the robots. The time and distance traveled were recorded during the experiments.

The results from the new algorithm were evaluated and examined to determine if the new algorithm restricts the movement of the robots based on the directional

constraints assigned to the cells. The results were also used to determine if the work was optimally spread between the robots.

Specific Research Methodology

A baseline was created by implementing the distributed coverage algorithm of Rekleitis et al. (2008), replicating the experiments, and comparing the results against their documented results. This algorithm by Rekleitis et al. does not have any directional constraints assigned to the cells and uses only a simple up-and-down movement to cover the individual cells. The basic stripe pattern is used so that the results from the experiments can be compared to the baseline data.

In order to be consistent with previous studies and to meet the stated objectives of this research, the following steps were followed for the new algorithm:

1. The environment with the objectives used by Rekleitis et al. (2008) was reproduced to make it possible to compare the new algorithm with existing methods.
2. The distributed coverage algorithm by Rekleitis et al. (2008) was recreated to collect the baseline data.
3. This algorithm was then modified to add the directional constraints.
4. Both the algorithm by Rekleitis et al. (2008) and the new algorithm were executed in simulation software and data was then collected.
5. The results were evaluated based on a set of predefined criteria as identified in the problem statement.

Player/Stage Robotic Simulation Package

The Player/Stage robotic simulation package was used for the experiments. The same environment and number of robots were used for both the algorithm created by Rekleitis et al. (2008) and the new algorithm. This made it possible to compare the results from the different algorithms and to evaluate the new algorithm constrained by the direction assigned to the cells.

The Player robot server and the Stage simulator form the Player/Stage system. This allows for research and rapid development of robot and sensor systems. The source code is released under the GNU General Public License and freely available from http://playerstage.sourceforge.net/. The Player server uses several abstraction layers that enable controller software to run unchanged on several different robot platforms. Player is based on a client/server model and uses TCP sockets to communicate. It is platform-, location-, and language-neutral. The Stage simulator simulates virtual Player devices that interact with Player just as the real device drivers. This allows development of control code in simulation that should work unchanged on real hardware (Vaughan, Gerkey, & Howard, 2003).

The simulation software is available from http://playerstage.org. There are also links available to documentation on the site. Player version 3.0.2 and Stage version 4.1.1 were used for this research on a computer running Linux as the operating system. The simulation system is started from the command line in a Linux terminal window. When Player is executed, the Stage graphical user interface is displayed. Player has the following usage:

```
player [options] [<configfile>]
```

The simulation software for the experiments in this study is started with the following command:

```
player experiment.cfg
```

Several files containing configuration and settings are used during startup. The different files and the hierarchy of these files that are unique for this study are shown in Figure 6.

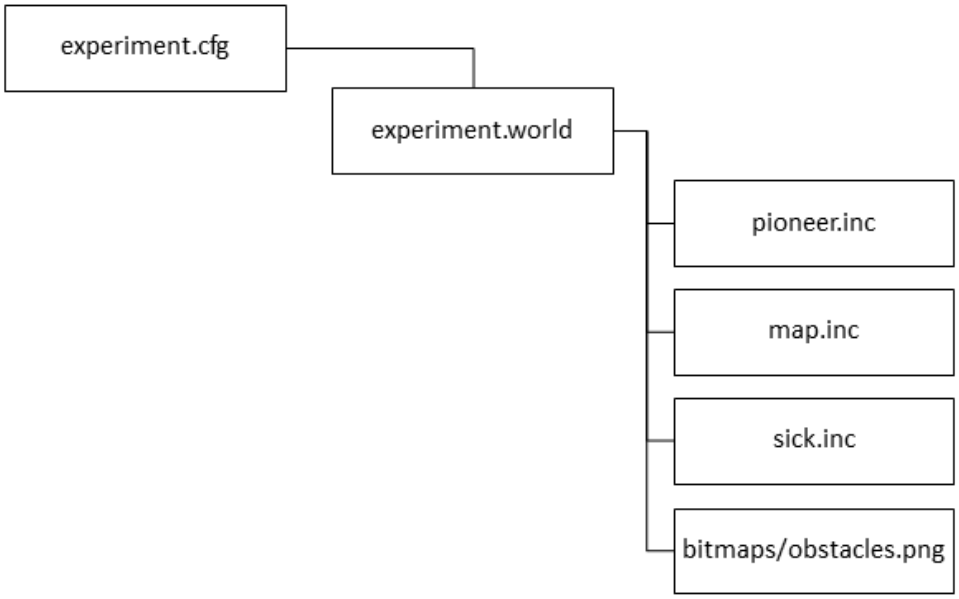

Figure 6. Files containing configuration and settings.

A few changes to the configuration files were required. The number of robots and the robot model info are defined in the `experiment.cfg` file. This file is the Player configuration file used for controlling Stage devices. The environment file (`experiment.world`) is specified in the "worldfile" section.

The size of the display window is defined in the `experiment.world` file. The size configuration (560 x 562) is the size of the window in pixels. The scale is the

number of pixels per meter and is set to 28.284. The size of the environment and the file name that should be used for the obstacles is set under the "floorplan" section. The size is set to [17.600 x 15.200 x 0.700] and the bitmap field is set to "bitmaps/obstacles.png". The robot type is also defined in this file.

The Pioneer 2-DX robot was used in the Player/Stage simulation framework. The specifications of the robot are defined in the `pioneer.inc` file and included from the `experiment.world` file. Examples of specifications include: dimension, mass (23kg), and differential steering. The "pioneer2dx" section in `experiment.world` is used for each of the robots with the localization set to "gps". This will report an error-free position in world coordinates so that the robots will get their position very accurately.

A few changes were required to the simulator source code to have the behavior shown in the work by Rekleitis et al. (2008). The way the footprints are drawn was changed in `Stage-master/libstage/model_draw.cc`. The second change was needed to increase the number of footprints that are displayed at any given time. The initial value (`uint32_t Model::trail_length(50)`) was changed from 50 to 1000 in `Stage-master/libstage/model.cc`. The g++ compiler is used to compile the source code to an executable binary file in combination with the make utility to install the software.

Experiments

The number of robots varied between the different simulation tests. This allows for comparisons to be made with extant algorithms that use one or more robots for the coverage task.

The purpose of the first set of experiments is to compare the reconstructed experiments to those conducted by Rekleitis et al. (2008) using the multi-robot coverage algorithm with unrestricted communication. No directional constraints were imposed on how cells should be covered. The number of robots was varied between three and five to determine if the algorithm can handle different conditions, even though the same environment was used for all the experiments. The number of cells assigned to each robot changed as the number of robots varies. As a result of the different cell assignments, the robots get different obstacles or pieces of the obstacles in the area assigned to them.

The second set of experiments was conducted with directional constraints. The basic stripe pattern was selected for the experiments. To get the desired pattern, a directional constraint is assigned to each cell when the environment is decomposed into cells. The path planning portion of algorithm plan paths that minimize violating the directional constraints by moving a robot only in the direction assigned to the cell it is traveling in where possible. The directional constraint can be violated to get out of a dead-end situation or when the extra distance required to satisfy the constraint is over a predefined threshold. Any distance traveled while violating the constraint are recorded and used to evaluate the algorithm.

Auction Mechanism

Global communication makes it possible to use an auctioning system to enable cooperation among the robots. A basic mechanism is used to determine which cells and areas are covered by the different robots. In most instances, only estimated distances are used since the robots may not have enough information available to accurately calculate the costs. Auctions are used in one of two ways: robots without any tasks and robots that have extra cells (Rekleitis et al., 2008).

Environment

A rectangular environment is used in the simulated environment. The size of the environment is known. The obstacles are static but their size and position is unknown a priori. Figure 7 shows the environment used by Rekleitis et al. (2008).

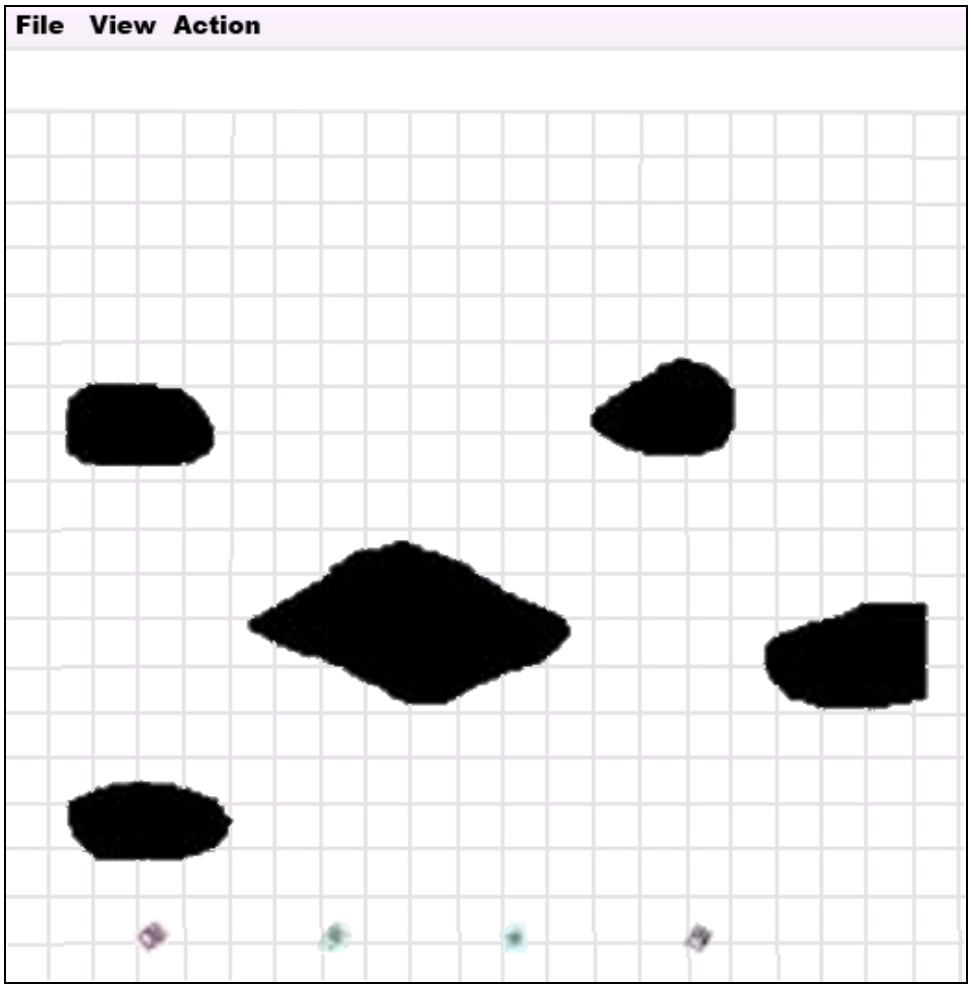

Figure 7. The environment and the four robots at the starting position in Stage. *Adapted from "Efficient boustrophedon multi-robot coverage: an algorithmic approach" by I. Rekleitis, A. New, E. Rankin, and H. Choset, 2008,* Annals of Mathematics and Artificial Intelligence, 52(2-4), *p. 136. Copyright 2009 by Springer Science + Business Media.*

To make it possible to compare results against the results from the experiments of Rekleitis et al. (2008), the environment with the obstacles was reproduced as accurately as possible. Figure 8. Created image of obstacles to match the obstacles in environment used for the experiments by Rekleitis et al. (2008). shows the recreated image with the obstacles that was used in the simulation software for the experiments. This image is loaded during startup of the simulation software. The location of this image is specified

for the "bitmap" variable under the floorplan section in "worldfile". In turn, the "worldfile" variable is defined in the `experiment.cfg` file and set to "experiment.world".

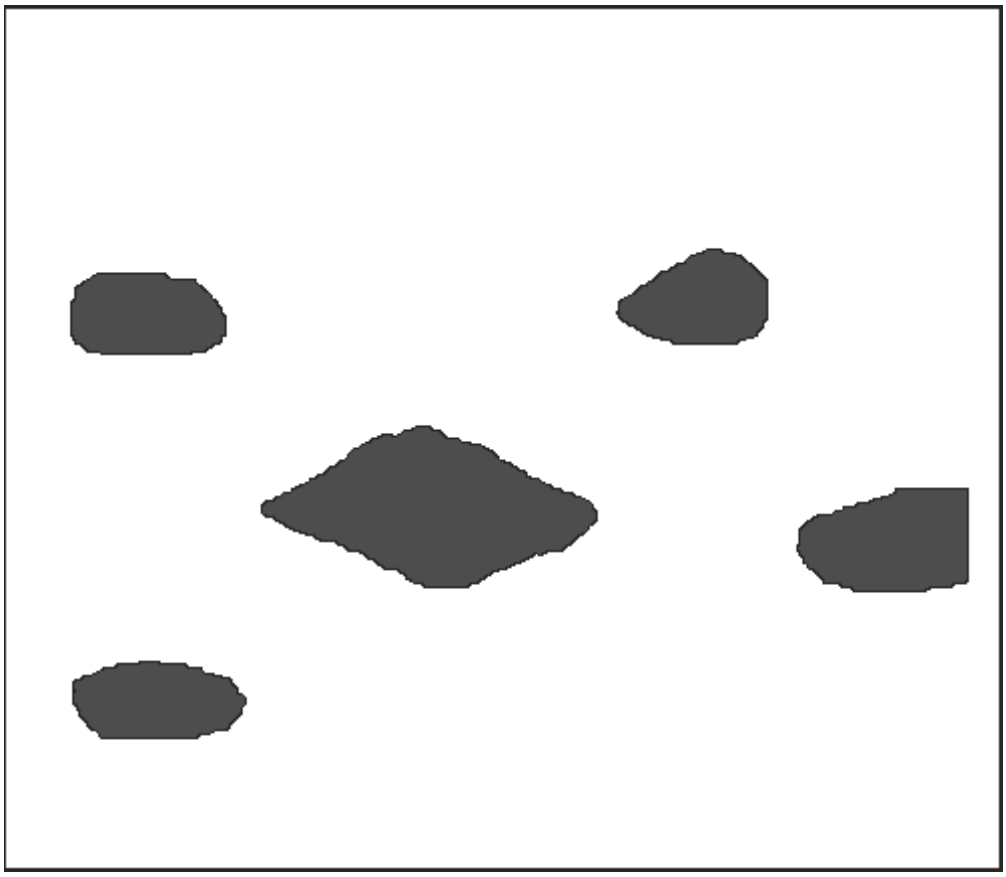

Figure 8. Created image of obstacles to match the obstacles in environment used for the experiments by Rekleitis et al. (2008).

Algorithm Design

Multi-Robot Coverage Problem without Directional Constraints

The distributed coverage algorithm with unlimited communication by Rekleitis et al. (2008) uses an adjacency graph to represent the decomposition of the environment. It has a finer resolution than the Reeb graph but is conceptually the same. The cells are

represented as the nodes and the connectivity is represented as the edges. Their algorithm has the following steps:

1. Decompose the environment into cells. The width of the cells is twice the width of the robots' footprint. The length of the cells is the same as the height of the environment.
2. Assign a single cell to each robot as a starting point. The selection of the starting cells should result in the robots being evenly distributed through the environment.
3. Each robot will cover the assigned cell with a simple up and down motion as an atomic operation, update its internal representation of the adjacency graph, and broadcast the information to the other robots. See the adjacency graph at the top in Figure 5 after the robots completed the coverage of cells A, B, and C. The next step is to select the closest uncovered cell from the adjacency graph, inform the other robots of the selection, and move to the selected cell. If the selected cell is not adjacent, the robot will move to the cell via the shortest path. This step is repeated until all the cells are covered.

Each robot is responsible for detecting the presence of any obstacles in the cells they cover. One of three possible scenarios can occur during the cell coverage. First, the robot does not encounter any obstacles while covering the cell. Figure 5 shows that cell D is added to the right of cell A by the first robot. Second, part of the cell is blocked with obstacles. The robot covering cell B will then add cells E, F, and G to the graph. Third, a cell is divided by an obstacle. New cells, H and I, are added. Whenever a robot updates its local copy of the graph, it is shared with the other robots so it can be merged with their graphs (Rekleitis et al., 2008).

Multi-Robot Coverage Problem with Directional Constraints

Covering an environment using a specific pattern cannot be done with simple back-and-forth motions. The distributed coverage algorithm by Rekleitis et al. (2008) was modified to take directional constraints into consideration during path planning. Introducing the directional constraints will limit the travel direction of the robots during the covering operation and the result should be the desired global pattern. For the lawnmower problem, the required pattern should be visible when the robots are done mowing the lawn.

The first modification is the width used for the cells and how the environment is decomposed into cells. Rekleitis et al. (2008) use a cell width that is twice the width of the robots' footprint. The cell width used for the new algorithm is the same as the width of the robots. The number of cells assigned to each robot doubled as a result, but the area is the same. A cover direction is assigned to each cell as the environment is decomposed into cells. The pattern selected determines the shape of the cells and the directional constraints for those cells. Rekleitis et al. (2008) used a simple back-and-forth motion as a coverage pattern for each cell. To be able to compare the new algorithm to that of Rekleitis et al., the basic lawn-striping pattern was used. The selected pattern requires the covering direction for adjacent cells to be in the opposite direction. Figure 9 shows the cells created during the decomposition of the environment using the width as the robots' footprint. It also shows the cover direction assigned to each cell. A unique cell ID is assigned to each cell and used in the algorithm to address the individual cells.

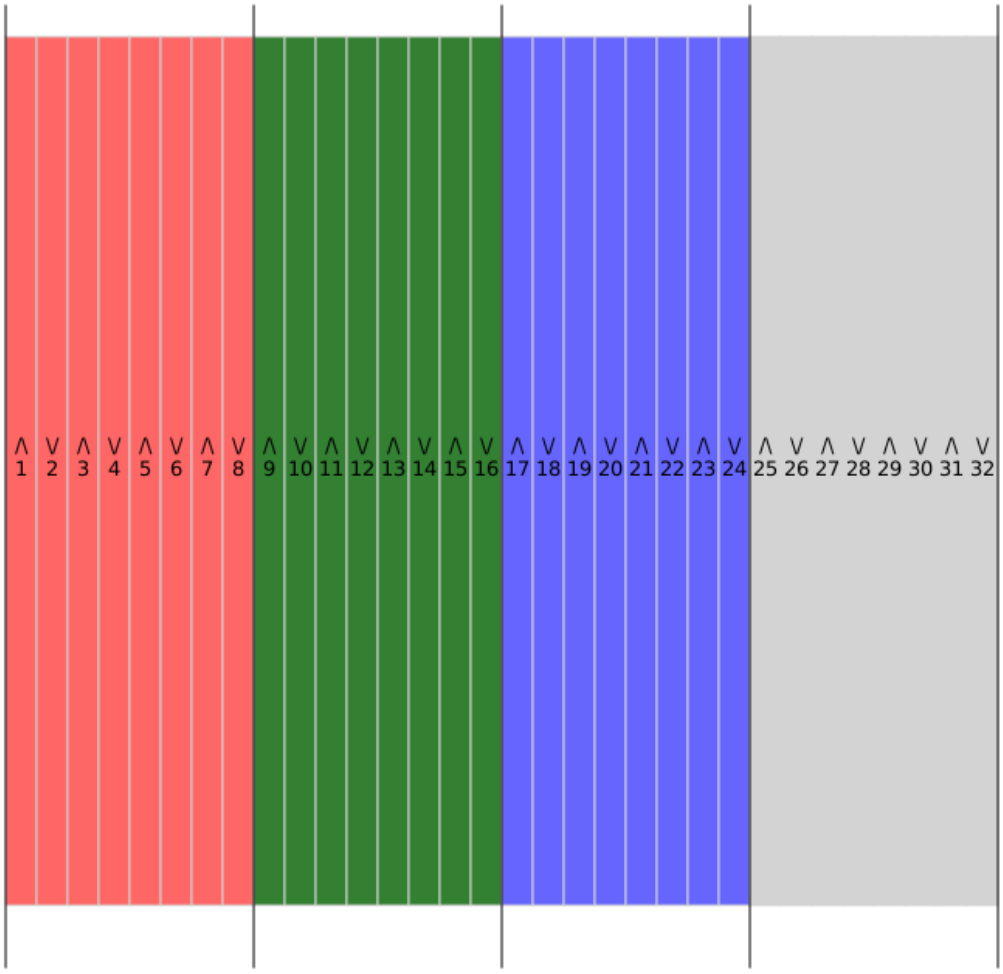

Figure 9. The decomposition of the environment into cells with an assigned direction.

The steps to decompose the environment into cells can be summarized as follows:

1. Divide the environment into cells that satisfy the selected pattern. The width of each cell is the same as the width of the robots' footprint or the end-effector. The length of each cell is determined by the boundaries of the environment.

2. Assign a cover direction to each cell based on the selected pattern. A simple stripe pattern is used for this research. This means that the cover direction

assigned to each cell alternates between North and South and that no neighboring cells will have the same cover direction.

3. Starting from the left, assign a number of cells to each robot. The number of cells to assign is calculated by dividing the number of cells created in step 1 by the number of robots. The number is changed to an even number to make it easier to compare against previous work. When the number of cell pairs is not divisible by the number of robots, some robots may have more cells than others.

After completing the steps required to decompose the environment and the assigning the cells to the robots, the robots are positioned at their starting points in the simulation software. Rekleitis et al. (2008) use a deployment vehicle to distribute the robots to their initial starting points. For the experiments in this research, the robots are moved to the starting point of their initial assigned cells instead of using a deployment vehicle. This position is calculated after each robot is assigned an area. The leftmost cell of the area assigned to each robot is used as the starting cell. The starting point is at the bottom since the directional constraint assigned to the initial cell is North.

Next, each robot will attempt to encircle the assigned area. The robots will travel on the perimeter of the assigned cells from the starting point, bottom left corner, and stopping short of the starting corner. The robot will actually stop on the last cell that is the closest to the starting corner and where cell direction matches the direction that the robot will travel next.

Not all the robots start at the same time with the exploration task in some of the experiments conducted by Rekleitis et al. (2008). This behavior was reproduced and a

delay mechanism was built so that a starting delay could be assigned to each robot. Figure 10 shows the different positions of the robots by delaying the start time of each robot. Each robot was started at a different time. As a result of the delay, the robots travelled different distances as shown in Figure 10. The green robot (second from the left) discovered the blocking obstacle.. Several tests were conducted to get delay values that represent the delays used by Rekleitis et al. Their paper includes images showing the position of the robots at different intervals. No start delay was assigned to the robot on the right, but the rest of the robots were assigned a different delay. The delay time got progressively bigger for the robots to the left.

By delaying the start time of the robots, certain events can be controlled. For example, Figure 10 shows that the blue robot (second from the right), is in a good position to win the auction for the cells that the green robot (second from the left) could not reach during the encircle operation due to the blocking obstacle.

Any obstacles detected during the encircle operation is encircled in an anti-clockwise direction. The encircle operation will stop when the robot is about to move out of the assigned area. Figure 10 shows the green robot (second from the left) not able to cover the entire length of the cells due to the blocking obstacle. It does not encircle the entire object since the object goes beyond the current assigned area. New cells are created on the North side of the detected obstacle and put up for auction. The robot will be the auctioneer for this type of auction. The robots will participate in any active auctions for cells that are not accessible by any given robot while busy encircling the initial assigned set of cells.

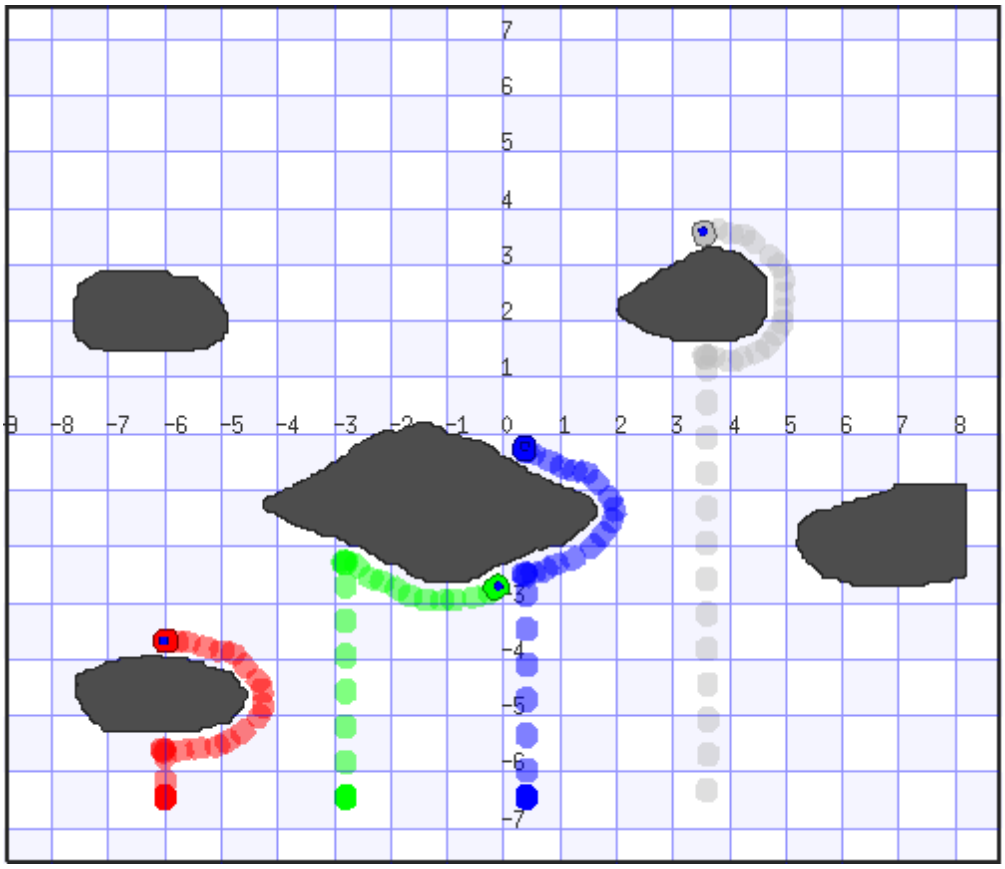

Figure 10. The green robot (second from the left) discovered the blocking obstacle.

The blue robot (second from right) won the auction since it is the closest to the cells. Manhattan distance between the robot and the closest point is used for the estimate cost. The cells from the auction are assigned to the blue robot which then moves to its new starting point (top right corner) of the area. Figure 11 shows the robot moving South to start encircling the cells won in the auction.

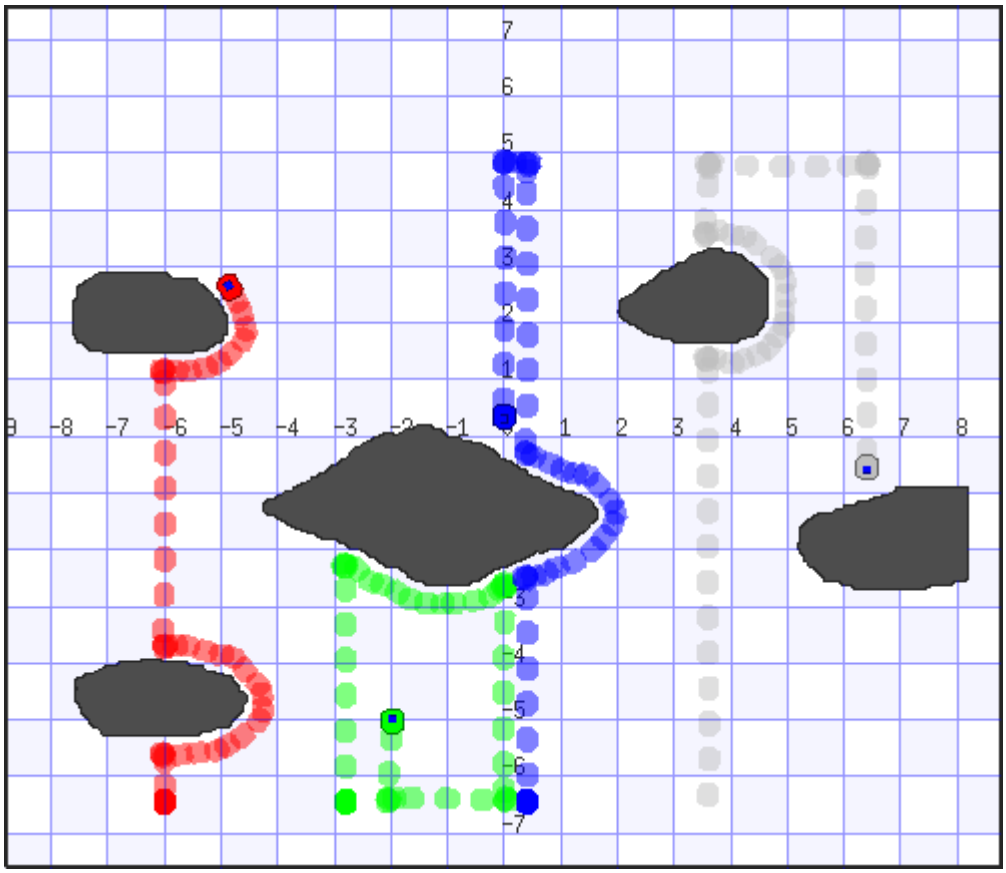

Figure 11. Blue robot (second from right) encircling the cells won from the auction.

Figure 12 shows the cells in the environment after the robots completed the encircle operation. In this case, all the obstacles were discovered during the encircle operation.

A robot can participate in auctions while busy with the encircling process. If a robot wins an area from another robot it bid on, the current area it is busy encircling will be put up for auction. Any robot that won an area during the encircling process will not be allowed to participate in similar types of auctions again.

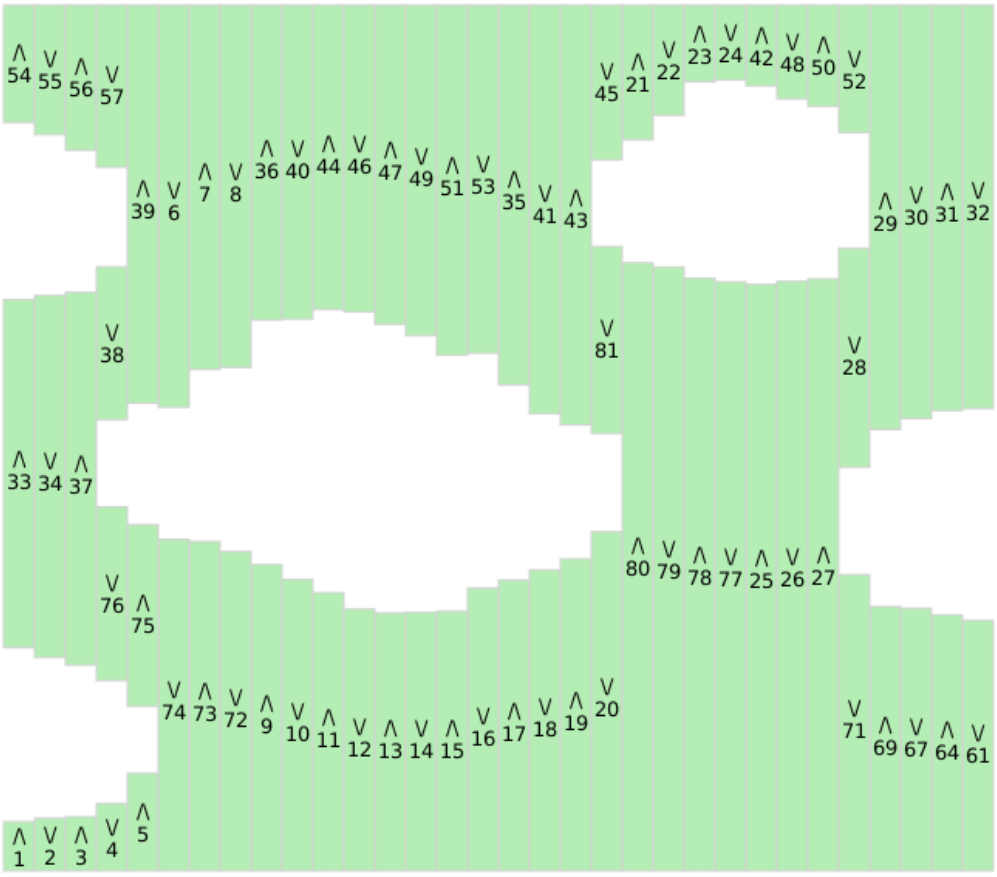

Figure 12. The cells in the environment after the robots completed the encircle operation.

The second modification to the algorithm by Rekleitis et al. (2008) is the movement of the robots in the environment. Instead of using the shortest path to travel to a point, the robots use the assigned direction to calculate the path and attempt to travel in the assigned direction when moving through the cells. The robots travel on the perimeter of the environment when the number of cells between the robot's current position and the cell that will be covered next exceeds a predefined threshold. If the directional constraint results in a robot getting stuck in a dead-end, the directional constraint is ignored until the robot is free to resume with the coverage task using the cell's assigned direction.

When a robot travels through a cell, it should use the assigned direction of the cell. If the robot travels in another direction when moving through a cell, it is considered a violation. During the path planning process, the distance that a robot is violating the directional constraint is doubled and added to the travel distance. The travel distance is used to determine the shortest path and to evaluate the performance of the algorithm. The violation distances are recorded separately for each robot.

After a robot is done encircling the assigned area, it will start to cover the uncovered cells. The following steps are used to cover the cells.

1. Get the next uncovered cell that is the closest and that has the shortest direction violation distance to get to the start position.
2. Cover the cell in the assigned direction.
3. Repeat these two steps until no more uncovered cells are available.

There are a few steps involved in getting the next uncovered cell. The first step is to look at the neighboring cells. If there is a neighboring cell that is uncovered, it will be used. If both the neighboring cells are uncovered, the cell that is closest to the starting corner of the robot will be used. If there are no uncovered neighboring cells to the current cell the robot is in, all the cells assigned to the robot will be evaluated to get the uncovered cell that is the closest in terms of travel distance. The pseudocode for getting the next cell to cover is listed in Figure 13.

```
1  function getNearestUncoveredCell (robotXPosition, robotYPosition) returns
nearest uncovered cell
2     nearestCell ← null
3     shorestDistance ← -1.0
4     for each cell in cells do
5        if cell.isCovered = false then
```

```
6        toYPosition ← (cell.coverDirection = NORTH) ? cell.startYPosition :
cell.endYPosition
7        distance  ← getManhattanDistance(robotXPosition, robotYPosition,
cell.getStartXPosition, toYPosition)
8        if shortestDistance < 0.0 or distance < shortestDistance then
9          shortestDistance ← distance
10         nearestCell ← cell
11   return nearestCell
```

Figure 13. Pseudocode for getting nearest uncovered cell.

After the nearest uncovered cell has been identified, all possible paths from the robot's current position to the uncovered cell are built. This pseudocode for this logic is listed in Figure 14.

```
1  function getAvailablePaths(currentCellID, destinationCellID) returns a set of
available paths between cells
2    connectingCells ← buildConnectingCells(currentCellID, destinationCellID)
3    paths ← addInitialNeighboringCells(currentCellID, connectingCells)
4    repeat
5      pathsChanged? ← false
6      for each path in paths do
7        lastCellID ← get last cell ID in paths
8        if lastCellID != destinationID then
9          neighboringCells ← getNeighboringCells(lastCellID)
10         for each cell in neighboringCells do
11           if cell is not in path then
12             //add a path record (clone path record and append
13             newPath ← clone path
14             append cell to end of newPath
15             add newPath to paths
16             pathChanged? ← true
17             purgeMarkedPaths(path)
18   until pathsChanged? = false
19   return paths
```

Figure 14. Pseudocode for getting available paths between two cells.

For example, when the robot is done covering cell number 10 and located at the bottom of the cell, the next uncovered cell that is closest to the robot's current location is

cell number 13. The reason why cell number 12 was not picked is due to the distance that the robot has to travel to get to the top end of the cell. The direction assigned to the cell requires covering to start from the top end.

The following cells are in the path: 11, 12, and 13. Cell number 10 and 11 are already covered, cell 12 should be covered in a Southern direction, and cell 13 should be covered in a Northern direction. Figure 15 shows one of the paths and the cells that is involved to get from cell 10 to cell 13.

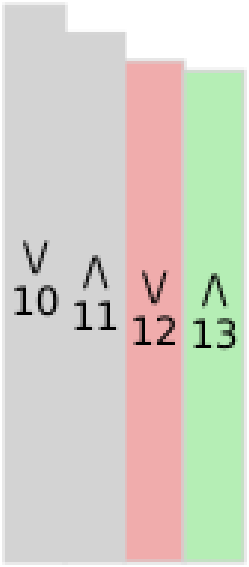

Figure 15. Cells for path 10:[11,12,13].

A route is then planned for each path. A route consists out of a set of waypoints that will be used by the robots for point-to-point movement. A violation distance is calculated when the robot does not travel through a cell or through part of a cell in the direction assigned to a cell. This distance is added as a penalty to the overall distance for the route. The shortest route is selected to move the robot from its current position to the start of the uncovered cell. The logic is listed in Figure 16.

```
1  function getPathWaypoints(path, robotXPosition, robotYPosition) returns a set
of waypoints
2  shortestRoute ← empty set of waypoints
3  shortestDistance ← -1.0
4  pathLegs ← getPathLegs(path, robotXPosition, robotYPosition)
5  if SIZE(path) > 8 then
6    bitString ← an empty string
7    if cover direction of last cell in path = NORTH then
8      // use perimeter on South to move to last cell
9      bitString ← APPEND('0', SIZE(path))
10   else
11     // use perimeter on North to move to last cell
       bitString ← APPEND('1', SIZE(path))
12   shortestRoute ← getBitStringWaypoints(path, pathLegs, bitString,
robotXPosition, robotYPosition)
13   if shortestRoute = empty then
14     numberOfOptions ← 2^numberOfCells
15     for optionNumber ← 0 to numberOfOptions-1 do
16       bitString ← binaryString(optionNumber)
17       waypoints ← getBitStringWaypoints(path, pathLegs, bitString,
robotXPosition, robotYPosition)
18       waypoints ← optimizeRoute(waypoints)
19       violationDistance ← waypoints.violationDistance(robotXPosition,
robotYPosition) * 2.0
20       if shortestRoute = empty then
21         shortestRoute ← waypoints
22         shortestDistance ← shortestDistance + violationDistance
21       else
22         if waypoint.distance + violationDistnace < shortestDistance then
23           shortestRoute ← waypoints
24           shortestDistance ← pathWaypoints.distance + violationDistance
25 return shortestRoute
```

Figure 16. Pseudocode for getting waypoints for a path.

Table 1. Waypoints to move the robot from cell 10 to cell 13. contains an example of waypoints calculated to move the robot from its current position, cell 10, to cell 13.

Table 1. Waypoints to move the robot from cell 10 to cell 13.

Cell Number	Waypoint (x, y, direction)
11	(-1.600, -6.400, 0.000)
12	(-1.200, -6.400, 0.000)
13	(-0.800, -6.400, 0.000)

Figure 17 lists the pseudocode for the getBitStringWaypoints function. This function builds a list of waypoints based on the values of the path and bitString variables passed to the function as parameters. The value of a bit in bitString is used to determine if the North or South point of the corresponding cell in the path will be used as a waypoint. The pathLegs variable contains a list of waypoints that is used to move from the start, end, or inside position of a cell to either the start or end position of another cell. The values are built only one time for each path to help improve performance.

```
1  function getBitStringWaypoints (path, pathLegs, bitString, robotXPosition,
robotYPosition) returns a set of waypoints
2  if cover direction of last bit in bitString != cover direction of last cell in path
then
3     return null
4  waypoints ← empty set of waypoints
5  xPosition ← robotXPosition
6  yPosition ← robotYPosition
7  for each path in paths do
8     if FIRST(path) then
9        pathLegType ← bitString.charAt(0) = '1' ? ROBOT_TO_END :
ROBOT_TO_START
10       pathWaypoints ← pathLegs.getPathLegWaypoints(fromCellID,
pathLegType)
11    else
12       fromStartOrEnd  ← bitString.charAt(cellNumberIndex-1) = '1' ?
END_Y_POSITION : START_Y_POSITION
13       toStartOrEnd  ← bitString.charAt(cellNumberIndex) = '1' ?
END_Y_POSITION : START_Y_POSITION
14       pathLegType ← PathLegType.get(fromStartOrEnd, toStartOrEnd)
15       pathWaypoints ← pathLegs.getPathLegWaypoints(fromCellID,
pathLegType)
16 return waypoints
```

Figure 17. Pseudocode for getting waypoints for a path.

After a robot has covered the initial assigned area, it will start to offer its service to cover cells assigned to other robots. This is done via the auctioning system. Individual

cells are assigned to the robot with the lowest calculated cost. That is, the lowest cost is the shortest distance between the robot's current position and the starting point of the cell to cover the robot in the assigned direction. The cell that is the closest to the robot's starting corner is preferred. In general, this allows for the shortest travel distance to get to the uncovered cell. Without this requirement, it is possible for the robot to skip the cell that is the closest to the starting corner. To cover this skipped cell may result in extra travel distance just to get back to the uncovered cell at some point.

The algorithm by Rekleitis et al. (2008) guarantees complete coverage by using Boustrophedon decomposition. The new algorithm also guarantees complete coverage, but it is doing so by planning a path that covers all the accessible area of the environment. That is, the new algorithm guarantees complete coverage when all the cells identified during the decomposition of the environment except parts of the cells that are blocked by obstacles have been covered.

Portions of cells may not be accessible if obstacles restrict robots from moving to or through a cell. A cell is broken into smaller cells when an obstacle is detected while moving in a cell. The portion of the cell that the robot travelled over retains the original cell ID and is marked as covered. The portion of the cell that is blocked by the object is used to create a new cell. A new cell ID is assigned to the new cell and it will remain as uncovered. The new cell will be put up for auction to be covered later.

Data Analysis

The results produced by the algorithm were compared against results from similar experiments conducted by Rekleitis et al. (2008). The performance of the algorithm should meet the following criteria:

1. All of the accessible area should be covered. That is, the union of all the cells represents the free space and all the cells have been visited by at least one robot.
2. The cells should be covered in the same direction as assigned to each cell.
3. The time to complete the task should be minimized.

An efficient coverage algorithm should avoid repeat coverage. To verify the efficiency of the algorithm, the distance traveled by the robots, the time it took to complete the coverage task, and the distance that the directional constraint was violated (robot did not travel in the assigned direction while moving through a cell or portion of a cell) are captured.

Resources

The following resources were used to complete the experiments and research tasks:

1. Computer software: The Player/Stage robotic simulation package was used to conduct the experiments for this research. The Javaclient2 package (version 2) was used as the interface between the Java code and the simulation software. Eclipse that is bundled with Spring Tool Suite 3.8.1 was used to develop the code for the algorithm. Open JDK 64-bit version 1.8.0_181 was used to compile the Java source code and the Java Virtual Machine was used to run the Java bytecode. Ubuntu 16.04.1 LTS was loaded as the operating system on the computer used for the development work and running the simulation software. The graphics package paint.net 4.0.8 was used to clean up the original images with the obstacles and Inkscape 0.91 was use to recreate the obstacles. Microsoft Excel was used to store and analyze the data. Microsoft Word was used to document the findings.

2. Computer hardware: A standard desktop computer (Intel® Core™ i5-3470T CPU @ 2.90 GHz x 4 with 4GB of RAM and 60GB SSD) was used to develop the algorithm and run the simulation software. A Windows-based laptop was also used to run the Microsoft software.

Summary

A novel algorithm for the complete coverage problem with directional constraints on cell coverage was introduced and described in this chapter. The steps involved to break down the environment into cells and how the path planning works with the directional constraints were also described.

Chapter 4

Results

Introduction

This chapter presents the results and findings from the different experiments that were conducted using an implementation of the new complete coverage algorithm with directional constraints. The new algorithm was tested in a variety of environments with different number of robots. The results from these experiments are presented in the following section.

Baseline Results

Rekleitis et al. (2008) tested their distributed coverage algorithm in a variety of environments with two to five robots. They conducted several experiments using the same environment. Table 2 shows the results from experiments conducted by Rekleitis et al. The same environment is used, but with a different number of robots for the distributed coverage algorithm.

Table 2. Results from experiments conducted by Rekleitis et al. (2008).

Number of Robots	Time (minutes)
3	30.0
4	25.8
5	23.3

They note that the distances travelled by the different robots are approximately the same. This indicates that the workload is evenly distributed between the robots. Several

assumptions were made for the implementation of the distributed coverage algorithm by Rekleitis et al. (2008). Estimated values were used where the values were not given, for example, the dimensions of the environment, the size and location of the obstacle, and the performance of the robots. For these experiments, the width and height selected for the environment are 17.6 m and 15.2 m respectively.

Table 3 shows the results from the experiments conducted with the reconstructed obstacle image and different number of robots. The algorithm does not take directional constraints into consideration for path planning. The travel distances are not reported by Rekleitis et al. (2008) for the experiments with the distributed coverage algorithm.

Table 3. Results from experiments with no directional constraints to establish the baseline.

Number of Robots	Time (minutes)	Travel Distance (meters)
3	21.0	141.8
	19.9	126.6
	19.2	146.6
4	15.2	95.4
	15.6	79.4
	16.3	87.3
	15.0	106.9
5	14.2	74.6
	14.6	86.4
	15.6	65.3
	7.0	33.4
	13.8	90.4

The coverage task is only considered done when the last robot completes the last uncovered cell. The size of the environment, maximum forward speed, acceleration speed, deceleration speed, turn speed, and start delays have huge impacts on the time it takes the robots to complete the coverage operation. The majority of these values are

estimated. Not using the same values as Rekleitis et al (2008) may explain the variation between the results noted by Rekleitis et al. and the experiments for this work.

Several screenshots were captured during the experiments. When the "Save Screenshots" option in Stage is checked, screenshots will be saved to disk at regular intervals. The configuration that was used for the simulator resulted in one screen every 100 ms. These screenshots can be used as a visual aid to inspect the coverage operation as well as a visual tool to verify that no accessible area is not covered. The screenshots of the environment for the first three experiments are displayed in Figure 18, Figure 19, and Figure 20. There were no cells or portions of cells that were not covered except where the obstacles are located. The screenshots are a visual confirmation that no cells were left uncovered.

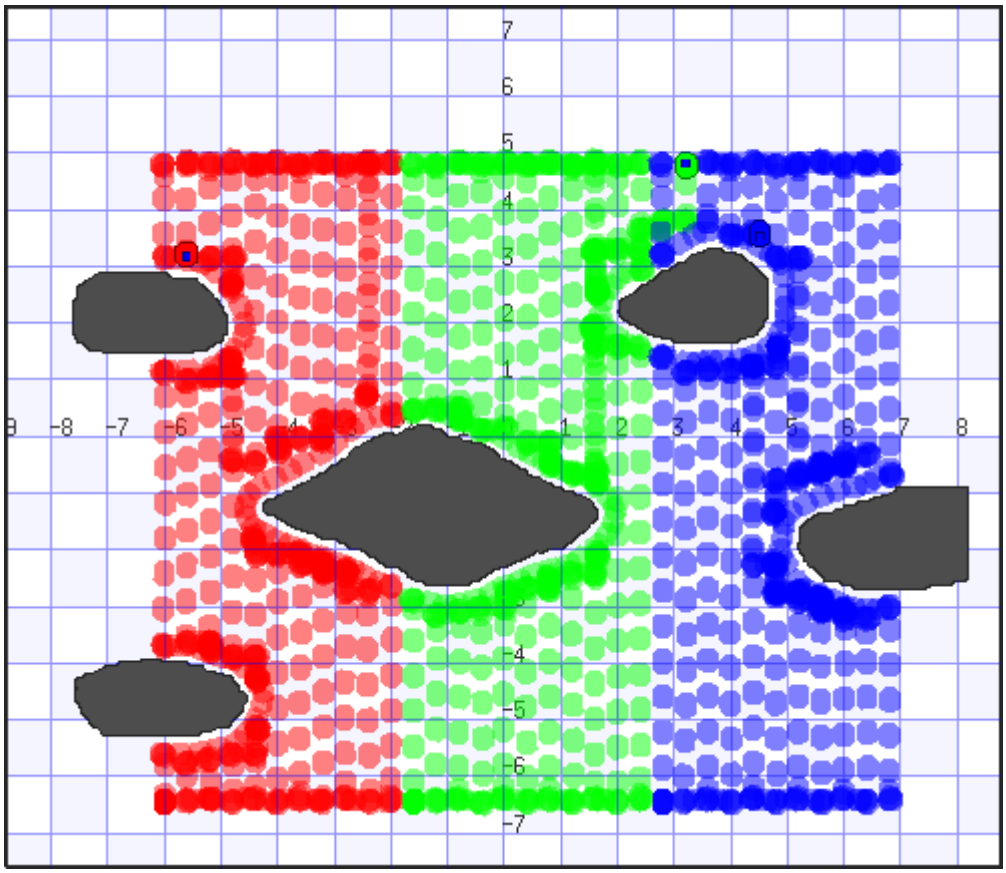

Figure 18. Complete coverage using three robots and no directional constraints.

Encircling an obstacle takes longer than covering a cell. The robots' differential steering and behavior to stay at a predefined distance away from the obstacles make the robots move slower. The robots also slow down when an obstacle is detected. This is done in order to reduce the chance that the momentum will push a robot into the obstacles or overshoot a stopping point. The middle (green) robot in Figure 18 covered more distance than the other two robots since it only had one obstacle in the area it covered. The robot on the left (red) has three obstacles in the initial area assigned to it, and as a result, travelled the shortest distance.

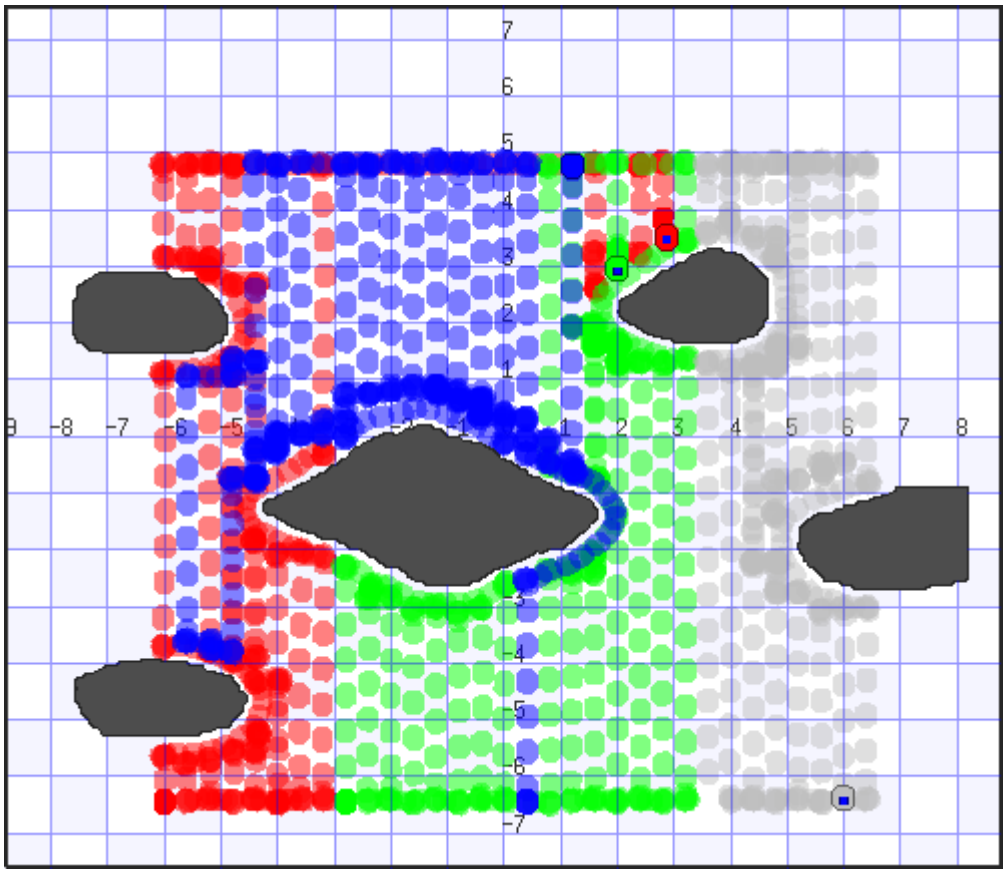

Figure 19. Complete coverage using four robots and no directional constraints.

The length of the starting delay has a huge impact on which robot will cover a specific cell. At the beginning of an experiment, it is easy to predict which cell will be covered by a robot, but it becomes impossible to predict after a few auctions. This was also noted be Rekleitis et al. (2008).

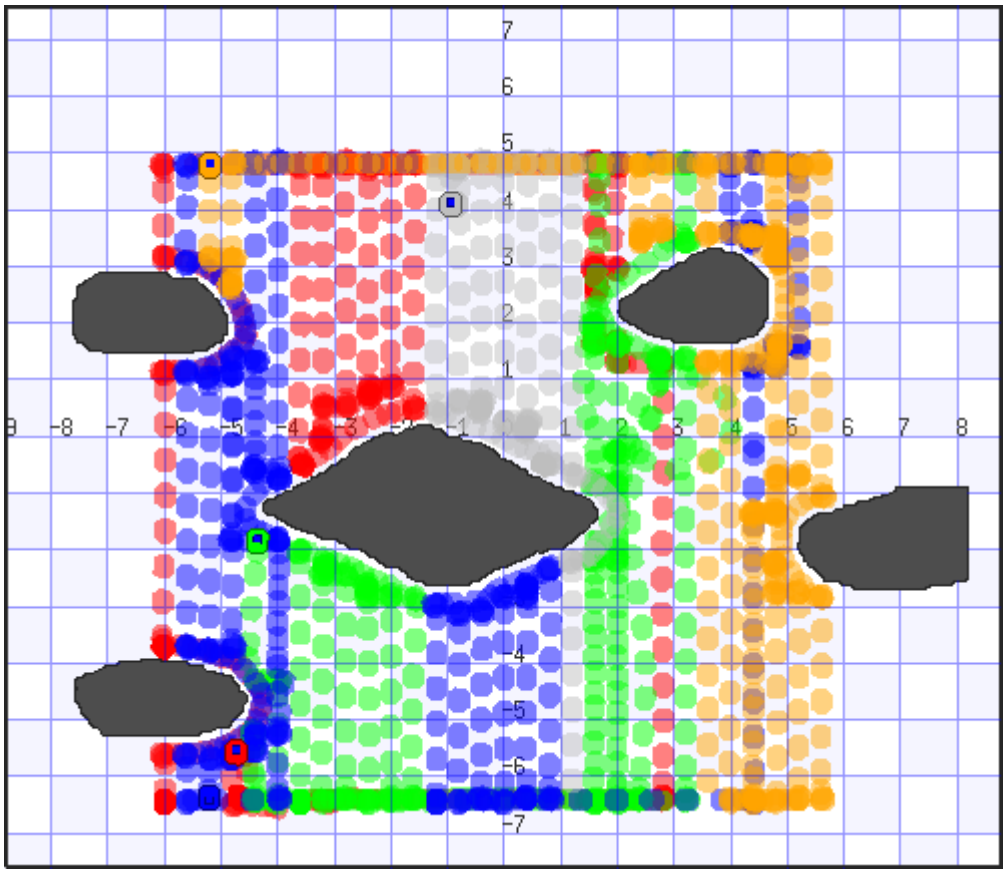

Figure 20. Complete coverage using five robots and no directional constraints.

Experiments with Proposed Algorithm

This section reports the results from the experiments that evaluate the new algorithm taking directional constraints into account for path planning. The same environment that was used to get the baseline results, was used to get results for the new algorithm. The number of robots ranged from three to five. The steps are shown in Figure 21 through Figure 33. The results from the experiments with the new algorithm using a different number of robots are presented in Table 4.

Table 4. Results from experiments with the new algorithm.

Number of Robots	Time (minutes)	Travel Distance (meters)	Constraint Violation Distance (meters)
3	21.7	156.9	0.0
	16.9	143.8	0.0
	17.0	125.3	1.7
4	10.5	73.5	3.2
	17.4	99.4	2.3
	12.1	74.3	0.0
	18.1	124.5	1.8
5	9.7	70.0	0.0
	10.6	81.0	14.2
	10.7	58.1	8.4
	11.7	73.8	2.3
	10.1	75.5	4.1

After the environment is decomposed into cells and directions assigned to the cells, the cells are assigned to the robots in pairs of two. The cell on the left of the assigned area for each robot is used as the starting cell. The bottom is used as the starting point for each robot since the cover direction of the starting cell is north. The robots are then moved to their starting positions. Figure 21 shows the robots at their starting positions.

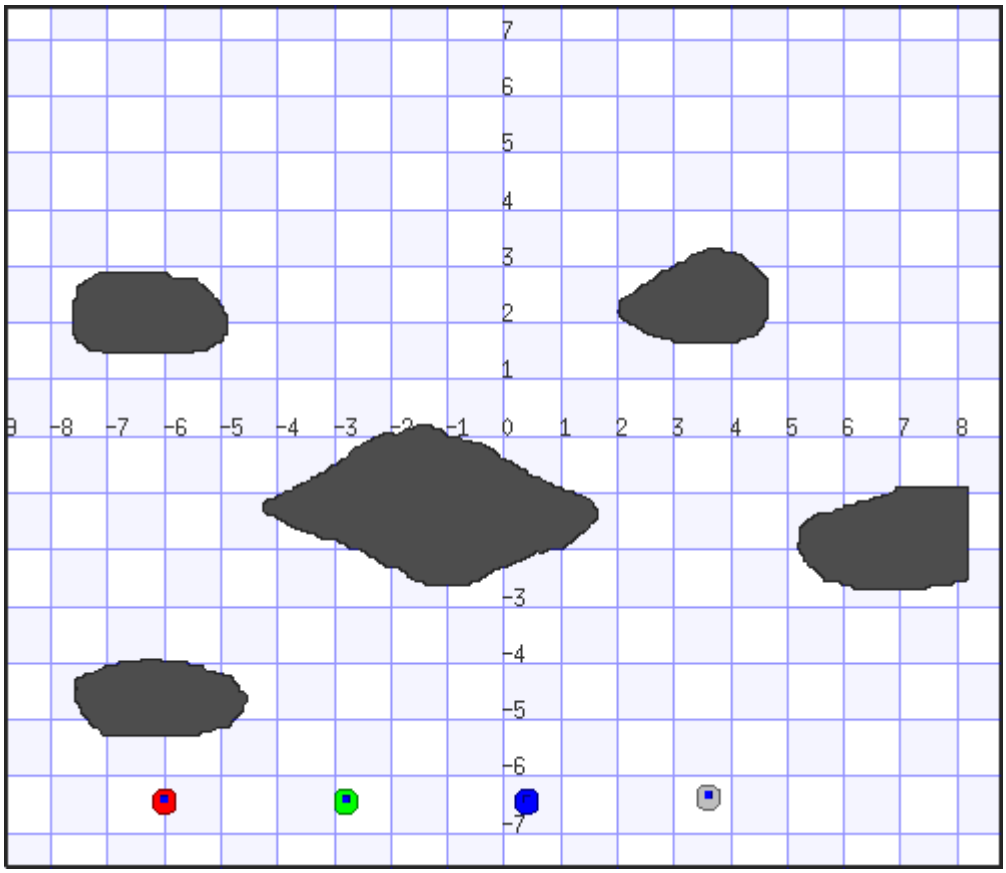

Figure 21. The robots at their starting points.

After the robots are positioned at their starting point, they wait until the predefined delay assigned to each individual robot is over. The delays were selected based on the position of the robots shown in the work of Rekleitis et al. (2008). When the delay assigned to a robot is over, the robot can start exploring the area assigned to it. Figure 22 shows the robot on the right busy with the exploration phase.

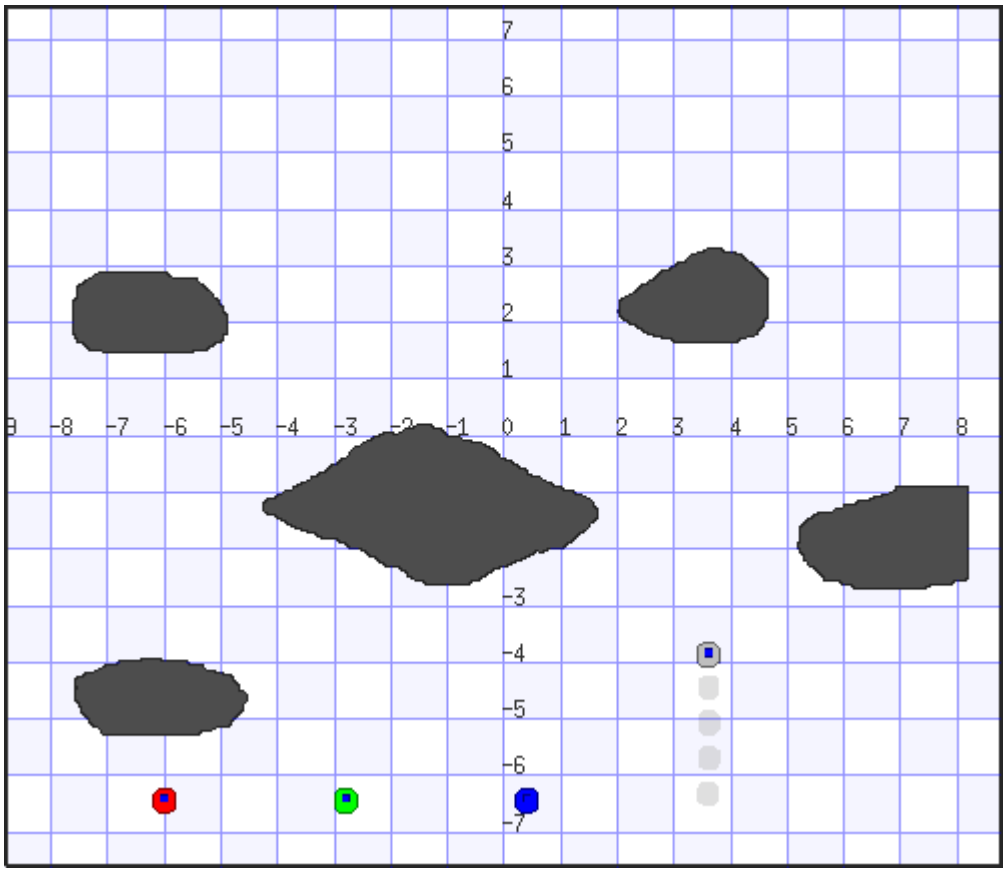

Figure 22. The robots start exploring after a predefined delay.

The location and size of the obstacles are not known beforehand. The point where the robot detects an obstacle is known as the critical point. Figure 23 shows two of the robots detecting obstacles in the cells they are moving into. The current cell is split into two at the critical point. The cell that the robot is currently on will retain the original cell ID, but it will be marked as covered. The new cell will get a new cell ID. The robot will then start encircling the obstacle.

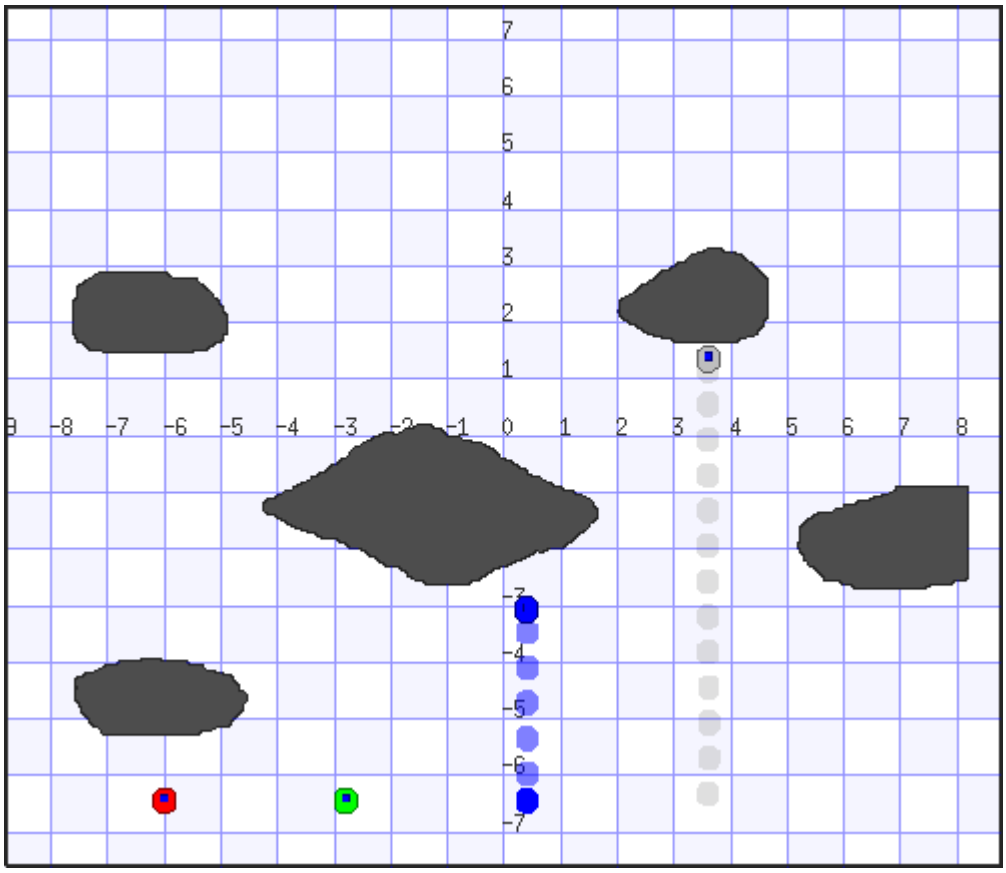

Figure 23. Robots detecting obstacles while busy exploring.

Figure 24 shows that all four robots are busy encircling the obstacles that they detected. New cells are created during this operation and will be put up for auction.

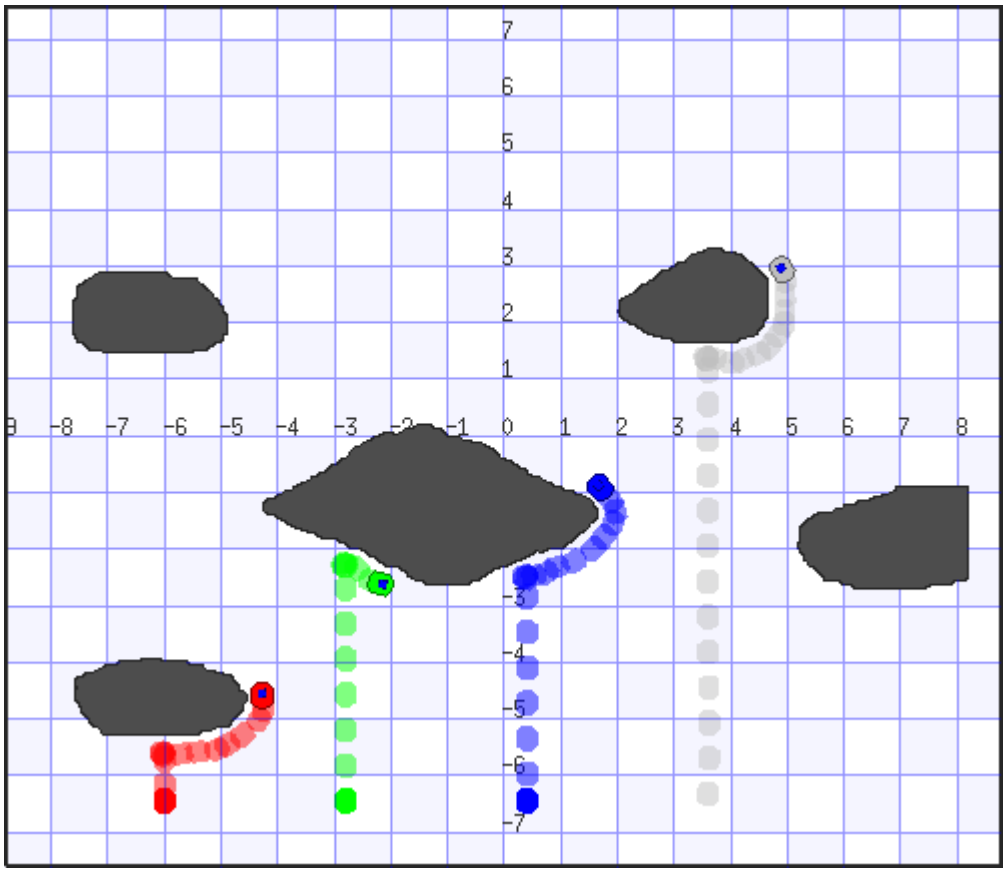

Figure 24. All four robots busy encircling the obstacles detected in their path.

A robot will travel only in the assigned cells when busy encircling an object. Figure 25 shows that Robot 2, the second robot from the left (green), cannot encircle the entire object since it is outside of the assigned area. It will proceed to explore the perimeter of the area. In this case, it is busy turning south towards the bottom right corner that was assigned to it. At this point the robot will start auctioning the new unreachable cells that it is not able to access. Any robot that is still busy exploring its starting cell can bid on this auction. The robot that is the closest to any of the new cells will most likely win the auction.

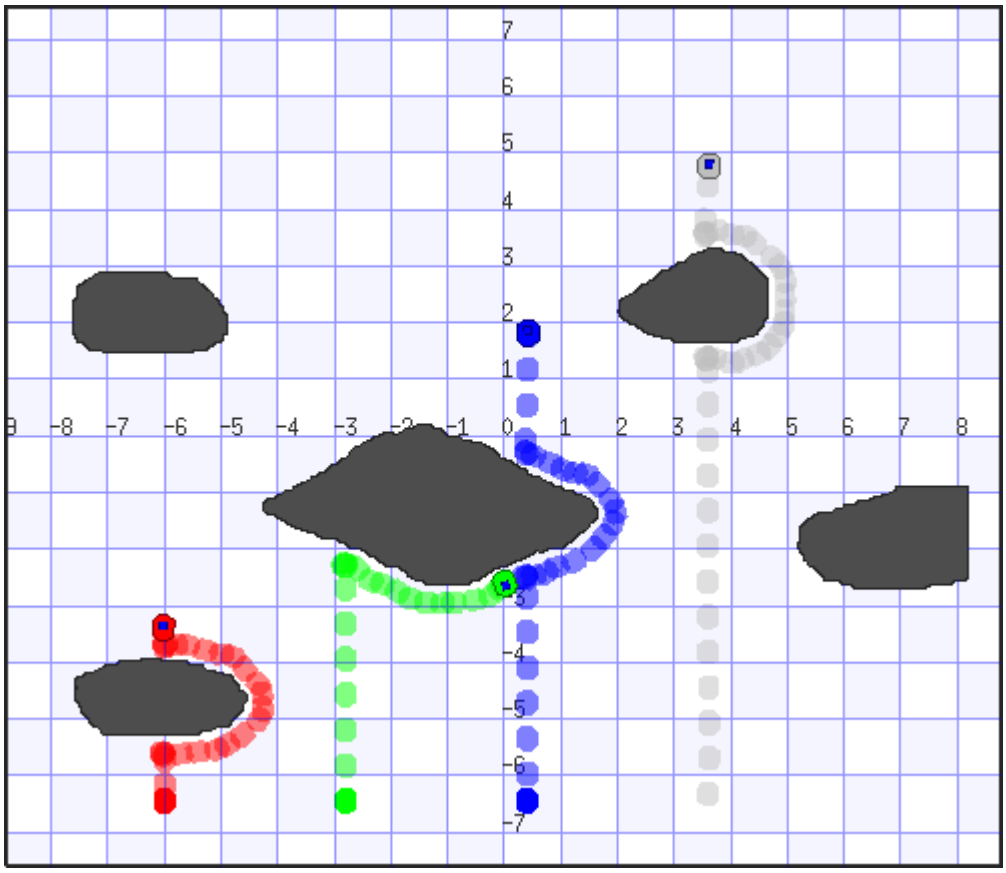

Figure 25. Robot 2, the second robot from the left (green), is not able to encircle the obstacle.

Robot 3, the second robot from the right (blue), won the new cells auctioned by Robot 2. It will move to the end of the current cell and then start exploring its new area as shown in Figure 26. The cells that were initially assigned to the robot will be put up for auction. If there is no winner in the auction, the robot will return to these cells after it is done covering the new cells.

Figure 26 also shows that Robot 2 is done exploring its cells. It will stop short of the starting corner of the area. The cell that is selected for the covering process has the

same direction assigned to it that the robot needs to move in next. In this case the directional constraint is North.

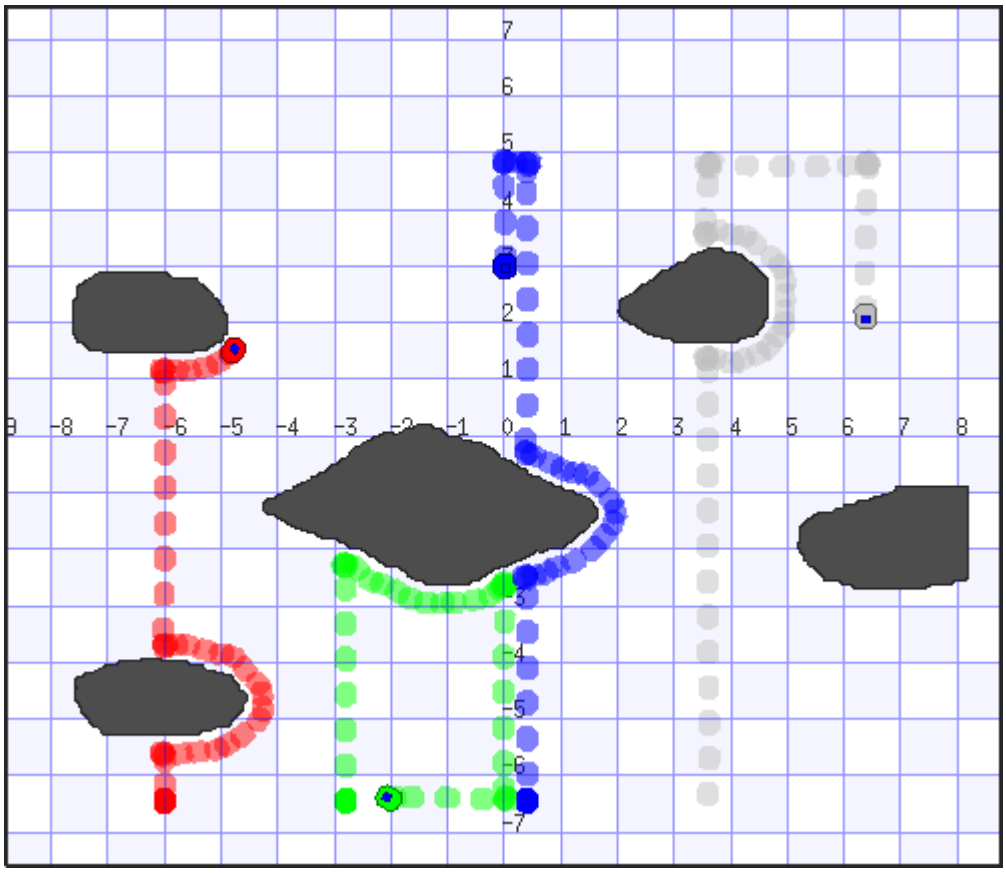

Figure 26. Robot 3, the second robot from the right (blue), busy exploring its new area.

After the first cell has been covered by Robot 2, it will proceed to cover the nearest uncovered cell. The robot will select the cell on the left to cover next as shown in Figure 27. The cell on the right is actually closer, but skipping the cell on the left has undesired consequences. When a cell like this is skipped and the cell on the right is covered, the next uncovered cell that is the closest to the robot after the cover operation will be the one on its right. This will leave the cell on the left uncovered, until all the other cells

have been covered. Extra distance is required to return to cover the cell later. The algorithm was modified to try to cover the cells closest to the starting corner first as an attempt to reduce the extra distance required to return to skipped cells later. The average extra distance is about the width of the area in most cases. The algorithm was modified to not skip the cells closest to the starting corner based on the findings during the implementation phase.

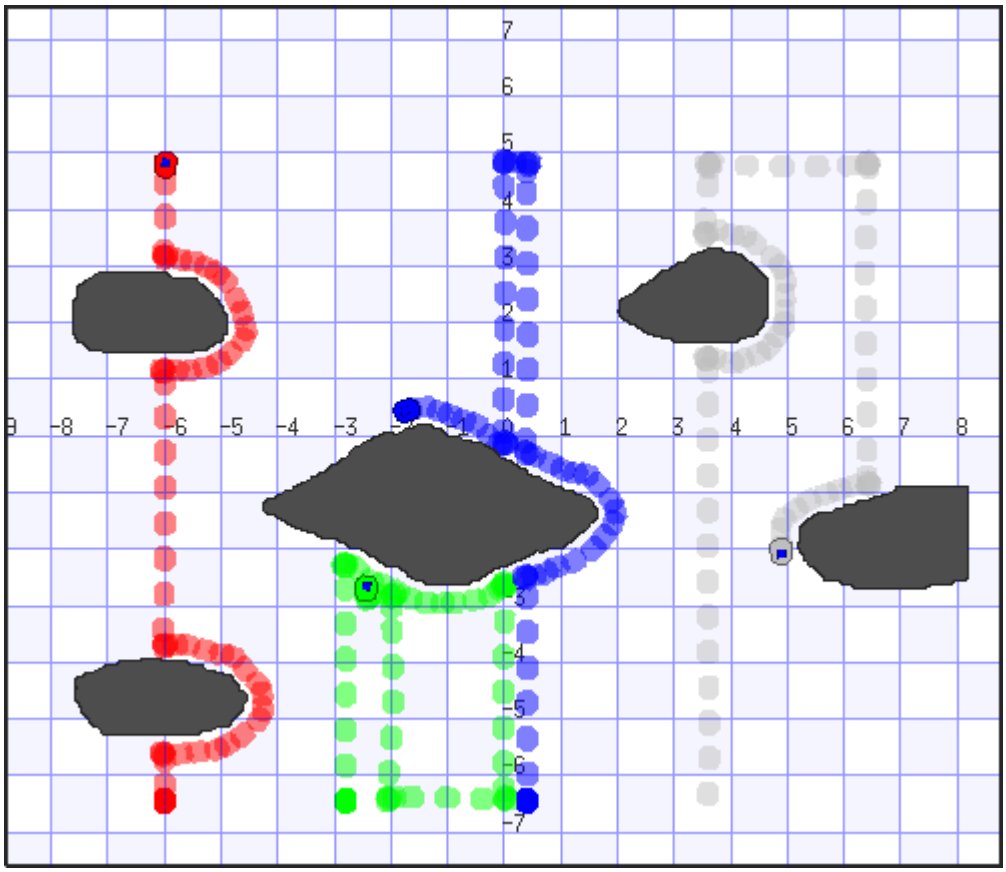

Figure 27. Robot 2 (green) is busy covering its second cell.

Figure 28 shows Robot 2 going North while covering the cell and Robot 4 going South. The path planning part of the new algorithm restricts the movement of the robots based on the direction assigned to the cells they are moving in.

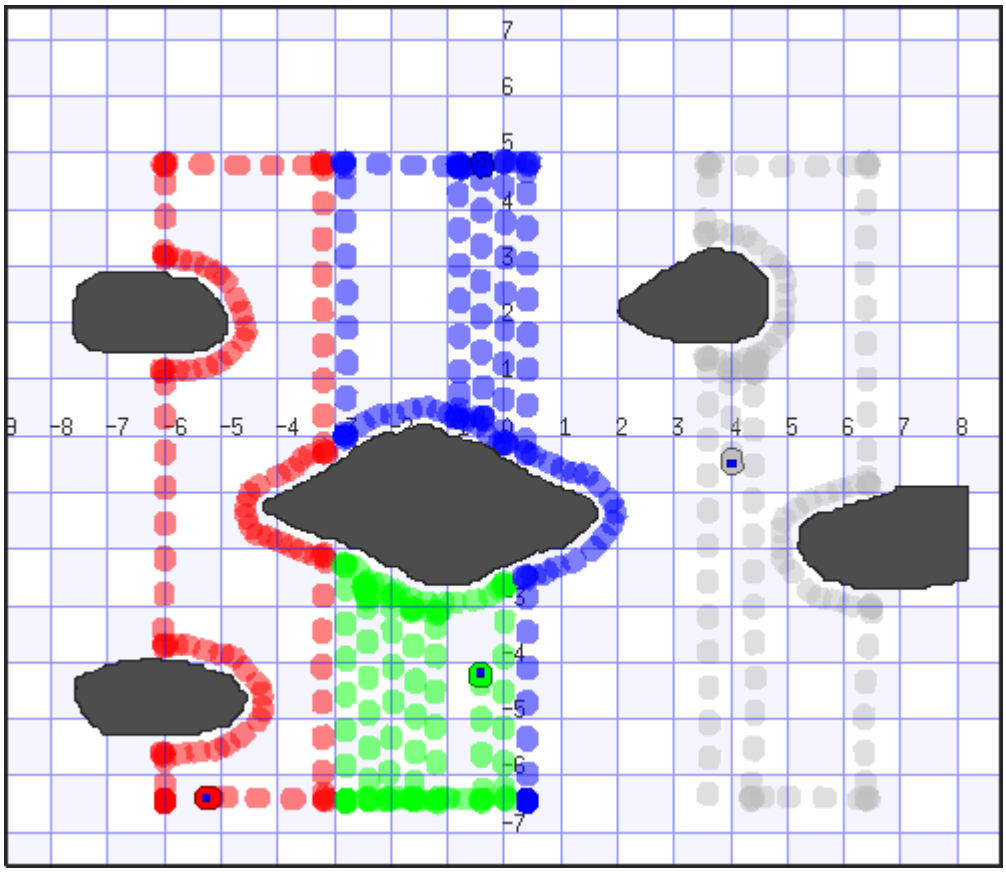

Figure 28. Robots moving in the same direction than what is assigned to the cells.

When a robot covered the initial area, either the initial area assigned to it or the cells won in an auction during the exploring phase, it will start offering its service to cover cells assigned to other robots. Figure 29 shows Robot 2 done covering its initial area. At this point it will create a service auction to offer up its service.

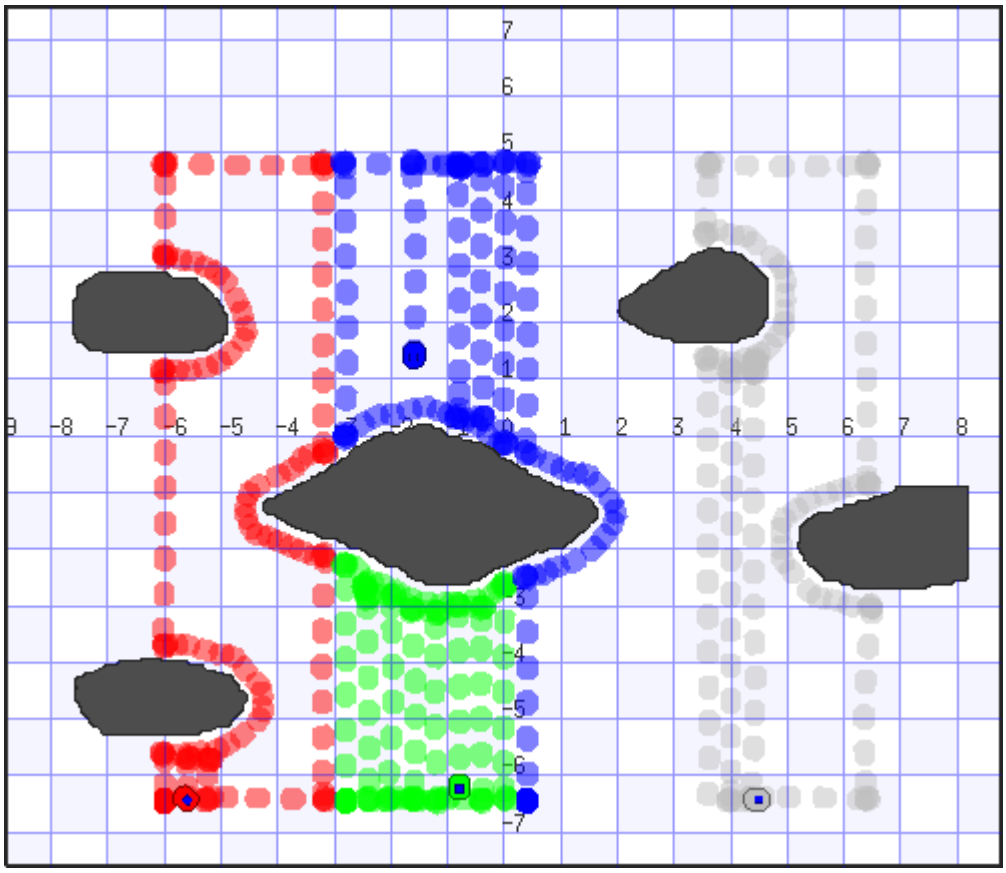

Figure 29. Robot 2 (green) is done covering the initial assigned area.

Robot 2 created a service auction and won cells from Robot 3. Robot 3 did not finish exploring the cells before it won cells from Robot 2 earlier. Since Robot 3 is still busy covering those cells, it did not consider these cells during the covering phase. Figure 30 shows Robot 2 busy exploring the cells won in the auction by traveling on the perimeter of the area.

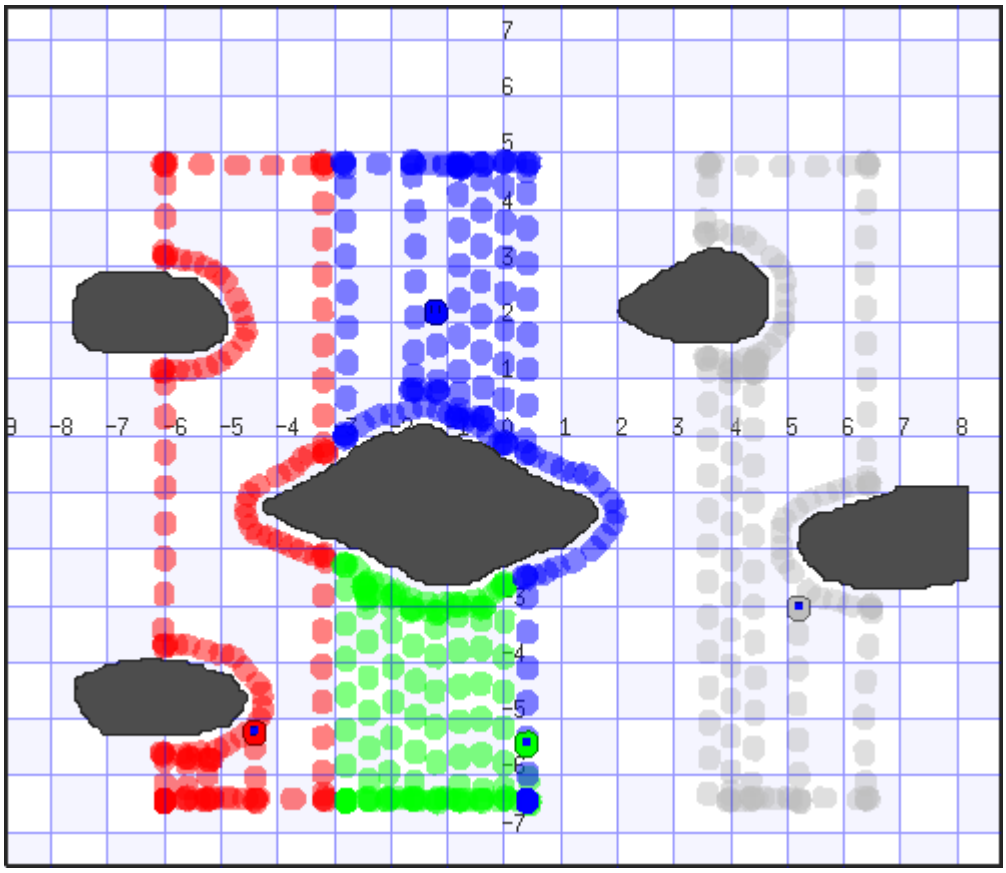

Figure 30. Robot 2 (green) won cells from Robot 3 (blue) with a service auction.

After Robot 3 has covered the area it won in an earlier auction, it created a service auction since Robot 2 won the cells initially assigned to it. Figure 31 shows Robot 3 busy covering a cell it won in a service auction from Robot 1 (red). Each robot that has uncovered cells excluding the cell it is covering currently is considered for the auction. The cell that is the closest to the robot that starts the service auction will be won by the robot. The distance to the starting point of the cells is used and the starting point (North or South) end of the cell is determined by the direction assigned to the cell.

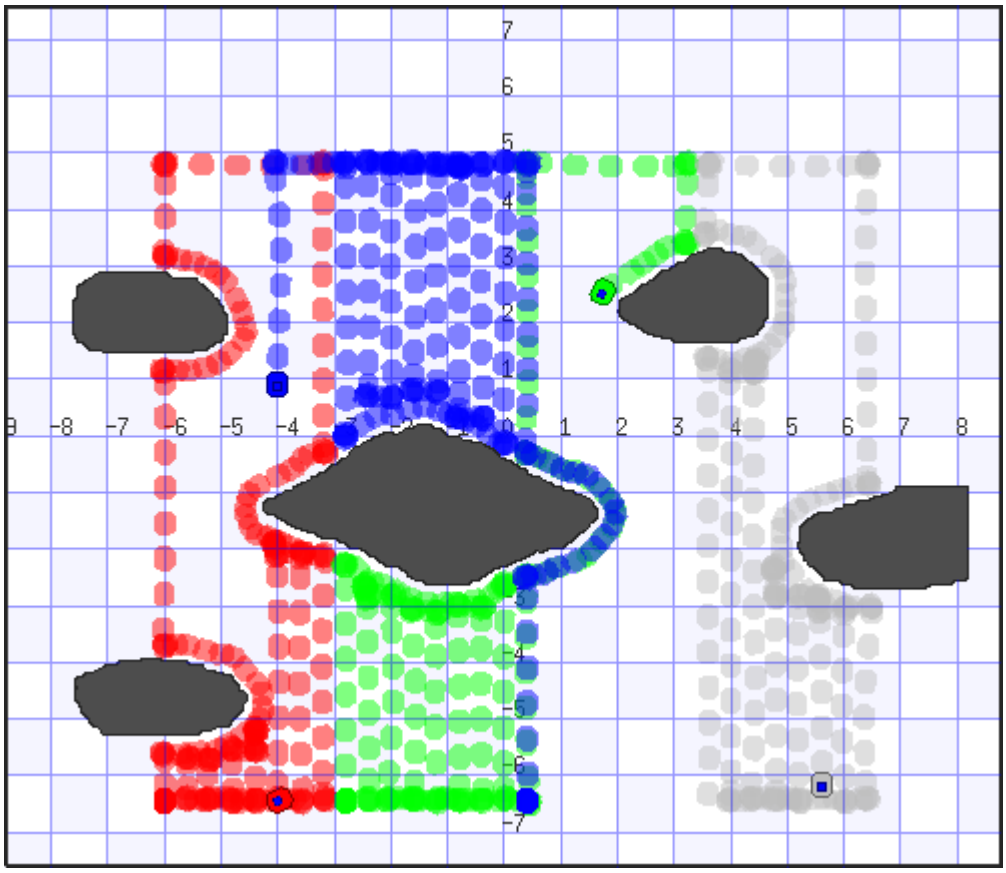

Figure 31. Robot 3 (blue) busy covering a cell won in a service auction.

Figure 32 shows Robot 2 busy covering the cells that it won in a previous auction after travelling around the perimeter as part of the exploring phase. Robot 2 has repeated the coverage of the left perimeter after Robot 3 already explored it. The algorithm was designed to have this behavior to make sure it can accurately detect and encircle any obstacles that might be located in the second cell from the starting corner. This was decided after obstacles were not encircled properly when they start in the second cell.

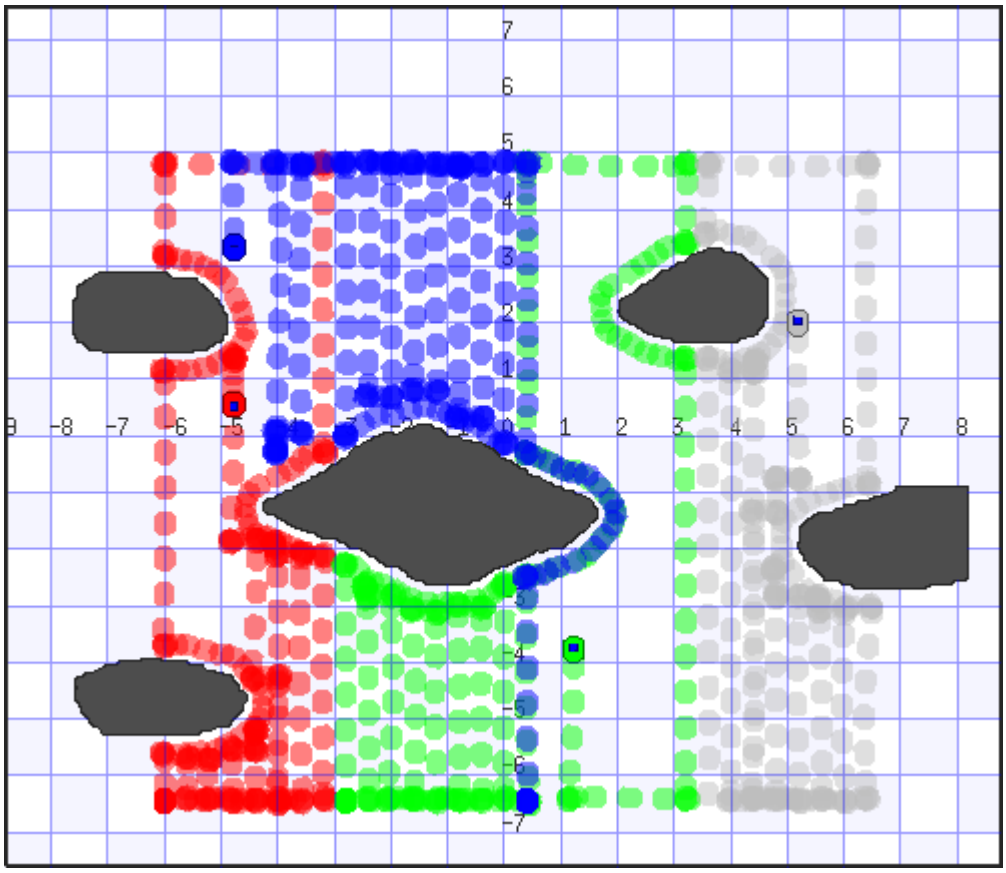

Figure 32. Robot 2 (green) starts covering the cells it won in a previous auction.

The algorithm stops running after all the cells have been covered. Figure 33 shows the environment after the coverage operation is complete as well as the location of the robots where they stopped. The cells are colored when the robots travel through them. Most of the cells have only one color and it shows which robot covered the cell. Cells with more than one color may have more than one robot that can claim it covered the cell. A robot can only claim that it covered the cell when it was the first one to travel through it for the entire length of the cell. Any cell that is marked as covered is marked in the robot's local copy of the cells. The global copy of the cell is also marked at that time.

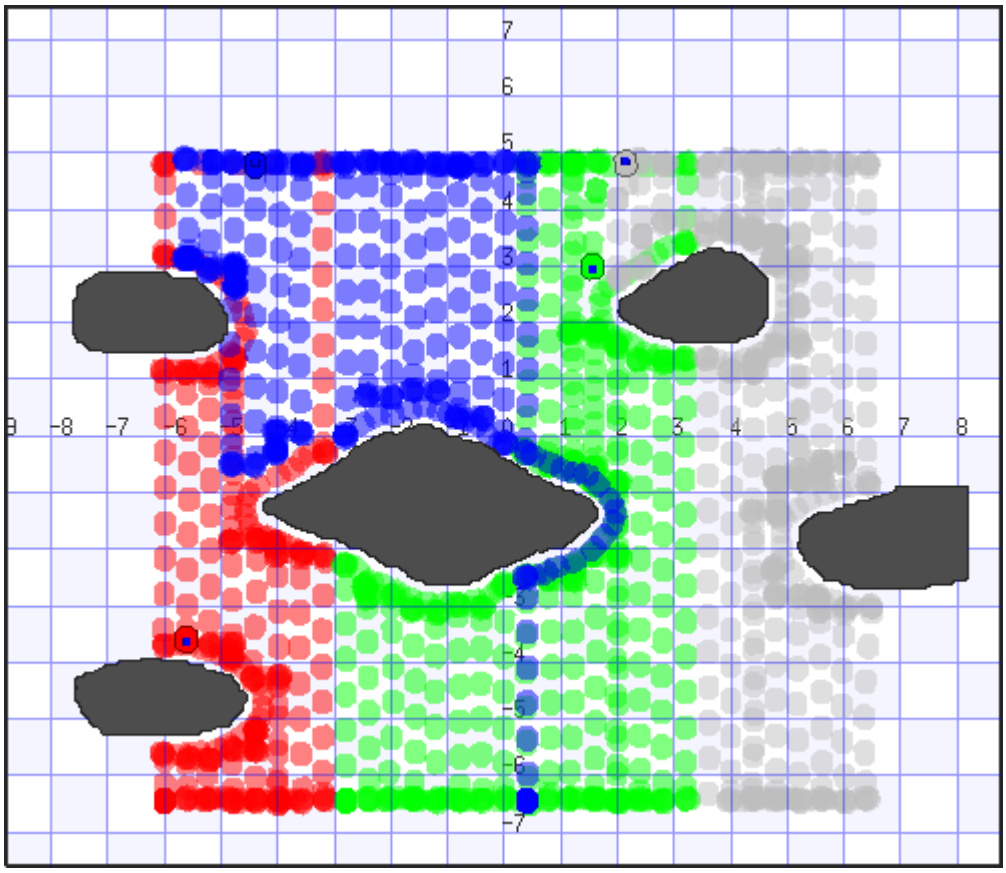

Figure 33. Complete coverage using four robots with directional constraints.

The screenshots of the covered environment for the experiments are displayed in Figure 33, Figure 34, and Figure 35. It shows the covered environments from the experiments conducted with the new algorithm and a different number of robots. Just like the experiments conducted to get the baseline results, there are no cells or portions of cells that were not covered except where the obstacles are located. These screenshots can be used for visual confirmation that no cells were left uncovered.

Three robots were used for the first experiment with directional constraints. The environment was decomposed into 30 cells and each robot was assigned 10 cells. The environment after the coverage operation is complete is shown in Figure 34. All the

robots started exploring their assigned areas at the same time. Robot 2 (green) started exploring the area in the middle, but the object made it impossible to travel around the perimeter without going outside the assigned area. While Robot 1 (red) was busy exploring the area assigned to it, it won the auction to cover the new cells from Robot 2. It moved to the new set of cells and covered them. After Robot 2 finish covering its initial assigned area, it won the area that was initially assigned to Robot 1. It then proceeded to explore that area by traveling along the perimeter of the area. After Robot 1 covered the cells won from the previous auction, Robot 2 was still busy exploring. It covered some of the cells assigned to Robot 3 (blue).

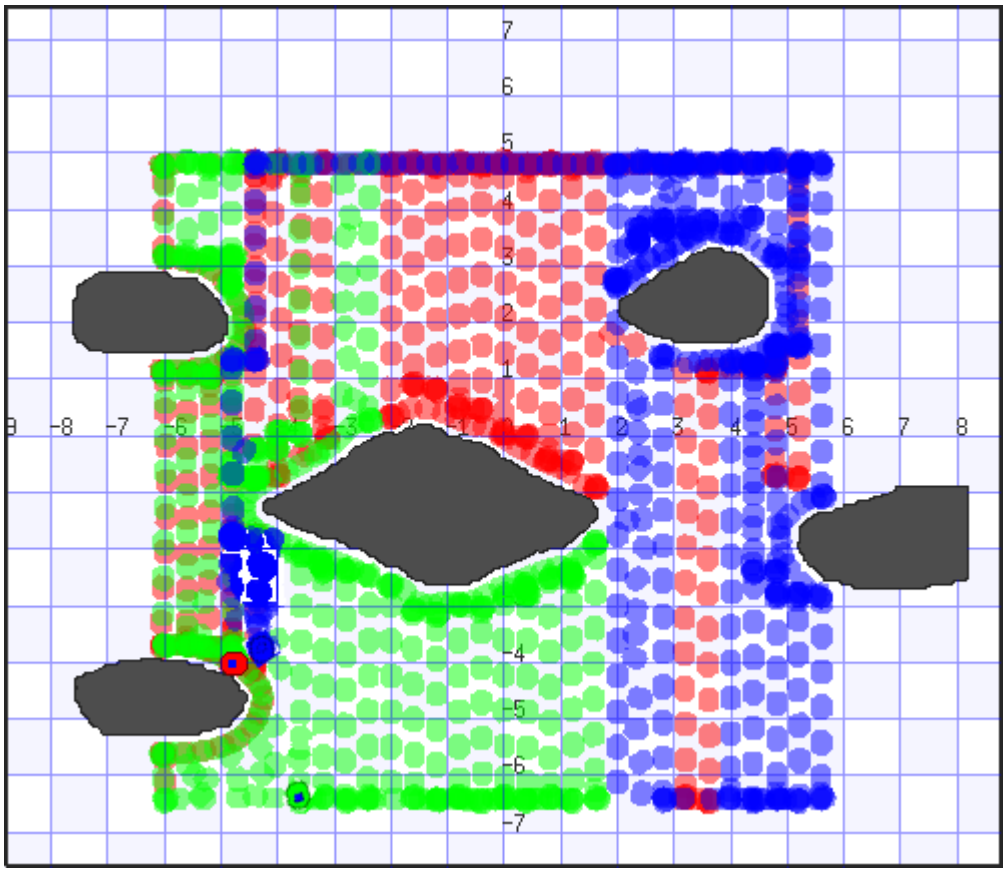

Figure 34. Complete coverage using three robots with directional constraints.

Robot 1 and Robot 3 travelled extra distances to cover cells assigned to the other robots. Figure 34 shows that Robot 1 and Robot 3 moved across the middle area to cover cells in each other's area. If Robot 1 finished the exploring task of the initial area assigned to it before it explored the unreachable area auctioned by Robot 2, the extra travel distance would have decreased since the cells closer to the robots would have been available to cover.

Four robots were used for the second experiment with directional constraints. The robots did not start the exploring task at the same time, but estimated delays were used that represent the delays used by Rekleitis et al. (2008). The environment after it was covered is shown in Figure 33. Each robot travelled a shorter distance compared to the distance travelled using three robots. The time it took to finish the coverage operation also decreased.

Five robots were used for the last experiment. Figure 35 shows the environment that was covered. No starting delays were used, and all the robots started exploring their individual assigned area at the same time. Looking at the distance travelled and the time it took to cover the environment listed in Table 4, it is clear that the amount of time was reduced and each robot travelled a shorter distance compared to the results where three and four robots were used.

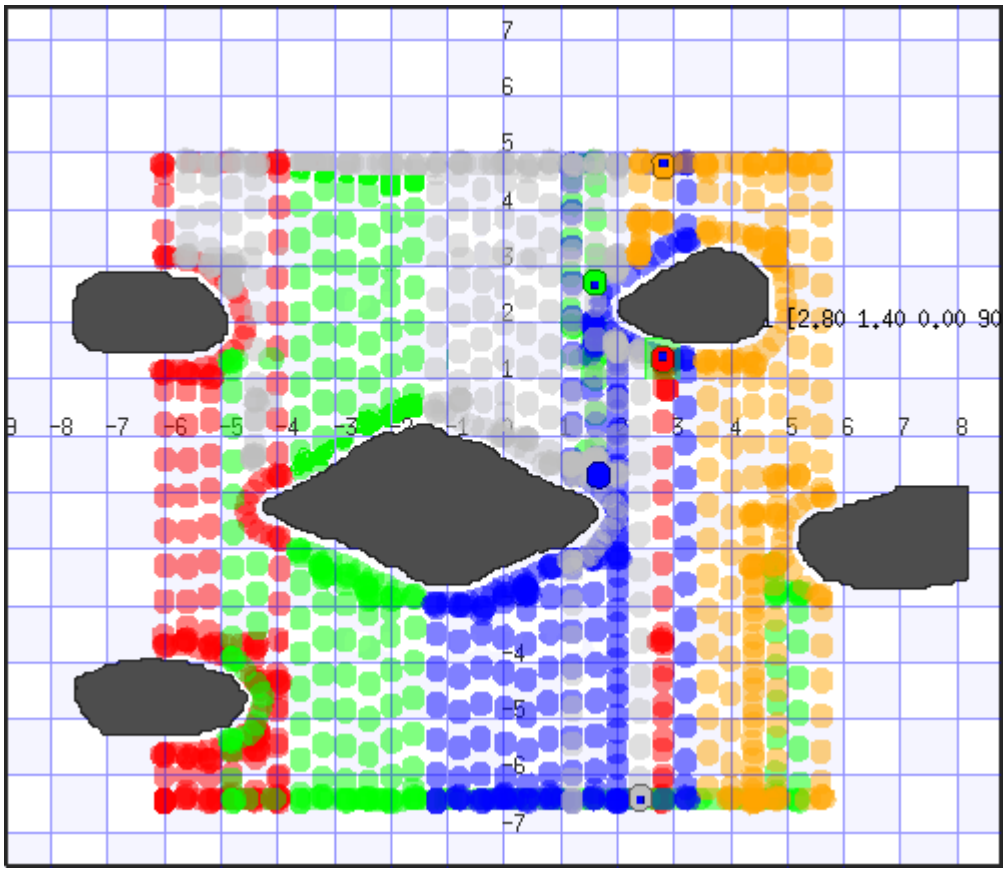

Figure 35. Complete coverage using five robots with directional constraints.

When comparing the results between the experiments, it is clear that fewer robots travelled across areas to get to uncovered cells assigned to other robots when more robots are used. This can be seen when comparing the covered areas shown in Figure 33 and Figure 35 with that of Figure 34. Having uncovered cells close to the robot's current position, requires shorter travel distances.

Summary

This chapter presented the results and findings of the research undertaken using a new complete coverage algorithm. The findings prove that the proposed solution can achieve complete coverage with directional constraints. Both the completeness of the

planned path and the adherence to the directional constraints by the new algorithm were examined. The distance traveled by each robot was examined to make sure the work is evenly distributed between robots. The ideal is that all the robots will complete the coverage at the same time.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Introduction

This chapter presents the overall summary of the research. It also contains recommendations for future research.

Conclusions

The goal of this research is to introduce a new multi-robot complete coverage algorithm that use directional constraints while doing path planning. The new algorithm was evaluated with a Java implementation running against simulation software.

The results revealed that it is possible to add directional constraints to the complete coverage algorithm and still achieve the high-level goals of complete coverage. The robots finish covering the area roughly at the same time, with about the same travel distance, while keeping the violation distances low.

Using specific starting delays can help reduce extra travel distances. To get the correct starting delay should be easier for an offline algorithm, but since the new algorithm is an online algorithm, it is not easy to get an accurate delay value. Using an incorrect delay value can also have unintended consequences since the delays can control some of the decisions and auction results. Most of the auctions are influenced by the position of the robots and the cells that are still uncovered during the bidding process of an auction. The results would be very different if all robots started at the same time.

The results in Table 4 show that the robots travel a shorter distance as the number of robots increase. Shorter travel distances reduce the total time required to complete the coverage operation.

Some of the experiments revealed that robots must travel further distances to get to uncovered cells. The reason for this is that the robots are still busy exploring their assigned areas and none of the cells in those areas are available for auction. If the robots finish the exploration task before bidding on an area that is unreachable by another robot, the cells from the initial areas become available for auction. This results in uncovered cells being closer to the current position of the robot during the auction window.

Another factor that impacts coverage time is the location and size of the obstacles. When an obstacle is detected, the robot slows down and then stops before turning away from the obstacle. Encircling the detected obstacles is a slow operation. Constant direction changes are required to keep the robots from crashing into the obstacles.

The distance a robot has to travel between points is also a factor to consider for coverage time. A shorter travel distance between points increases the time to complete the coverage operation of the environment. A robot accelerates from the starting point until the predetermined maximum speed is reached. It also must slow down before reaching the destination point.

Implications

The new algorithm allows for better path planning when there are directional constraints assigned to the cells used for complete coverage problems. As the number of robots increases, the likelihood of a collision between robots also increases. When there

are no directional constraints on the travel direction the robots can use when traveling in the cells, the likelihood of collisions and collision avoidance is quite high. This was observed but not recorded during the experiments. When all the robots are required to travel in the same direction in a specific cell, the probability of a collision is low.

Future Work

From looking at the results, it seems that the new algorithm should give us the ability to add a directional constraint to complete coverage. A few limitations were revealed while building and evaluating the algorithm. Below are a few of the limitations.

When the environment is broken down into cells, the cells cannot always be equally divided between the number of robots. Spreading the workload is important for the overall goal to reduce the time it takes to cover an area.

In some cases, the new algorithm will cover a cell more than once. A reduction in repeat coverage would lead to an increase in performance.

The direction assigned to each cell is based on the pattern that was selected. The basic lawn-striping pattern was used in this study. More complex patterns were left for future work. Choset et al. (2000) listed a few patterns and include: spiral, spike, and diamond.

Varying the start time for the robots with delays has a huge impact on the cover operation. It is a great way to test how the different conditions are handled. The conditions are triggered at different times and by the position of the robots in the environment. For example, it can control which robot wins the first set of cells that are not reachable by another robot due to obstacles blocking the cells as well as prevent

collisions between robots. It has a cascading effect and will change which robot will cover the different cells. More research is needed to look into creating dynamic delays for complete coverage.

This research could have been much easier if basic robot navigation libraries were used for the implementation of the algorithm. Future work should include this to make it easier to start a research project when robot navigation is needed in the Play/Stage simulation software.

Reference List

Acar, E. U., & Choset, H. (2000). Critical point sensing in unknown environments. *Proceedings of the IEEE International Conference on Robotics & Automation*, 4, 3803-3810. doi:10.1109/ROBOT.2000.845324

Batalin, M. A., & Sukhatme, G. S. (2002). Spreading out: a local approach to multi-robot coverage. *6th International Symposium on Distributed Autonomous Robotics Systems*, 373-382. doi:10.1.1.1.2838

Butler, Z. (1998). *CC R: A complete algorithm for contact-sensor based coverage of rectilinear environments* (Doctoral dissertation, The Robotics Institute, Carnegie Mellon). doi:10.1.1.42.7393

Choset, H. (2000). Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots*, 9(3): 247-253. doi:10.1023/a:1008958800904

Choset, H. (2001). Coverage for robotics–A survey of recent results. *Annals of mathematics and artificial intelligence,* 31(1-4), 113-126. doi:10.1023/A:1016639210559

Choset, H., Acar, E., Rizzi, A. A., & Luntz, J. (2000). Exact cellular decompositions in terms of critical points of Morse functions. *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, 3, 2270-2277. doi:10.1109/ROBOT.2000.846365

Choset, H., & Pignon, P. (1998). Coverage path planning: The boustrophedon cellular decomposition. Field and Service Robotics, 203-209. doi:10.1007/978-1-4471-1273-0_32

Dias, M. B., & Stentz, A. T. (2001). *A market approach to multirobot coordination* (CMU-RI -TR-01-26). Pittsburgh, PA: Robotics Institute, Carnegie Mellon University.

Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12): 1258-1276. doi:10.1016/j.robot.2013.09.004

Hameed, I. A., Sorrenson, C. G., Bochtis, D., & Green, O. (2011). Field robotics in sports: automatic generation of guidance lines for automatic grass cutting, striping and pitch marking of football playing fields. *International Journal of Advanced Robotic Systems*, 8(1), 113-121. doi:10.5772/10534

Huang, W. H. (2001). Optimal line-sweep-based decompositions for coverage algorithms. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on,* 8(1), 27-32. doi:10.1109/ROBOT.2001.932525

Hazon, N. & Kaminka, G. A. (2008). On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems,* 56(12), 1102-1114. doi:10.1016/j.robot.2008.01.006

Kapanoglu, M., Alikalfa, M., Ozkan, M., & Parlaktuna, O. (2012). A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time. *Journal of intelligent manufacturing,* 23(4), 1035-1045. doi:10.1007/s10845-010-0404-5

Min, T. W., & Yin, H. K. (1998). A decentralized approach for cooperative sweeping by multiple mobile robots. *In Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on,* 1, 380-385. doi:10.1109/IROS.1998.724649

Mannadiar, R., & Rekleitis, I. (2010). Optimal coverage of a known arbitrary environment. *In Robotics and Automation (ICRA), 2010 IEEE International Conference on,* 5525-5530. doi:10.1109/ROBOT.2010.5509860

Oh, J. S., Park, J. B., & Choi, Y. H. (2001). Complete coverage navigation of clean robot based on triangular cell map. *IEEE International Symposium on Industrial Electronics*, 3, 2089–2093. doi:10.1109/ISIE.2001.932038

Oksanen, T., & Visala, A. (2009). Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8), 651-668. doi:10.1002/rob.20300

Preparata, F. P., & Supowit, K. J. (1981). Testing a simple polygon for monotonicity. *Information Processing Letters*, 12(4), 161-164. doi:10.1016/0020-0190(81)90091-0

Rekleitis, I., Lee-Shue, V., New, I., Choset, H. (2004). Limited communication, multi-robot team based coverage. *Robotics and Automation. Proceedings. ICRA '04. 2004 IEEE International Conference on,* 3462-3468. doi:10.1109/ROBOT.2004.1308789

Rekleitis, I., New, A., Rankin, E., & Choset, H. (2008). Efficient Boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, 52(2-4), 109-142. doi:10.1007/s10472-009-9120-2

Scag Power Equipment (2010). Lawn striping and lawn patterns - How do they work? Retrieved from http://www.scag.com/lawnstriping.html

Senthilkumar, K. S., & Bharadwaj, K. K. (2012). Multi-robot exploration and terrain coverage in an unknown environment. *Robotics and Autonomous Systems*, 60(1), 123-132. doi:10.1016/j.robot.2011.09.005

Solanas, A., & Garcia, M. (2004). Coordinated multi-robot exploration through unsupervised clustering of unknown space. *Proceedings 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1, 717–721. doi:10.1109/IROS.2004.1389437

Vaughan, R. T., Gerkey, B. P., & Howard, A. (2003). On device abstractions for portable, reusable robot code. *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, 3, 2421-2427. doi:10.1109/IROS.2003.1249233

Wang, X. & Syrmos, V. L. (2009). Coverage path planning for multiple robotic agent-based inspection of an unknown 2D environment. *17th Mediterranean Conference on Control and Automation*, 1295-1300. doi:10.1109/MED.2009.5164725

Yao, Z. (2006). Finding efficient robot path for the complete coverage of a known space. *In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 3369-3374. doi:10.1109/IROS.2006.282514

Yong, Z., Zhang, L., & Zhang, X. (2008). Mobile robot path planning base on the hybrid genetic algorithm in unknown environment. *2008 Eighth International Conference on Intelligent Systems Design and Applications*, 2, 661-665. doi:10.1109/ISDA.2008.18