3-13-2014

MULTI-ROBOT COVERAGE WITH DYNAMIC COVERAGE INFORMATION COMPRESSION

Zachary L. Wilson
University of Nebraska at Omaha

MULTI-ROBOT COVERAGE WITH DYNAMIC COVERAGE INFORMATION COMPRESSION

A Thesis

Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment
of the Requirements for the Degree

Master of Science in Computer Science

University of Nebraska at Omaha

by

Zachary L. Wilson

March 13, 2014

Supervisory Committee:

Professor Prithviraj Dasgupta

Professor Stanley Wileman

Professor Robert Todd

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

MULTI-ROBOT COVERAGE WITH DYNAMIC COVERAGE INFORMATION COMPRESSION

Zachary L. Wilson, MS

University of Nebraska, 2014

Advisor: Professor Prithviraj Dasgupta

This work considers the problem of coverage of an initially unknown environment by a set of autonomous robots. A crucial aspect in multi-robot coverage involves robots sharing information about the regions they have already covered at certain intervals, so that multiple robots can avoid repeated coverage of the same area. However, sharing the coverage information between robots imposes considerable communication and computation overhead on each robot, which increases the robots' battery usage and overall coverage time. To address this problem, we explore a novel coverage technique where robots use an information compression algorithm before sharing their coverage maps with each other. Specifically, we use a polygonal approximation algorithm to represent any arbitrary region covered by a robot as a polygon with a fixed, small number of vertices. At certain intervals, each robot then sends this small set of vertices to other robots in its communication range as its covered area, and each receiving robot records this information in a local map of covered regions so that it can avoid repeat coverage. The coverage information in the map is then utilized by

a technique called spanning tree coverage (STC) by each robot to perform area coverage. We have verified the performance of our algorithm on simulated Coroware Corobot robots within the Webots robot simulator with different sizes of environments and different types of obstacles in the environments, while modelling sensor noise from the robots' sensors. Our results show that using the polygonal compression technique is an effective way to considerably reduce data transfer between robots in a multi-robot team without sacrificing the performance and efficiency gains that communication provides to such a system.

Dedicated to Bridgette, the light of my life.

Acknowledgements

I'd like to acknowledge and give my thanks to Professor Raj Dasgupta, Professor Robert Todd, and Professor Stanley Wileman for their time and consideration as my committee. I'd also like to acknowledge the CMANTIC Research Group for the resources and the Office of Naval Research for the funding which made this work possible.

Contents

Chapter 1

Introduction

Multi-robot systems are systems which utilize some number of robotic platforms which work in parallel to perform a given task. This sort of system is utilized in applications where a single monolithic robot is too expensive but a single small robot is incapable of performing the required task in an adequate time-frame. Multi-robot systems are also useful in situations in which robustness and resource redundancy are desirable traits. A cooperative multi-robot system is made up of interacting robotic platforms which solve a problem in a distributed manner (Parker, 1999). However, communication itself incurs a cost on real-life systems in the form of power expenditure and computational resources which are at a premium in certain types of systems. One problem for which cooperative multi-robot systems are typically used for is efficient area coverage; that is, ensuring that the sensor of a robot fully covers some region of space while minimizing overlap. This thesis addresses the problem of minimizing communication costs associated with cooperative multi-robot area coverage. Cooperative multi-robot area coverage is a challenging problem due to the fact that excluding coverage data results in an inaccurate map being shared between robots in the system. This, in turn, results in repeated coverage and additional overhead. Similarly, limiting time between communications enhances the possibility of repeat coverage between data bursts due to the fact that a robot may cover an area covered by another robot

before receiving up-to-date coverage information. Therefore, the system must communicate often enough that maps are kept reasonably up to date, yet must limit the amount of data sent while still offering a sufficiently accurate representation of the area covered by each robot to every other robot in the system.

The problem of autonomous multi-robot coverage falls into the field of autonomous robotic control, which concerns itself with the implementation of software *agents* which utilize the sensors and actuators on a robotic platform to interact intelligently with the robot's surrounding environment. Autonomous multi-robotic control has the added complication of requiring interaction between multiple robotic platforms and their controlling agents to ensure that the system as a whole works in a coordinated and efficient manner. Such a system of multiple robots with interacting agents is called a *multi-robot system*. In the study multi-robot systems, there is a strong focus on time efficiency and completing goals with the minimum distance travelled possible (Gabriely and Rimon, 2003). However, little research which addresses the approach of minimizing communication while limiting potential time and distance costs has been done.

In this thesis, the problem of limiting data transfer between agents in a multi-robot system as applied to the coverage problem is accomplished by utilizing *polygonal approximation* to perform compression on coverage information. The main idea of our compression scheme is to significantly reduce the amount of coverage information communicated between robots, while slightly sacrificing the exactness of the coverage data. The primary contribution of this thesis is the novel application of polygonal approximation to coverage data compression, for purposes of limiting data transfer in multi-robot coverage systems.
To show the operation of this system, we have applied the technique specifically to the problem of robotic coverage of an unknown environment with obstacles inside the environment. The utilization of robotic systems to cover an environment can be applied to humanitarian

de-mining (the removal of land mines from past conflict areas for the benefit of civilians), application of chemicals to crops, vacuuming floors, and any other situation where it's desirable to have every position within an area subject to a sensor or tool. In this research, an accurate simulated model of the Corobot robot from CoroWare has been created within the Webots robot simulation suite. This is a relatively cheap commercially available wheeled robotic platform which runs on Windows XP or Linux; a larger, outdoor-capable variant called the Explorer robot is also available at an increased price. The primary advantages to using a commercially available, off-the-shelf robotic platform are economy of scale in production results in lowered costs and that hardware support from the original vendor is available, as well as the simple fact that any algorithm implemented on this hardware can be put immediately into use.

Our experimental results show that the premise of the algorithm is sound; a super-linear increase in the team's coverage and effectiveness is shown, demonstrating that severely restricted, approximate coverage information exchange still permits effective cooperative behavior in multi-robot teams.

The rest of this document will examine the topics discussed in this chapter more closely. Chapter 2 covers the body of previously performed academic work relevant to the topic. Chapter 3 describes the basic, single-robot portion of the coverage algorithm utilized by our implementation. Chapter 4 presents the novel aspect of our algorithm, the use of polygonal approximation to compress area coverage information for communication between robots in a team. Our experimental results are presented and discussed in chapter 5, and this work is summarized in chapter 6.

Chapter 2

Related Work

In this chapter, we will introduce work currently applicable to the area of multi-robot coverage and dynamic coverage information compression. In the first section, we will introduce mobile robots and software agents. In the second section we will introduce multi-robot coverage and discuss why it is superior to single robot coverage. Finally, we will introduce work pertaining to area information compression and how it applies to systems with multiple robotic platforms.

2.1 Mobile Robots and Software Agents

A *robot* is defined as a mechanical device which is capable of gathering information about its surrounding environment using sensors and interacting with the environment via actuators; a mobile robot is a robot which has actuators that allow it to move around within the surrounding environment. The logic which maps sensory input to mechanical output is called a *controller* which is a type of *software agent*. Software agents perform tasks *autonomously* (without direct human intervention).

The ability of a robot to act autonomously based on what it senses in its environment makes it a flexible tool for use in applications where tasks are too dangerous, too expensive, or too

sensitive to error to be performed by human beings. Static robots are used in automobile assembly lines to rapidly and precisely weld vehicles together, while mobile robots can take the form of agricultural tractors (Reid et al., 2000), for example. Mobile robots can also be designed for tasks such as humanitarian de-mining, search and rescue, and battlefield surveillance. However, demanding applications can take a large amount of time for one robot to perform and large, expensive robots are not ideal for hazardous environments which may result in a loss of hardware. In these situations, a system which utilizes multiple less capable, less expensive robots which work in parallel has advantages over those which use a single more capable, yet more expensive, robot.

2.2 Multi-Robot Coverage

The use of multiple robots to accomplish a task is a broad subject; however in this thesis we will focus primarily on how a group of robots can be used to *cover* (touch every part of) an environment. A group of robots can be used to accomplish such a common task through coordinated use of the sensors and actuators available to the group as a whole and in doing so can complete the task more quickly than a single robot. Such systems can utilize multiple overlapping sensors to reduce the effects of sensor noise (Stachniss et al., 2008). This permits scalability, as robots can be added as task size increases and robustness, as the failure of an individual robot does not result in failure of the task.

There are challenges introduced when utilizing multiple robots to perform coverage which either don't exist or are trivial when implementing a single-robot solution. These include, but are not limited to (a) avoiding repeated coverage to limit time and energy waste; (b) storing and communicating coverage information to permit intelligent coverage behavior; and (c) permitting the task to continue when robots move out of communication range. Examples of this style of approach are *target utility based exploration* (Burgard

et al., 2005), *cooperative rectilinear environment coverage* (Butler, 2000), *pheremone-based coverage* (Koenig et al., 2001), *Boustrophedon coverage* (Rekleitis et al., 2008), and *segment-partitioning exploration* (Wurm et al., 2008). While none of these approaches are utilized in our algorithm, details follow for the purpose of comparison.

In *target utility based exploration*, multiple robots are assigned target points based on the cost to the robot to reach that point and the expected utility of exploring that point; assignment of a point to a robot reduces the expected utility of the environment area visible from that point. Any given area is ”explored” or ”unexplored” as a binary value and coordination is performed without consideration to the amount of information being sent between the agents performing the work. The system detailed is able to map the initially unknown environment utilizing ranged sensors and unlimited communication in both real and simulated environments.

Cooperative rectilinear environment coverage, in contrast to the previous algorithm, performs actual coverage of the environment rather than simply building a map of the area. This algorithm is designed around the use of square robots with minimal sensing capability – contact sensors only – to cover bounded environments which are able to be split up into a finite number of discrete rectangles (environments whose boundaries and obstacles are straight and intersect at right angles). The free space of the environment is bounded by the robots, which identify points at which partitioning of the environment would produce a convex rectangle containing no unreachable points; said rectangle then becomes a subtask to be assigned to one of the cooperating robots. The algorithm is shown to be complete via proof and simulation for the subset of environments for which it was designed; however, hardware experimentation was not done and specifics with regard to communication were not discussed. This is a centralized algorithm with an overseer which collects information about the environment and assigns tasks to the robots.

Pheromone-based coverage utilizes very limited robotic platforms which use non-optimal real-time search techniques and alleviate the need for on-board maps by leaving markers in the environment which signify that the area in question has been covered in the past; said markings can be sensed by every robot in the team and therefore not only serve as a means of replacing on-board memory, but also as a means of communication between the agents in the system. Such an approach has clear advantages in terms of cost savings in hardware, but also replaces fairly compact electronic components with an actuator which is capable of altering the environment in a measurable way and a sensor which is capable of detecting those alterations. This adds complexity to the hardware which, at least partially, offsets the benefits of the algorithm's simplicity in real-world applications.

Boustrophedon coverage has two primary modes of operation: the restricted and unrestricted communications cases. In restricted communication scenarios, robots split into two teams – one for exploration and mapping of region boundaries and one for coverage of the mapped regions. In the unrestricted communication scenario, the exploration and coverage tasks occur simultaneously. In each case, the environment is partitioned into strips which are further partitioned into cells: in the unrestricted case, partitions are then auctioned off to robots based on cost of each robot to cover the indicated strip. Actual coverage is a fairly typical cellular decomposition and cellular coverage problem. In either case communication occurs frequently even if the amount of information shared is restricted: robots are aware of each others' locations. The unrestricted case is shown to be efficient and complete, while the restricted case – due to coverage being restricted by the speed of the exploration team – can result in significant idle time for the coverage team.

Segment-partitioning exploration is an exploration algorithm which splits up the boundary between explored and unexplored area and assigns those partitions to the robots in a way

that minimizes sensor overlap and travel costs. Partitioning and assignment of the robots to ”frontier” regions is done on-line. Communication is not a consideration in this algorithm, nor are the details of the map representation.

There are two categories of coverage algorithms which will, in contrast to the above, be the primary focus of this thesis. These are algorithms which accomplish one of the desired tasks – either complete coverage or complete minimization of communication – and thus are of great interest as starting points for our work. The first approach, *dispersion-based coverage*, emphasizes simplicity over efficiency. In dispersion-based algorithms, robots do not store or exchange any coverage information and disperse themselves via *potential fields* (Batalin and Sukhatme, 2002; Howard et al., 2002).

Dispersion by the use of potential fields is accomplished by assigning a virtual repulsive force to each object the robot should be avoiding, then summing those forces to generate a composite vector. The directional component of the resulting vector is the direction in which the robot should travel to avoid the objects in question. This technique is relatively well-established and commonly used for the purposes of dispersion (Howard et al., 2002). Potential fields dispersion has the advantage of requiring only positional information from its environment (if avoiding obstacles and/or environmental features) and/or robots (if a robot is dispersing from its teammates). No higher-level processing is necessary; however, as a result, repeated coverage can be a significant drain on algorithmic efficiency.

The second approach is to build a cellular graph which models the environment, and allow the robots to construct the least-cost spanning tree of the graph; examples include *MSTC* (Hazon and Kaminka, 2008) and *collaborative on-line cellular swarm-based coverage* (Rutishauser et al., 2009). *MSTC* is a multi-robot implementation of the STC algorithm originally proposed by Gabriely and Rimon, which is utilized as a component of our own

algorithm. In STC, an area is partitioned into a cellular grid composed of square cells; each cell has four sub-cells which are roughly the size of the footprint of the covering robot. Cells are created in a spanning tree from the cell at the robot's initial position, and traversal of the tree is done using depth-first search. Each logical edge traversal prompts a physical cell traversal. In MSTC, the STC path is computed and then partitioned into sections for traversal by multiple robots, which know the complete tree and follow their subtrees of the path. Communication is considered in this work only as a requirement, but implementation details are not discussed. However, robustness and efficiency of motion/time are shown to be excellent in the best case. Running time depends heavily on the robots' initial positions; worst-case performance is shown to be approximately equivalent to the single-robot STC performance.

Collaborative on-line cellular swarm-based coverage is an algorithm designed for complete coverage using a large number of small robots with error-prone sensors. Limits in communication range are considered, though no hardware testing is performed in this work. In the single-robot case, the environment is decomposed into cells which are traversed by robots moving to the closest (in terms of graph distance) uncovered cell. In multi-robot cases, collaboration is performed by the robots broadcasting their complete map of covered cells every time a new cell is covered, which allows the other robots to add any cells in that robot's covered set to their covered set, and delete those cells from their uncovered sets. This is a very robust method of map data transfer, but is also exceptionally costly. Throughout the system, maps are transmitted once for every cell in the map; therefore, the system-wide efficiency of the communication of the coverage data is $O(n^2)$.

Our approach fuses elements of classic, communication-centric coverage algorithms, potential field dispersion, and single-robot spanning tree coverage algorithms in order to maximize coverage while minimizing the amount of data transmitted between robots. The en-

vironment, as a result, is decomposed in layers, all computed and explored on-line while communication is limited to very small bursts of positional data only when coverage of a significant subsection of the map is completed.

2.3 Map Information Compression

While modern computing hardware has reduced the need for minimizing the size of datasets in most computational situations, mobile robotic platforms come in many different forms; some of these forms have limited working memory, communications bandwidth, or both. In such situations, compression of the explored area information can have noticeable benefits in terms of memory used and communications overhead when transmitting that data.

Work has been done on the topic of map compression in the past; however, the techniques in question have limitations which make those approaches unsuitable for use in the context of mobile exploration of an unknown environment.

Both approaches we shall examine model the environment as a set of vectors, rather than a rasterized cellular map; as previously discussed, our model of the environment and the regions which our agents have explored is also in vector form, using continuous coordinates as opposed to a discrete cellular model.

The first approach utilizes correlation between observations by static sensors in proximity to one another to ensure compressibility of the aggregate output of the sensor network (Baek et al., 2004). Optimal compression is achieved by distributing the sensors in such a way that correlation is maximized. The weakness in this approach for our use case is that mobile robots performing coverage in an unexplored environment can not ensure that all sensor observations will be optimally distributed for maximization of correlation between

observations: the algorithm depends on a known and unchanging position from which measurements of the environment are made; this is incompatible with our problem definition.

The second approach performs dictionary encoding on the map information prior to transmitting or storing the data, which relies on common elements within the vector information to construct a dictionary with which to compress the information (Shekhar et al., 2002). This is incompatible with our use case as well, as dictionary compression requires the creation of a new encoding dictionary every time new information is added to the dataset to be compressed. Our algorithm explores an unknown area and is continually updating its map with a list of previously covered regions within that area; consequentially, the coverage map is constantly changing. Change in the dataset to be compressed causes continual reconstruction of the encoding dictionary and, for other robots to be able to decode that information, the updated dictionary must be sent with the compressed data every time the dictionary changes. This causes a large amount of communications overhead and largely negates the efficiency gains made by compressing the map coverage data.

Our algorithm instead uses a mathematical approximation of the enclosing polygon of the area covered by a robot via usage of the *min*-ε algorithm (Perez and Vidal, 1994). This algorithm outputs a polygon with a user-defined number of vertices which most closely approximates a given source polygon with a larger number of vertices. This causes a loss of coverage information, but permits any given region of coverage to be represented in constant space.

Chapter 3

Coverage

In this chapter, we will discuss the problem of performing complete coverage with a team of mobile robots. We will also examine applications of multi-robot coverage as well as sources of inefficiency and overhead, and introduce the rationale for our approach. We will then present the simulator we used to develop our approach, the simulated robotic platform used, and the real hardware it simulates. Finally, we will explain the single-robot coverage technique used by the robots in our solution.

The problem of coverage can be defined informally as ensuring that a sensor, actuator, or other device touches every part of an area. It is assumed for our purposes that said area is a patch of ground represented in two dimensions. This is a common real-life problem which occurs in agricultural, military, and humanitarian situations such as crop-spraying, de-mining, and search and rescue operations. A typical coverage algorithm must determine what areas need to be covered and assign each coverage subtask to an individual robot. Each robot must then travel to the region it was assigned and perform coverage while storing the information obtained via sensor into a map. That map is then shared in some way with the robot's teammates (Stachniss et al., 2008).

The effectiveness of a given coverage algorithm can be measured in two ways: completeness of coverage (which should be maximized) and repeated coverage (which should be minimized). In order to have any appreciable effect on these values, a map of the space the robot has previously covered is generally required, and allows the robot to remember where it has been and where it has not yet covered. This requires the robot have adequate memory to (a) store the complete map; (b) store some subset of the map or (c) store an approximation of the map.

In coverage algorithms utilizing multiple robots, preventing repeated coverage requires that each robot not only builds a map of the space it has previously covered, but also communicates that map to the other robots so that those robots will not cover the area it has already covered. This requires the robot have energy to accomplish the communication, as well as adequate time and communication speed to send the complete map, a subset of the map, an approximation of the map, or an incremental update of the changes to the map since the last communication sent.

Limiting factors on a multi-robot system's coverage effectiveness are inefficiency and overhead. Inefficiency is defined as coverage that occurs in excess of what is required (repeated coverage) while overhead is defined as robot travel between periods of coverage, computations spent on map operations and both computation and time spent communicating with other robots. Ideally, minimizing inefficiencies and overhead is the idea. However, there exists a trade-off between these two. Reducing inefficiency usually has an overhead cost and reducing overhead usually has an efficiency cost in practice. Finding a balance between the two factors which minimizes the total detriment to the algorithm as a whole results in a more capable algorithm than an algorithm which focuses on minimizing one factor and does not consider the other; this is called cost minimization.

Every action a robot takes incurs a cost in the form of energy used and time passed while the task is accomplished. When considering energy costs, a mobile robot without a tether has finite energy stores which must be carried with the robot itself. From a perspective of cost and robustness, minimizing the amount of energy which must be stored on a robot in order for that robot to complete a particular task also results in a minimization of cost for that robot, and the development of efficient algorithms which limit the waste of energy minimizes the cost of the system required to run that algorithm. In the case of the use of off-the-shelf hardware, maximizing the area which can be covered by a single robot before energy depletion allows fewer units to be used to complete the system's assigned task, which reduces hardware costs of the system. In both situations, minimizing energy expended in movement, communication, and data processing has real-world benefits in terms of monetary cost.

In certain situations utilizing very small robots with limited capabilities – such as search and rescue and extraterrestrial exploration – time is also a significant factor: in the former case locating severely injured people in a timely manner is extremely important and exploration of tight spaces may require very small platforms, while in the latter case platform size and weight is limited due to the prohibitive costs of launching heavier, more capable systems. In both cases, time costs incurred in map computation operations and map communication may be unacceptable, and as such the amount of time spent on overhead should be minimized.

Traditionally, multi-robot coverage algorithms have taken two primary forms: the first is the dispersion-based approach, which minimizes overhead by not communicating at all, but makes no attempt to prevent repeated coverage; while the second requires agents to send complete or partial map data frequently and guarantees that minimal or no repeat coverage takes place. This thesis will present an algorithm that combines these approaches to maxi-

mize coverage while minimizing both repeat coverage and communication.

3.1 Algorithm Description

The algorithm that is the focus of this thesis has two primary components. The first is the main coverage algorithm which handles the sensing, navigation, and coverage of the robot's environment, and that is the portion that will be covered in this section. The next chapter focuses primarily on the contributions of the author with respect to how the map is stored and communicated between robots using polygonal approximation. We call this the *Polygonal Approximation Coverage algorithm* or *PAC algorithm*.

The robot's controller is constructed as a finite state machine which changes between reactive behaviors according to certain conditions; the behaviors in question are dispersion, region/block definition, and region/block coverage. See Figure 3.1 for a visual representation of the algorithm's state machine and Figure 3.2 for the region coverage algorithm in pseudocode. The initial state is dispersion, where the robot examines its map and moves away from previously covered areas and the boundaries of the work area using potential fields until the robot reaches a local minima in the field. At this point it designates a new region, and enters the region covering phase of the algorithm. Regions are represented as polygons while work area boundaries are represented as line segments; however, the virtual repelling forces which act on the robots are point sources. Regions are represented as point sources

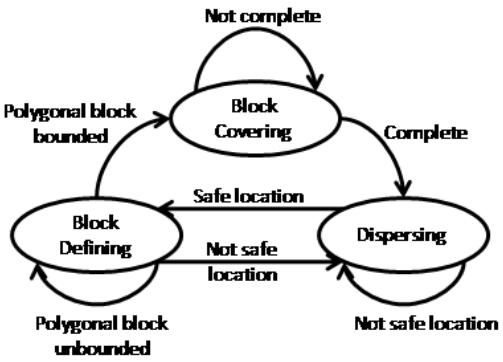


Figure 3.1: Robot controller state diagram.

by their centroids, whilst work area boundaries are represented by the point nearest the robot which lies upon the line segment. Force intensity is directly proportional to the area of the region in question, for regions, and equal to the area of the largest region for work area boundaries; forces fall off linearly with distance. The algorithm by which the dispersion angle is calculated on each iteration of the controller while in the dispersion state is more formally defined in Figure 3.3.

The region covering phase of the algorithm is split into 3 distinct parts: polygon definition, polygon bounding, and polygon coverage. A region is initially defined as a single polygon of either three or an even number of sides. However, upon completion of the coverage process of a given polygon P, a new polygon P+1 is defined with its first edge directly opposite the first edge of the previous polygon P. Polygons are, as a result, constructed in a continuous strip such that a straight line can be drawn through the centroids of P, P+1, and P+2, as seen in Figure 3.4. Polygon bounding occurs immediately after polygon definition: the robot moves from vertex to vertex of the polygon, bounding its exterior and discovering any obstacles which intrude or bisect it and discovering which vertices are reachable and which vertices are not. After this is accomplished, the robot performs coverage of the interior of the polygon boundary using the single-robot spanning tree coverage (STC) algorithm (Gabriely and Rimon, 2003).

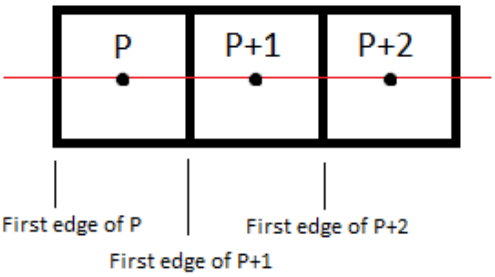


Figure 3.4: Diagram showing the polygon chaining behavior. Note that a single line can be drawn through the centroids of all three polygons.

Throughout the algorithm, various reactive behaviors can be activated and deactivated by conditions encountered by the robot and the robot's current state, which are illustrated in

Figure 3.5. For the polygon bounding portion of the algorithm, behaviors include:

- edge following (in which the robot moves along a virtual edge between two points in two dimensional space) which is triggered by the polygon bounding state,
- obstacle avoidance (in which the robot sharply turns to avoid an obstacle in its path of travel) which is triggered by the front-facing infrared sensors detecting an object,
- wall following (in which the robot maintains a distance from an obstacle on either its right or left side) which is triggered by the obstacle avoidance behavior while in the polygon bounding and dispersion states.
- block aborting (in which the robot detects that it's either moved into a previously covered region, or has encountered an obstacle it cannot get around)

The obstacle avoidance behavior overrides all other behaviors to prevent a damaging encounter between the robot and obstacles in the environment, which leads to the wall follow behavior. The wall follow behavior only terminates when the robot crosses from the interior to the exterior of the virtual polygon it was attempting to bound, at which point the robot notes the edge it has just crossed, marks any vertices between the edge it was on when it began the behavior and the current edge as unreachable, and resumes the edge following behavior on the newly encountered edge to continue the bounding task.

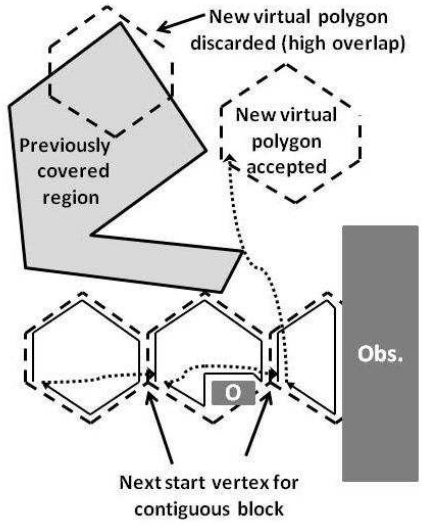


Figure 3.5: Diagram showing the edge following, obstacle avoidance, wall following, and block aborting behaviors.

Note that these behaviors are exclusive to the bounding state and do not take place while actual coverage occurs: the recursive STC algorithm instead begins by splitting each polygon into a discrete cellular grid – each cell of which is split into four sub-cells – which it then covers by selecting a cell, scanning it for obstacles, and moving into one of the sub-cells of the cell. When the chain of cells which the robot has been covering comes to an end due to the robot either encountering the polygon boundary or an obstacle, the robot then backtracks physically and logically, moving through the uncovered sub-cells of its previously covered cells as it does so. It then finds the last cell with another valid, unoccupied neighboring cell and begins to recurse from that point down another branch of the tree of cells.

Figure 3.2: Pseudocode for region-based coverage algorithm.

```
state ← NEW-REGION-SETUP
while state! = STOP do
    if state = NEW-REGION-SETUP then
        disperse();
        if accessibleAreaFound() = true then
            state ← NEW-POLY-SETUP
        else
            state ← STOP
        end if
    else if state = NEW-POLY-SETUP then
        if lastPolyVerticesUnaccessible() = true then
            state ← NEW-REGION-SETUP
        else
            define-poly-in-random-direction();
            state ← DEFINE-BOUNDARY-EDGE
        end if
    else if state = DEFINE-BOUNDARY-EDGE then
        if checkForObstacles() = true then
            state ← DEFINE-BOUNDARY-AVOID-OBSTACLE
        else if allVerticesVisited() = true then
            state ← TRAVEL-TO-CENTER
        else
            followEdge();
        end if
    else if state = DEFINE-BOUNDARY-AVOID-OBSTACLE then
        if checkForObstacles() = true then
            avoidObstacle();
        else
            state ← DEFINE-BOUNDARY-WALL-FOLLOW
        end if
    else if state = DEFINE-BOUNDARY-WALL-FOLLOW then
        if isInPoly() = true then
            followWall();
        else
            state ← DEFINE-BOUNDARY-EDGE
        end if
    else if state = TRAVEL-TO-CENTER then
        travelToCenter();
        state ← STC
    else if state = STC then
        performSTCCoverage();
        state ← NEW-POLY-SETUP
    end if
end while
```

Figure 3.3: Dispersion Angle Calculation

Given that all polar coordinates are with respect to a robot d and that:

$\mathbf{d}$: position of robot

$\mathbf{R}$: set of regions

$\mathbf{B}$: set of boundaries

$\mathbf{R_i^c}$: centroid of region i

$\mathbf{B_i^c}$: closest point to robot d

$\mathbf{R_i^A}$: area of region i

$\mathbf{dist}(\mathbf{p},\mathbf{d})$: distance between point p and robot d

$\mathbf{bear}(\mathbf{p},\mathbf{d})$: angle between point p and robot d

$\mathbf{cart}(\mathbf{r},\theta)$: global cartesian point represented by polar coord. (r,θ) w/ respect to robot d

$\mathbf{polar}(\mathbf{x},\mathbf{y})$: polar point w/ respect to robot d represented by global coord. (x,y)

$\mathbf{a_{max}}$: area of the largest region in R

$\mathbf{p_x}$:x coordinate of point p

$\mathbf{p_y}$:y coordinate of point p

$\mathbf{p_\theta}$:θ coordinate of point p

...and...

$$\forall r \in R, (D \leftarrow D \cup dist(r^c, d), A \leftarrow A \cup r^A)$$

$$\forall b \in B, (D \leftarrow D \cup dist(b^c, d), A \leftarrow A \cup a_{max})$$

Then the angle at which the robot disperses is:

$$polar\left(\sum_{i=0}^{||R||}\left[cart\left(\frac{A_i}{D_i}, bear\left[R_i^c, d\right]_x\right)\right], \sum_{j=0}^{||R||}\left[cart\left(\frac{A_j}{D_j}, bear\left[R_j^c, d\right]_y\right)\right]\right)_\theta$$

Chapter 4

Map Compression by Polygonal Approximation

The primary contribution of this work is not in the coverage algorithm itself, but how the coverage information is stored in memory. While detailed cellular coverage maps are held for long enough to cover each polygon within a region, cellular maps are also large and of variable size. A more abstract representation which approximates the cellular data can be retained between region coverage operations. Approximation of the original coverage information requires less memory to store and less time and energy to transmit than the full map. The detailed cellular data gathered by the coverage algorithm shall be called *short-term memory*, and the approximate polygonal region representation derived from that data shall be called *long-term memory*.

4.1 Polygonal Representation of Regions

The storage of map data can be performed in one of two major ways: raster-type discrete cellular maps (as discussed in several of the algorithms in Section 2.2) or vectorized continuous maps (as discussed in Section 2.3). In Section 2.3, it was briefly stated that the proposed algorithm represents regions previously covered by the robots in vector form;

more precisely, each region is a convex polygon, stored as a set of points P and edges E. Point and edge data is redundant in the case of convex polygons; that is, with the information in one set, the other can be reconstructed. Therefore, to reduce overhead, only P is sent between robots when coverage information is shared.

4.2 Polygon Compression

Compression of the information about the robots' coverage histories is extremely important. Coverage information is captured as a series of points which are collected during the polygon bounding process and during the STC algorithm. Transferring these points as a whole would be resource-intensive, and the points themselves are not of much utility to the robots. Transforming the point data into a useful, compact form is accomplished in two steps: first the coverage data of a given region is trimmed down to just those points which make up the convex hull of the whole dataset, then the resulting convex hull is approximated via the min-ε algorithm. These steps are illustrated in Figure 4.1.

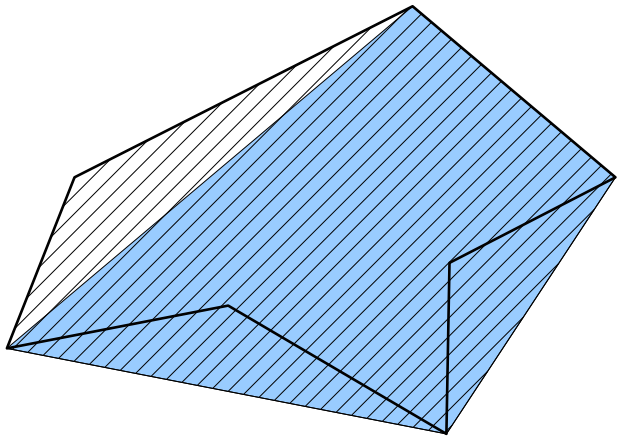


Figure 4.1: Illustration of the polygon compression algorithm. The bold outlined area is the original area, the crosshatched area is the convex hull of the original area, and the shaded area is the min-ε approximation for $n = 4$ points.

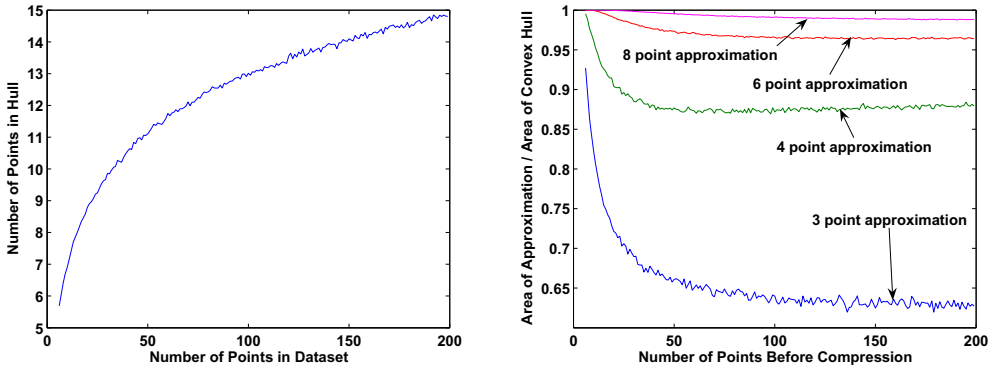
The convex hull is a well-documented problem in computational geometry; the two-dimensional

convex hull is a trivial problem with many pre-existing algorithms used to solve it. The algorithm used in this work is the *gift-wrapping algorithm*, which sacrifices efficiency for ease of implementation: it is an output-sensitive algorithm with efficiency $O(nh)$, where n is the number of input points and h is the number of output points. Better algorithms exist which can reach a running time of$O(n*\log h)$. The output of this algorithm is a set of points which represent the vertices of a convex polygon. A convex hull operation is capable of greatly reducing a set of coverage data, not least because a great deal of coverage information is gathered during the STC process; this comes at the cost of occasionally erroneously including area which has not been covered into the "covered" region. The resulting average number of points output by the convex hull algorithm given for between 4 and 200 random input points is shown in Figure 4.3a; 200 input points is reduced to, on average, under 15 output points.

The *min*-ε algorithm (Perez and Vidal, 1994) is a less well-known algorithm which generates the least-error approximation of a convex polygonal curve of n points using a maximum of m points, where m is defined by the user. This permits finding the polygon of a constant number of vertices (and thus a constant amount of data) which best approximates the convex hull found in the first step of the compression algorithm; by sacrificing data fidelity, we are able to compress a large number of data points to a single region of known, constant size. Figure 4.3b shows the accuracy of approximations

```
void compressPolygon returns double e,
                Polygon newPoly
inputs: VertexSet oldPoly; int m;
variables: Matrix distance, indices;
//initialize matrices
m ← m + 1;
indices[0..oldPoly.size][0..oldPoly.size] ← ∞;
distance[0..oldPoly.size][0] ← ∞;
//minimum-e computation
for i = 1 to m
    for j = i to oldPoly.size
        distance[j][i] ← min(distance[i-1][i-1] +
                squaredError(i-1, j,oldPoly),...
                ...,distance[j-1][i-1] +
                squaredError(i-1, j, oldPoly));
        indices[j][i] ← minIndex(distance[i-1][i-1] +
                squaredError(i-1,j, oldPoly),...
                ...,distance[j-1][i-1] +
                squaredError(i-1, j, oldPoly));
//backtracking
newPoly[m-1] ← oldPoly.size - 1;
for i = m TO 0 descending
    newPoly[i-2] ← indices[newPoly[i-1]][i-2];
e ← distance[oldPoly.size - 1][m - 1];
return newPoly; e;
```

Figure 4.2: Pseudocode of *min*-ε algorithm



(a) Hull size for values of n (b) Fidelity of *min*-ε for values of n and m.

Figure 4.3: Results of performing convex hull and *min*-ε algorithm on random datasets of size n and number of output points m.

of randomly generated convex polygons of

various sizes for several different approximation sizes, while Figure 4.2 shows the algorithm itself. More vertices in the approximation results in less error at the cost of additional memory; for the experiments in chapter 5, we selected $m = 4$ points to demonstrate the algorithm effectiveness when using a small number of vertices with a relatively large amount of error.

We believe that our approach – which values the compactness of the data over the accuracy of the data – is very suitable for multi-robot coverage applications. Our compression ratios, which – in the case of a 4-point approximation – can reach a reduction in data size of 98% for a 10% loss of fidelity compare favorably to the relatively small compression ratios achieved by standard, lossless data compression algorithms such as DEFLATE, as shown in Figure 4.4.

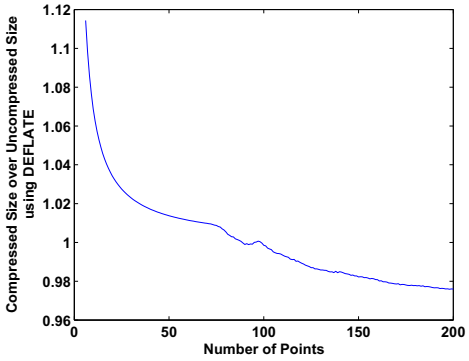


Figure 4.4: Average data compression ratio of random point sets using the DEFLATE algorithm.

4.3 Region Combination

In situations where a robot receives information about a covered region which overlaps another region in its memory in some way, it would be beneficial to combine those regions in some way if they overlap substantially and if the combination operation does not introduce too much error into the resulting combined region. In this section, a relatively straightforward method of region combination will be discussed.

Upon receipt of a communicated region, a robot checks for overlap between that region and all regions currently in its memory. When an overlap is detected, the robot combines the regions through the following process: first the two regions' vertices are combined into a single set. The convex hull of the set is then approximated with the min-ε algorithm. The resulting approximation is the combined region.

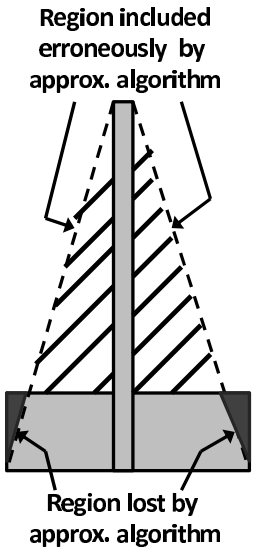


Figure 4.5: *Min*-ε region combination of two overlapping regions where $m = 4$. Solid areas indicate original polygons; the dark gray area indicates explored area lost during combination, while the lined area indicates the uncovered area included during combination.

A combined region can either be accepted or rejected based on two metrics: the area erroneously included in the approximate combined region which is not a part of either of the original regions, and the area in the original regions which falls outside of the approximate combined region (see Figure 4.5). These are expressed as ratios; the first is measured in comparison to the total area of the approximate combined region, while the second is measured in comparison to the combined area of the original two regions. In either case, high ratios indicate a poor fit which should be rejected,

while low ratios indicate a good fit which should be retained; the exact point where a metric indicated a good or poor fit depends on the application and can be set by the user of the system.

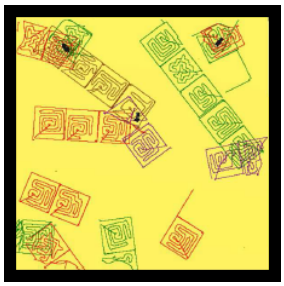
Chapter 5

Experimental Results

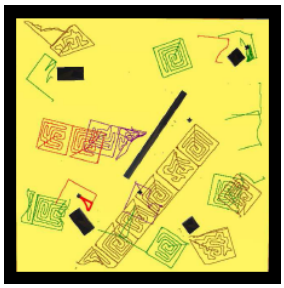
In this chapter, we discuss how the PAC algorithm was tested and the performance of the both the compression phase on its own as well as the effect of the compression and communication phases with respect to reference algorithms.

5.1 Hardware and Software

In this work, the coverage system has been implemented using Webots version 6.1.3, a realistic robot simulator which models physical interactions between objects in the environment, sensor noise, the effect of obstacle coloration on sensors using emitted light, and other real-world complications encountered in an actual work environment. Webots offers the advantage of including these complications, but permitting multiple experiments to be run rapidly. It simulates in faster than real time while also removing complicating factors such as physical space constraints, cost of purchasing many robots for larger scale experiments, and time lost in recharging and servicing actual hardware. Instead, each controller is instantiated as its own process while the physics simula-



(a) Empty (0% obstacles) arena.

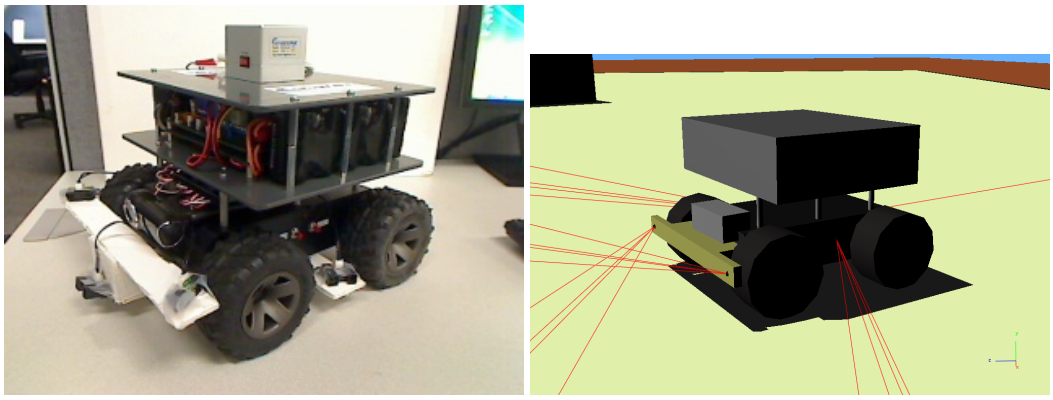


(b) 10% obstacles arena.

Figure 5.1: Webots used as a simulation environment.

tion and visualization are handled in the main simulator process,

which permits true modeling of interaction between asynchronously controlled robots in an environment which is not forced to slow down and wait for inefficient robot controllers.



(a) A Corobot with Stargazer module. (b) A simulated version of the Corobot.

Figure 5.2: Comparison of real and simulated Corobots.

The robot modeled in Webots for this work is the Corobot (manufactured by CoroWare Incorporated), configured to mirror the real hardware used for experimentation in the C-MANTIC laboratory. It is a 4-wheel robotic platform, twelve inches long and thirteen inches wide, with a maximum battery life of 2.5 hours and a differential wheel motor controller. For our application, we have added two crossed infrared sensors in a bumper configuration on the front of the robot to permit obstacle detection and avoidance, one infrared sensor on each side of the robot to permit wall following, and the Stargazer localization device which can localize the robot's position within two centimeters. Other devices which appear on the hardware version of the robot – such as the webcam and optional robotic arm – are unused by any of the tested algorithms. This robot was chosen as a cheaper alternative for indoors experimentation to the more expensive Explorer robot, which is a more capable robot manufactured by the same company used for outdoors applications with stronger motors and a suspension system. The robots share a common API and thus, for experimentation purposes and potential future transfer of this work to hardware, are nearly identical

apart from the modelled hardware and capabilities.

In order to show that our algorithm can function in a realistic environment, accuracy of the simulated model was a priority. This was accomplished by ensuring that complications that would affect a physical robot were modelled in the simulated environment. The rolling friction of the robot's wheels, the weight of the robot, the reflectivity of objects in the environment and the noise in the Stargazer module's results were all taken into account. The components with the most complicated behavior, however, were the two Sharp GP2Y0A21YK 80cm sensors mounted on the front of the robot for obstacle avoidance, and the two Sharp GP2Y0A02YK0F 150cm sensors mounted on the sides of the robot for wall following. Both of these sensors are analog infrared range-finders that work by measuring the angle at which emitted IR radiation reflecting off of an obstacle impacts the IR receiver on the sensor package, and both have a non-linear response curve which report inaccurate results within a minimum range, as shown in Figure 5.3 (Sharp Corporation, 2005).

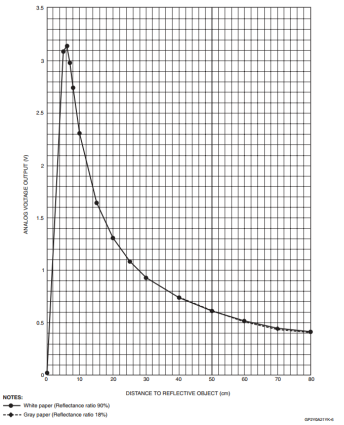


Figure 5.3: Response of Sharp GP2Y0A21YK sensor, showing output voltage vs. distance.

The algorithm itself was implemented in C++ and compiled by the MinGW (Minimalist GNU for Windows) compiler bundled with Webots and the Webots robotics libraries. The controller is 7126 lines of code, split up into the main controller loop and methods at 2145 lines, the STC sub-controller at 2079 lines, the $min - \varepsilon$ algorithm at 604 lines, plus additional utility code: 1387 lines of computational geometry objects and algorithms, 308 lines of code to interface the controller with the Webots libraries, and 603 lines of miscellaneous code. Table 5.1 shows the different parameters and their values that were used to test our proposed algorithm.

Table 5.1: Configurable parameters within the algorithm and values used during experimentation.

Experiment Main Controller Parameters			
Parameter Name	Description	Units	Value
INSIDE_POLY_PERMISSIBLE _ERROR	Number of degrees of error allowed in the winding angle algorithm's return value at which the algorithm will assume the point is inside the polygon.	degrees	42.0
NEXT_VERTEX_IF_CLOSER _THAN	Distance from a vertex at which the robot controller assumes the robot is "at" the vertex in question.	meters	0.21
TURN_IF_DEVIATION _GREATER_THAN	Angle offset at which a robot following an edge will stop and turn back towards the edge.	degrees	8
PLOT_IF_DEVIATION _GREATER_THAN	Angle offset from previous coverage history data-point plot at which the robot controller stores another data-point.	degrees	12
PLOT_ON_TURN_ALWAYS	Whether or not the robot plots a point every time the robot executes a turn.	boolean	FALSE
EDGE_LENGTH_IN_METERS	Length of the edges of the individual regular polygons of a region.	2.0	2.0
NUM_SIDES	Number of sides in the individual regular polygons of a region.	4	4
Experiment STC Controller Parameters			
Parameter Name	Description	Units	Value
angle_threshold	As per TURN_ERROR, but for the STC portion of the algorithm.	degrees	8
edge_threshold	Distance from an edge at which the robot is considered to be "at" that edge.	meters	0.15
cell_distance	Size of an individual STC cell.	meters	0.45
width_of_robot	Size of the robot in its largest dimension.	centimeters	24.5

5.2 Simulation Results

The system was tested with simulated Corobots in a 20 meter by 20 meter square arena, bounded on each side by a wall. Four different arena environments were tested: an empty arena, an arena in which 10% of the area within was occupied with various obstacles, an arena in which 25% of the area within was occupied with various obstacles, and an arena which simulated two rooms separated by a narrow hallway. Each environment was covered with a two robot team, a three robot team, and a four robot team for a total of twelve environment/team-size pairings, described in Figure 5.2. Each pairing was simulated ten times at two hours each; two hours was chosen as an arbitrary period of time to gauge the performance of the algorithm during a known period of time. All results were averaged over the ten simulation runs, and the results have been reported after removing runs in which the entire system failed (those runs where total system coverage was equal to zero); this is to remove bias and error introduced by issues related to inaccuracies in the physical simulation and code faults.

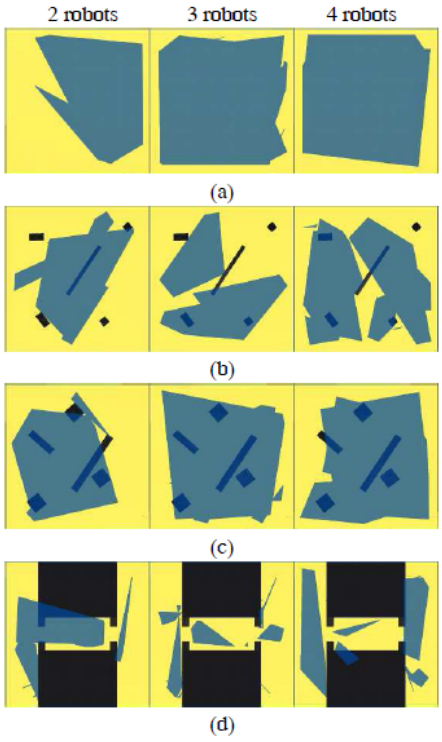


Figure 5.4: Coverage area of arenas that are: (a) empty; (b) 10% obstructed; (c) 25% obstructed; (d) corridor-like.

Identical sets of simulations were run using two comparison algorithms. The first such algorithm is the PAC algorithm, but with all ability to communicate disabled; this is to illustrate the performance gain or loss which is introduced by the small amount of com-

munication within the multi-robot team. The second such algorithm is a standard coverage approach in which the environment is partitioned into a Voronoi field based on each robot's initial position and each partition is covered by a single robot using the STC algorithm.

Table 5.2: Description of the environments used in experimentation.

Scenario Name	Arena Area	Obstacles	Number of Robots	Run Time
0%, 2 robots 0%, 3 robots 0%, 4 robots	400 sq. meters	None	2 robots 3 robots 4 robots	2 hours
10%, 2 robots 10%, 3 robots 10%, 4 robots		Objects, 10% of total arena area	2 robots 3 robots 4 robots	
25%, 2 robots 25%, 3 robots 25%, 4 robots		Objects, 25% of total arena area	2 robots 3 robots 4 robots	
Corridor, 2 robots Corridor, 3 robots Corridor, 4 robots		Corridor and rooms	2 robots 3 robots 4 robots	

The metrics tested via simulation were: the total amount of area covered by the multi-robot team, the total distance travelled by the team while performing coverage, the total distance travelled by the team while travelling between coverage areas, the total number of regions communicated between the robots within the team, and the total number of times region coverage was aborted due to a robot entering a region which was already previously covered. Only communication-enabled PAC reports all metrics; PAC without communication does not report a communication metric, and the Voronoi/STC algorithm only reports the area of all completely covered cells.

Area Covered

The first metric to examine is, perhaps, the most obvious: how much area was covered by the multi-robot teams in two hours of simulation? By examining this metric, we can get an immediate picture of the effectiveness of the algorithm as a whole.

Figure 5.5 shows quite clearly that in the case of an empty, unobstructed arena, the coverage algorithm performs very well: on average, a three robot team covers the entirety of the arena, and a four robot team covers the area fully and then some (the arena is 400 square meters in area). Coverage in excess of 400 square meters is due to repeated coverage.

In comparison, the reference algorithms perform much more poorly in this environment. Non-communicative PAC shows uniformly inferior performance to standard PAC regardless of the number of robots, and Voronoi/STC coverage is unable to complete a full unwind of many of the STC cells before the simulation completes; however, Voronoi/STC coverage is much more consistent.

In the 10% and 25% environments coverage does not approach completion, showing that the environment is more difficult. However, moving from two to three robots more than doubles the performance of the system as a whole, and moving from three to four robots doubles performance once again, showing a super-linear improvement in system performance. The super-linear increase in performance between a two and three robot system in both the unobstructed arena and in the twenty-five percent obstructed area, and the super-linear increase in performance between the three and four robot systems in the ten percent obstructed arena, suggests that the robots are working together to maximize efficiency in coverage.

Comparison to non-communicative PAC shows inferior performance in the three and four robot cases in the 10% obstructed environment, and widely varying performance in the two robot case. Variations in performance are likely due to variable amounts of conflict and overlap each run due to randomized starting locations. Two robot Voronoi/STC displays near-zero performance, while performance in the three and four robot cases is

superior to both other algorithms on average; however, standard PAC occasionally outperforms Voronoi/STC as evidenced by the displayed deviation. In the 25% obstructed case, Non-communicative PAC outperforms standard PAC in the two robot system, but under-performs by a wide margin in the three and four robot systems. Voronoi/STC consistently under-performs in the two and three robot systems, but has higher average performance in the four robot system; however, as before, standard PAC occasionally outperforms Voronoi/STC.

The corridor environment is obviously the most challenging environment for the robot teams, and also has the least amount of free area. The two, three, and four robot teams seem to all perform poorly in this environment. Not much can be determined from this metric alone; however, it's likely that the other metrics will offer more insight into the reasons behind the failure of the system in this environment. By comparison, non-communicative PAC is similarly non-performant, Voronoi/STC only accomplishes coverage in three and four robot systems, with only marginal performance.

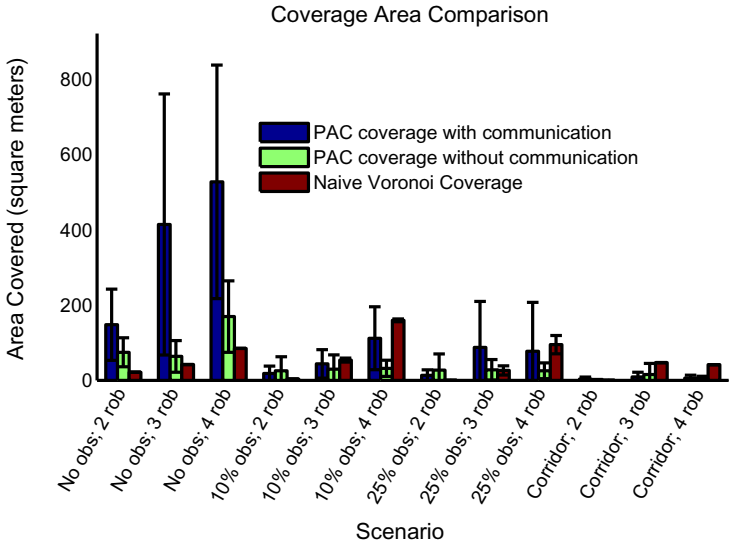


Figure 5.5: Average area covered by the team using each algorithm, by scenario.

Distance Travelled

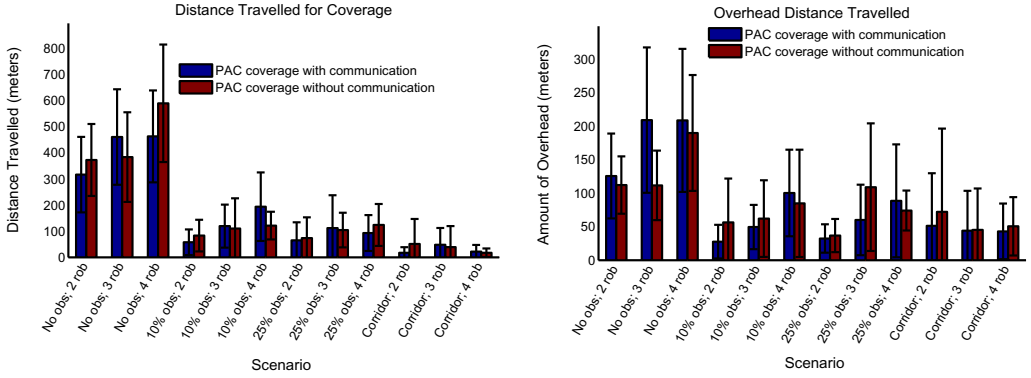
The metrics for distance travelled allow examination of the efficiency of the algorithm. Distance travelled is moderately related to time spent, as the robots move quickly during coverage and between regions, more slowly during wall-following, and very slowly during obstacle avoidance.

Robot motion for the unobstructed arenas using standard PAC is high, as is to be expected. However, the three and four robot teams travelled almost the exact same distance, yet the four robot team accomplished more coverage. This implies two things: first, communication between robots allowed the robots to recognize regions covered by others and subsequent dispersion from those areas; second, the four robot team had more overlapping coverage due to more simultaneous coverage (coverage of an area is only transmitted to the team at large once the region's coverage is finished). Non-communicative PAC shows a similar amount of useful travel but for a much lower amount of resulting coverage; this seems to indicate a large amount of repeated coverage; this is supported by lower overhead (dispersion) distances travelled, showing that less effort to avoid re-covering previously covered areas was undertaken.

Distance travelled by standard PAC is lower for the rest of the environments, indicating a slower pace due to obstacle avoidance. For the ten and twenty-five percent environments, the coverage/overhead ratio was over 1.0; more distance was travelled during coverage than while transiting between regions. The corridor environment's efficiency ratio is close to or under 1.0, showing that the environment was very difficult for the teams.

Non-communicative PAC shows fairly flat useful travel distance for the system in both the 10% and 25% obstructed arenas for all numbers of robots, which corresponds strongly to the fairly flat (and low) performance in terms of area coverage over those same cases.

Useful distance travelled in the corridor environment for non-communicative PAC is extremely variable; combined with the lack of performance in terms of area covered, the algorithm seems to be unable to cope with the challenging environment. Overhead/dispersion distance travelled is widely variable in almost every obstacle-containing area for non-communicative PAC aside from the 4 robot system in the 25% environment. Variability is much higher than for standard PAC in all these cases, indicating that the uncoordinated nature of the robots' behavior leads to widely varying system efficiency. The reason for more uniform overhead in the case of the 4 robot system in the 25% environment is unclear, but does not seem to have an effect on useful distance travelled nor system performance.



(a) Distance travelled during coverage. (b) Distance travelled between regions.

Figure 5.6: Total distance travelled by the robot team, by task, in each scenario.

Communication and Aborted Blocks

These metrics are both related to end-of-region activities. Communication occurs at the end of every successful region coverage, while aborted region coverage only occurs in the case of a robot stumbling into a previously covered area during coverage.

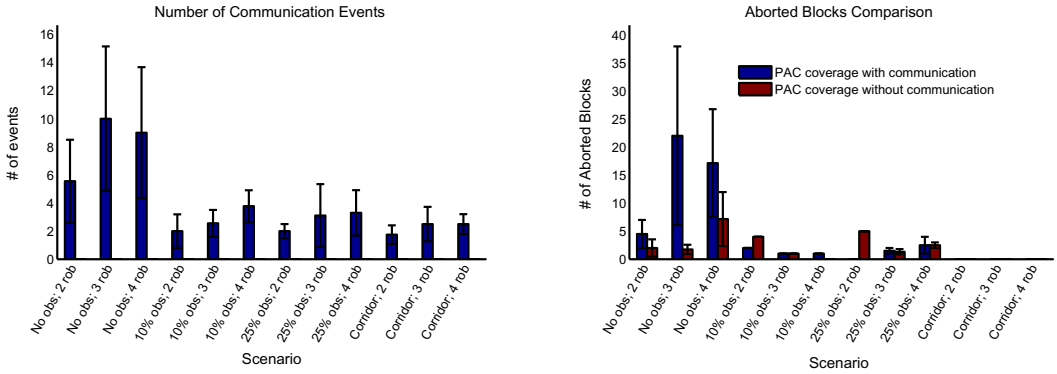
In the unobstructed environments, the number of successfully completed regions is greater than in the other environments and, in the three and four robot teams, the number of aborted block definitions is very high. This indicates a very high degree of arena coverage and at-

tempted overlapping coverage as the simulation progressed. The low number of aborted block definitions in the two-robot case indicates that the dispersion algorithm was successful in avoiding an undue amount of repeated coverage.

In the partially obstructed environments, aborted region coverage instances were all but nonexistent, indicating a reduced amount of coverage and success in dispersion. The number of communication instances shows that adding more robots resulted in a fairly linear increase in the amount of successfully covered regions in the ten and twenty-five percent environments, while the corridor environment showed a sub-linear increase in successful coverage – again, most likely due to the challenging nature of the environment.

Non-communicative PAC cannot be used to compare the communication metrics; however, a comparison of aborted blocks immediately demonstrates that in the environment most prone to repeated coverage – the unobstructed arena – the number of aborted blocks is extremely low. This seems to indicate that the lack of communication between robots within the system results in a large amount of repeated coverage of the same area, as less effort is expended in attempts to avoid it. This supports the same indications seen in other captured metrics.

Analysis of the experimental results shows that the algorithm is very successful in unobstructed environments analogous to fields, plains, deserts, and other agricultural, rural, and some military environments. It is shown that communication has a distinct positive, superlinear impact on coverage in these and lightly obstructed environments, showing that the algorithm enables successful teamwork with a very limited amount of shared data between the agents in the system. This is supported by comparison to the performance of the same algorithm without communication; the difference in performance resulting from even the most token amount of shared coverage information is striking. Similarly, a comparison



(a) Region transmissions during the standard PAC algorithm.

(b) Aborted region coverage attempts during each PAC algorithm.

Figure 5.7: Average number of different events by algorithm and scenario.

to naive Voronoi/STC coverage shows that in time-sensitive situations, coverage in unobstructed environments is more complete.

Chapter 6

Future Work and Conclusions

In this chapter, we discuss the challenges encountered while designing, implementing, and analyzing the PAC algorithm as well as discuss what additional work can be performed to refine the algorithm and increase its performance. We then go on to summarize the work and our results.

6.1 Lessons Learned

While the theory behind the PAC algorithm is robust, implementation challenges have had a considerable impact on the performance of the algorithm and the system utilizing it. While many projects of this type use an abstract and highly discrete model for robot behavior, we selected a simulation suite which performs accurate and near-continuous physics simulation and requires the robots to interact with their environment using nothing but their simulated sensors and actuators. The robots were given no *a priori* knowledge of the environment.

Our algorithm was required to be robust when encountering sensor noise modelled as part of the simulation, collisions with obstacles, loss of wheel traction, and other issues which can and did result in individual robots being placed in unrecoverable situations where they could no longer maneuver in or interact with the environment. Such issues greatly compli-

cated the implementation of a robot controller which reliably accomplishes its goals. Robot failures due to environmental hazards was an ongoing issue throughout the implementation process and the many failure modes of the system impacted the experimental results of the simulations.

6.2 Future Work

Given the effect of simulated failures on the system, the largest gains in algorithm performance and efficiency would be earned by further hardening of the algorithm against adverse environmental conditions. Such work would take the form of more reliable obstacle handling and avoidance techniques, cataloging obstacles and using that knowledge during the dispersion phase of the algorithm, and altering variables such as region combination fitness parameters, the number of vertices in polygonal approximations, and finding the highest safe limits to wheel speed of the robots in various controller states.

Future alterations to the algorithm's design might include: determining a termination condition for the algorithm; optimizing the amount of erroneous map gain and map loss for different algorithm applications; and transmitting robot position at the beginning of a region coverage phase for use in other robots' dispersion behavior to limit region overlap.

Further analysis and experiments might include: introducing limits to the number of regions which can be stored by the robots at a given time in order to model robots with extremely limited storage capabilities; comparing performance to a version of PAC that transmits its entire cellular coverage map; and determining at what point the algorithm begins to perform either excessive repeat coverage or is unable to complete a polygonal region segment without some number of aborts occurring in a row.

6.3 Summary

We presented a novel algorithm for multi-robot area coverage which focuses on minimizing the amount of coverage data transmitted between robots during the coverage process. The body of existing work related to this problem was presented, including work concerning mobile robots, multi-robot coverage, and map information compression. Having established that this problem has not been addressed in the past, we described the coverage algorithm and the polygonal compression algorithm, which is the main body of the work.

The results of our simulated experiments were shown and discussed in the context of four different environments, covered by teams of two, three, and four robots. After analyzing the area coverage, distance travelled during and outside of coverage, and the amount of communication and coverage interruptions due to overlapping coverage areas both of the proposed algorithm and a version of that same algorithm lacking communication, as well as a reference algorithm, it was determined that the algorithm was demonstrated to be effective even with extremely limited data transfer between the robots within the teams and that the implementation could be improved in order to show that effectiveness in more complex, challenging environments.

Bibliography

Baek, S., de Veciana, G., and Su, X. (2004). Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchal aggregation. *IEEE Journal on Selected Areas in Communications*, 22(6):1130–1140.

Batalin, M. and Sukhatme, G. (2002). Spreading out: A local approach to multi-robot coverage. In *in Proc. of 6th International Symposium on Distributed Autonomous Robotic Systems*, pages 373–382.

Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S. (2005). Collaborative multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386.

Butler, Z. (2000). *Distributed coverage of rectilinear environments*. PhD thesis, Carnegie Mellon University.

Gabriely, Y. and Rimon, E. (2003). Competitive on-line coverage of grid environments by a mobile robot. *Computational Geometry*, 24:197–224.

Hazon, N. and Kaminka, G. (2008). On redundancy, efficiency, and robustness in coverage for multiple robots. *Robotics and Autonomous Systems*, 56(12):1102–1114.

Howard, A., Mataric, M., and Sukhatme, G. (2002). Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *in Proc. of 6th International Symposium on Distributed Autonomous Robotic Systems*, pages 299–308.

Koenig, S., Szymanski, B., and Liu, Y. (2001). Efficient and inefficient ant coverage methods. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):41–76.

Parker, L. (1999). Adaptive heterogeneous multi-robot teams. *Neurocomputing*, 28(1-3):75–92.

Perez, J. and Vidal, E. (1994). Optimum polygonal approximation of digitized curves. *Pattern recognition letters*, 15(8):743–750.

Reid, J. F., Zhang, Q., Noguchi, N., and Dickson, M. (2000). Agricultural automatic guidance research in north america. *Computers and Electronics in Agriculture*, 25(1-2):155 – 167.

Rekleitis, I., New, A., Rankin, E., and Choset, H. (2008). Efficient boustrophedon multi-robot coverage: An algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):109–142.

Rutishauser, S., Correll, N., and Martinoli, A. (2009). Collaborative coverage using a swarm of networked miniature robots. *Robotics and Auton Systems*, 57(5):517–525.

Sharp Corporation (2005). *Sharp GP2Y0A21YK Optoelectronic Device*. 22-22 Nagaike-Cho, Abeno-Ku, Osaka 545-8522, Japan.

Shekhar, S., Huang, Y., Djugash, J., and Zhou, C. (2002). Vector map compression: a clustering approach. In *Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 74–80.

Stachniss, C., Mozos, O., and Burgard, W. (2008). Efficient exploration of unknown indoor environments using a team of mobile robots. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):205–227.

Wurm, K., Stachniss, C., and Burgard, W. (2008). Coordinated multi-robot exploration

using a segmentation of the environment. In *in Proc. of International Conference on Intelligent Robots and Systems*, pages 1160–1165.