Variable Detail Robotic Sensor Coverage

Kevin Bradner

Department of Electrical Engineering and Computer Science

Case Western Reserve University

January, 2020

Contents

List of Tables

List of Figures

Acknowledgements

Abstract

Robotic coverage problems task one or more robots with the goal of visiting every location in a region. Algorithms which can perform this kind of task in a time efficient manner are useful for purposes such as mapping, cleaning, or inspection. This work considers a multi agent robotic coverage problem in which the shape of the region to be explored is known in advance, but additional information and challenges are discovered during task performance. A software package is created to simulate such a scenario, generate virtual environments to be covered, and interface with policy programs that command the robots. Offline path planning algorithms are developed, and their performance on this task is evaluated. Next, online variants of these algorithms are developed to respond to events and information encountered during task execution. It is shown that the online algorithms are more robust and better performing than their offline counterparts.

Chapter 1

Introduction

1.1 Robotic Coverage Background

Robotic motion planning problems are typically concerned with finding an achievable path between two robot configurations. These configurations could be the current location and desired destination of a robotic vehicle, the current pose of a robotic arm and the pose required for it to grasp an object, or any other pair of robot states which correspond to "start" and "goal" configurations. *Robotic coverage* problems are another kind of motion planning problem. The goal of a robotic coverage problem is to find a path for the robot to follow, during which it will visit every location in a region. This region could be an area of floor to be vacuumed, the surface of a car body that needs to be painted, or the interior of a building that needs to be mapped. In all of these cases, the task is complete only once the entire region has been visited, or *covered* by a robot. Usually, another goal of these problems is to achieve coverage in a way that minimizes the time or number of movements required to complete the task. The robotic coverage problem can be viewed as a continuous generalization of the travelling salesman problem [7]. Thus, the problem is NP-Hard.

One particularly well studied type of robotic coverage problem is the case where the coverage region is some subset of the plane, and the robots are mobile robots that can only move in the plane. A good overview of this kind of coverage problem can be found in [1]. Algorithms to perform robot coverage in the plane have been studied since the late 1980's. Early algorithms for this purpose relied on the use of heuristics, and these algorithms usually could not guarantee that complete coverage would always be achieved. By around 2000, planar coverage algorithms were often able to guarantee complete coverage. A completeness guarantee is often proved by showing that the algorithm breaks down the coverage space into smaller regions that are relatively trivial to cover completely. Coverage algorithms that employ this kind of technique are known as *cellular decomposition* algorithms.

One type of cellular decomposition fits a square grid to the coverage space. Because arbitrary sets in the plane can not be exactly decomposed into square regions in a general case, this technique is often called approximate cellular decomposition. Typically, each cell in this kind of decomposition can be covered entirely by a single robot configuration. For example, consider dividing a grass lawn into squares for a robotic lawn mowing task. If each square in the cellular decomposition is small enough, a lawnmower whose center is aligned with the square's center will have mowed that entire square of the lawn. In coverage problems where this assumption applies, proving that a coverage path visits each cell in the approximate decomposition may be enough to guarantee complete coverage.

In addition to approximate cellular decomposition, there are also techniques that decompose a coverage space into a more diverse collection of cell shapes. Examples of such shapes include cells of fixed with whose top and bottom boundaries can vary in shape [3], or trapezoidal cells whose parallel sides all run in the same direction [4]. Depending on whether the union of these cells is exactly equal to the coverage space, such techniques are called either *semi-approximate* or *exact*

cell decompositions. In the case of trapezoidal cell decomposition, it is possible to combine the trapezoidal cells into a smaller number of large cells, each of which are able to be covered by a simple back-and-forth sweeping motion. This approach is known as a *boustrophedon* decomposition [4].

1.2 Spanning-Tree based Coverage Algorithms

When coverage is performed by a square shaped tool attached to a mobile robot, and when the coverage area can be exactly decomposed into a grid of squares which have double the side length of the robot tool, it is possible to compute an optimal covering path in linear time using an approach based on spanning trees [2]. The authors developed several variants of this approach with different time, memory, and prior knowledge requirements. In all cases, it was assumed that a mobile robot would complete the coverage task with a square shaped tool of size D. It was also assumed that the tool could only move in the four cardinal directions when completing the task. Finally, it was assumed that the region to be covered could approximated by cells in a grid of squares with size $2D$. The authors argue that the final assumption is justified when the size of the robots tool is significantly smaller than the dimensions of the area to be covered. Under these assumptions, the *Spanning Tree Covering* (STC) algorithm is able to generate a path through the space which visits each grid-aligned square cell of size D exactly once.

The offline variant of STC works by representing the coverage area as a graph G. Nodes on this graph represent the center of a square with size $2D$, and the set of nodes corresponds to the set of full cells when a grid with square size $2D$ is overlaid on the coverage space. Edges exist between any two nodes whose corresponding cells are adjacent in the coverage space. As the name suggests, STC's key step is the creation of a spanning tree on G, using any node in G as the root of the tree.

Many algorithms exist to compute spanning trees, but two popular examples are closely based on breadth-first-search and depth-first-search [9].

Once this spanning tree has been created, the coverage area is subdivided into squares of size D, such that each node of G coincides with the corners of four different squares in the subdivided map. Starting at any one of these squares, each of which is exactly the shape and size of the robot's coverage tool, the robot can simply move counterclockwise around the spanning tree as if it were performing wall following. In this way, each cell in the coverage space will be visited exactly once by the robot. Because of this property along with the fact that the path generated by STC end up adjacent to where it started, the path found by STC can be considered a Hamiltonian cycle on G. Although constructing a Hamiltonian path on a general planar grid is NP-complete [10], the assumed properties of G make it practical to compute such a path on this graph in linear time. It is also worth noting that this coverage path's start and end positions will be in adjacent cells. This can be beneficial in practice, as it allows deployment and retrieval of the robot to happen in the same place.

This algorithm has a few interesting variants. The first runs online, and effectively circumnavigates a depth first search spanning tree that it constructs on the fly [2]. Restriction to use of depth first search is the primary disadvantage of this approach compared to offline STC. While the coverage paths based on depth first search are equivalent to those created through any other method in terms of final path length, other spanning tree algorithms have practical benefits. For example, spanning tree algorithms such as Prim's algorithm or Kruskal's algorithm can create minimal spanning trees on weighted graphs, and this can be exploted to create coverage paths that minimze turning. In later work, the authors of STC do exactly that in an algorithm called Scan-STC, which also makes additional improvements on the original STC as discussed below. For details on Prim's algorithm and Kruskal's

algorithm, refer to [9].

The second online STC algorithms is described by its authors as *ant-like*. In this case, the robot must be able to mark visited sub-cells as visited through some means such as leaving pebbles on the ground. In addition, the robot must be able to sense the presence of these makers in the nodes of G adjacent to its current position. By marking each visited sub cell as the robot leaves that cell, the robot is able to proceed on the same path it would have created in the depth first search based online approach. Because this algorithm uses markers to identify visited nodes, it requires only O(1) memory. However, it requires the use of O(N) markers, where N is the number of sub-cells in the coverage area. Nonetheless, this algorithm presents interesting ideas for multi-agent approaches and prevention of drift in pose estimation.

Finally, there is a variant of STC that achieves *exact* coverage of an environment, even when that environment can not be represented by completely filled cells on a grid of squares with size $2D$. However, this approach still requires that the connectivity of the environment is represented adequately by such a grid. The resulting algorithm simply notices any cells that are partially occupied by a previously unseen obstacle when it passes next to them in the course of normal online STC. When such a cell is encountered, a neighborhood is swept around the entire obstacle. Then, the robot returns to traversing the fully in-bounds cells according to its usual procedure. For details concerning the size of this neighborhood and the correctness of the resulting algorithm, see [2].

In the previously mentioned Scan-STC algorithm, STC is improved to explicitly handle boundary cells, or graph cells that are partially occupied by an obstacle. Using the same notation as before, this algorithm generates a path with length no greater than $(n + m)D$, where n is the number of nodes in G, and m is the number of boundary cells [8]. The authors assume that no node will contain an arrangement

of obscales such that the graph of that node's subnodes is disconnected. If an obstacle exists such that the normal grid of nodes would violate this assumption, the corresponding location in the coverage area is assigned two different nodes. With that assumption in place, avoidance of obstacles is done by simply displacing the normal path that goes around the obstacle. This displacement tactic is what leads to the m term appearing in the maximum path length bound. The authors claim that in practical environments, the path length is closer to $n + \frac{m}{2}$.

The other significant insight of Scan-STC relates to its name. By allowing the mobile robot to sense the state of the eight cells the surround its current position, it is possible to discover that horizontal neighbors of the current cell has a free vertical neighbor, and that the current cell has a free vertical neighbor in the same direction. In this case, it is possible to skip the creation of a horizontal edge between the current node and its horizontal neighbor. The benefit of this approach is that it leads to coverage paths that move primarily in the vertical directions.

1.3 Muti-Agent Robotic Coverage

Another imporant feature of a robotic coverage algorithm is whether that algorithm controls a single robot or a group. Algorithms that control a group of robots for this purpose are known as multi-agent coverage algorithms. Multi agent approaches to coverage can often complete a coverage task several times faster than a single agent approach because of the division of work that use of multiple robots enables. Use of multiple robots can enhance robotic coverage by allowing for more precise localization through information sharing. Finally, multi-agent methods are often able to adapt to the failure of one or more robots, as there may be remaining robots capable of completing the coverage task [1]. However, multi-agent techniques often require more complex motion planning strategies to coordinate the motion of

multiple robots in a way that takes full advantage of these potential efficiency improvements and other benefits.

It is also possible to adapt a spanning tree coverage algorithm to the multi-agent case [6].

1.4 Problem Statement

This work presents multi-agent coverage algorithms based on spanning trees.

Chapter 2

Simulation Development

Chapter 3

Offline Planning Policies

Chapter 4

Online Planning Policies

Chapter 5

Conclusion

Chapter 6

Bibliography

[1] H. Choset, "Coverage for robotics - A survey of recent results," *Annals of Mathematics and Artificial Intelligence,* vol. 31, October 2001.

[2] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of Mathematics and Artificial Intelligence,* vol. 31, October 2001.

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

[4] H. Choset, E. Acar, A. Rizzi, and J. Luntz, "Exact Cellular Decomposition in Terms of Critical Points of Morse Functions," *Proceedings of the 2000 IEEE International Conference on Robotics and Automation,* April 2000.

[5] D. Kurabayashi, J. Ota, T. Arai, S. Ichikawa, S. Koga, H. Asama, and I. Endo, "Cooperative Sweeping by Multiple Mobile Robots with Relocating Portable Obstacles," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* November 1996

[6] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-Robot Forest Coverage," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* August 2005

[7] E. Arkin, S. Fekete, and J. Mitchell, "The lawnmower problem," *Proceedings of the 5th Canadian Conference on Computational Geometry,* August 1993

[8] Y. Gabriely and E. Rimon, "On-Line Coverage of Grid Environments by a Mobile Robot," *Computational Geometry,* April 2003

[9] T. Cormen, C. Lieserson, R. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition,* MIT Press and McGraw-Hill, 2001

[10] A. Itai, C. Papadimitrou, and J. Szwarcfiter, "Hamilton Paths in Grid Graphs," *SIAM Journal on Computing,* November 1982