A Practical Robot Coverage Algorithm for Unknown Environments*

Heung Seok Jeon[1], Myeong-Cheol Ko[1], Ryumduck Oh[2], and Hyun Kyu Kang[1,]**

[1] Department of Computer Science, Konkuk University, Chungju City, Korea
{hsjeon,cheol,hkkang}@kku.ac.kr
[2] Department of Computer Science, Chungju National University, Chungju City, Korea
rdoh@hanmail.net

Abstract. While there has been substantial research on coverage and SLAM algorithms, to our knowledge no previous research has considered using each of these algorithms with a service robot. As a result, the performance of these robots is less than adequate for most consumers, especially when the robots only rely on SLAM algorithms for their coverage services in an unknown environment. To address this problem, we propose a new coverage algorithm, AmaxCoverage, an area-maximizing coverage algorithm that efficiently integrates the SLAM solution. Our experimental results show that the AmaxCoverage algorithm outperforms previous representative coverage algorithms in unknown environments and therefore will increase consumers' confidence toward service robots.

Keywords: Practical, robot, coverage, SLAM, real-time.

1 Introduction

Coverage algorithms are one of the core technologies required for intelligent robots in domains such as cleaning, harvesting, painting, and lawn mowing. These algorithms determine the path a robot will take to ensure complete coverage of an unknown target area, i.e., the robot will cover every part of this area at least once. For example, a cleaning robot will follow a path planned by its coverage algorithm to clean the target area and guarantee that it does not miss a spot.

Coverage algorithms based on randomness, e.g., turn right whenever an obstacle is encountered, are widely used in commercial cleaning robots since they require very little overhead to implement. While these algorithms are simple to implement, they have one major flaw: they are not efficient. For example, as the area to be cleaned increases, a robot utilizing a random algorithm requires more time to finish the task and there is no guarantee that the area will be completely covered. For the consumer, the result of this poor performance is dissatisfaction with the product and possibly the idea of intelligent robots.

* This work was supported by Konkuk University in 2010.

** Corresponding author.

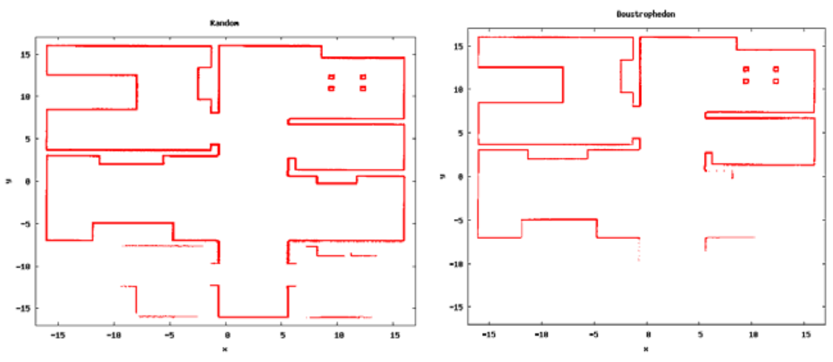
To overcome the problems associated with random algorithms, several smart coverage algorithms have been proposed [1]-[8]. The seminal work of Zelinsky et al. [1] is regarded as the cornerstone of these approaches. They proposed a complete coverage method that uses a distance transform to plan the coverage path. While this approach is more robust than relying on a random algorithm, it has limitations since it requires a complete gridded map of the target environment *before* the start of the robot's task.

Choset and Pignon extended the work of Zelinsky et al. so that a non-gridded map could be used [2]. The Boustrophedon Path algorithm they proposed is one of the most popular and basic coverage algorithms for recently commercialized cleaning robots. Using this algorithm, the robot crosses the full length of the target area in a straight line, turns around at the boundary, and then traces a new straight-line path adjacent to the previous one, similar to how an ox would plow a field, hence the phrase Boustrophedon Path. By repeating this procedure, the robot will cover the target area when given enough time [2].

However, this approach has limitations. While a robot using the Boustrophedon Path algorithm shows very good performance for an area that contains no obstacles, if the area has obstacles, the performance is dramatically reduced. For example, in a real environment, i.e., non-simulated environment, obtaining a map for a particular area of interest prior to the actual task is not trivial because the layout of each area may be unique and include obstacles that are static, e.g., a chair or other dynamic, e.g., a person. To overcome this difficulty and increase its performance, a robot using the Boustrophedon Path algorithm requires a map and the ability to localize itself within the map to reduce duplicated visits to an area and to plan the most efficient path.

The construction of a map for the target area and the ability for the robot to know its location at any point in time necessitate that the coverage algorithm be integrated with one of the solutions for the Simultaneous Localization and Mapping (SLAM) problem. This problem asks whether it is possible for a robot to incrementally build a complete map of an unknown environment while simultaneously determining its location within this map [13]. The SLAM problem has been a major issue for the mobile robotics community because its solution is to make a robot truly autonomous [13]. While EKF-SLAM [14] and FastSLAM [15][16] are the two most influential solutions, newer potential alternatives have been proposed. The SLAM problem is considered solved in robotics literature [15]. As a result, coverage algorithms should enable integration with a SLAM algorithm to efficiently cover unknown environments.

If coverage and SLAM algorithms are not integrated appropriately, however, the overall performance of the robot could decrease (Figure 1). These maps were constructed from an unknown environment after 30 minutes of coverage using the Random and Boustrophedon algorithms integrated with the FastSLAM algorithm. The results are interesting since the Boustrophedon algorithm, known as the one of the best-performing algorithms for unknown environments, shows a worse performance than the "simple-minded" random algorithm. This is because, in a complex environment, the Boustrophedon algorithm has an increased number of visits near obstacles. To address this issue, we propose a new coverage algorithm, AmaxCoverage, which we describe in the next section. In Section 3, we present the implementation details of the AmaxCoverage algorithm, followed by the performance evaluation results in Section 4. In Section 5, we present our conclusions and mention areas for future work.



(a) Random Algorithm (b) Boustrophedon Path Algorithm

Fig. 1. Maps Constructed after 30 Minutes of Covering

2 The AmaxCoverage Algorithm

The AmaxCoverage algorithm is based on the assumption that a robot does not have *a priori* knowledge regarding the map of the environment it has to cover. Without a map, a robot cannot plan the path it needs to take to guarantee complete coverage. Our approach for this kind of situation is to have the robot optimize its coverage within a given time. Hence, the goal of the AmaxCoverage algorithm is not to minimize the total coverage time but to maximize the coverage efficiency in real time, i.e., cover the most area in the shortest time. To do this, the algorithm finds and covers the largest and most obstacle-free area. This means that the AmaxCoverage algorithm places a higher priority on large and empty areas with few obstacles than to small and complex areas.

Compared with the algorithms mentioned, the AmaxCoverage algorithm provides better results when the total time for the task is not long enough to allow the robot to cover the target area. In this situation, the robot assigns a higher priority to covering large obstacle-free areas. We believe that this is a plausible assumption. Consider the following possible scenarios: 1) you are expecting a visitor to arrive sooner than the total time required for the cleaning task to be completed, or 2) the remaining battery charge of the robot is not sufficient to cover the target area and recharging requires more time (therefore, adding to the total time for the task). In each of these cases, having the robot maximize the coverage area will result in increased efficiency.

Another benefit of the AmaxCoverage algorithm is that a robot that uses this algorithm can efficiently cooperate with a human counterpart. In real life, we cannot always expect a robot to perfectly complete its cleaning task all by itself without any assistance from a person. The interior layouts of most houses have atypical and dynamic characteristics. The structures of residences vary in size and shape. Moreover, the arrangement of tables, sofas, and other obstacles are not fixed, so this increases the complexity of the target area. Usually, the corners of rooms, areas beneath low obstacles, and the tops of obstacles need manual cleaning to achieve satisfactory quality [12].

For these reasons, it is more practical for people to cooperate rather than fully rely on the cleaning performance of a robot. For example, a robot can initially clean a large and mostly empty portion of a room that has few obstacles. People then complement the task by focusing on the complex parts of the room. This process may consume the same time for the robots to finish the cleaning job, but with assistance, the actual time required for cleaning will be reduced. Therefore, to increase the efficiency of cleaning robots with the help of people, robots can start the cleaning process in less complex areas first. This is another reason why the AmaxCoverage algorithm puts priority on large empty areas with few obstacles rather than on complex areas.

3 Implementation

As stated, the AmaxCoverage algorithm works under the assumption that the robot has information about the area to be covered and that new information and a map will be provided while the robot covers that area. If the robot encounters a new obstacle in the environment during the coverage action, the map will have a real-time update to include the new information.

This process performed by the AmaxCoverage algorithm as follows. First, it computes the Minimum Bounding Rectangle (MBR) in the scope of the currently available map. The MBR is the minimal, i.e., smallest, rectangle that includes all the areas that are free of obstacles (free areas), areas that contain obstacles (non-free areas), and the walls of the map (boundaries). The MBR is a current snapshot of the map that can dynamically change over time. A robot using the AmaxCoverage algorithm will always want to perform its task on free areas first; therefore, these areas will become the target areas for coverage. Second, if the robot finds new information that is outside the MBR while it is covering the free area, the MBR will be updated. The strategy used for changing the MBR will be discussed at the end of this section.

As described in Section 1, the Boustrophedon Path algorithm is one of the most efficient solutions when there are no obstacles in the target area. However, when obstacles are in an environment, it is more difficult to plan a Boustrophedon Path that will have complete coverage. In non-simulated environments, an indoor area could have various obstacles such as tables, chairs, sofas, people, etc.

To maximize the performance of the Boustrophedon Path algorithm, we decompose the MBR into free sub-areas, i.e., areas within the MBR without obstacles. This allows the AmaxCoverage algorithm to efficiently use the Boustrophedon Path algorithm.

Decomposition utilizes the Rectangle Tiling scheme [17], a mathematical technique that finds all possible sub-rectangles within a rectangle. Given a rectangle of size m x n, N(m, n) sub-rectangles will be in it, where N(m, n) can be computed following Equation (1). For each bottom left corner with coordinates (i, j), there are (m - i)(n - j) possible top right-corners.

$$N(m,n)=\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(m-i)(n-j)=\tfrac{1}{4}m(m+1)n(n+1) \tag{1}$$

To ensure the efficiency of the Boustrophedon Path algorithm, we need to exclude all the sub-rectangles that contain obstacles. However, finding *all possible* sub-rectangles within the MBR may result in Rectangle Tiling taking an unacceptable amount of time to process. To limit this time and increase performance, the AmaxCoverage algorithm discards smaller sub-rectangles below a threshold value by using a RectangleTilingwithoutObstacle algorithm that will generate the rectangles that are good candidates for efficient sub-rectangle decomposition.

```
Algorithm: RectangleTilingwithoutObstacle (mbr)
1: let mbr is a rectangle with an m x n rectangles
2: for i = 0 to m-1 do
3:   for j = 0 to n-1 do
4:     for p = i+1 to m do
5:      for q = j+1 to n do
6:         If(rectangle(i, j ,p, q) does not have any occupied cell and
                sizeof(rectangle(i, j, p, q) > robot size) then
7:              Add rectangle(i, j, p, q) to rectangles
8:         endif
9:       repeat
10:     repeat
11:   repeat
12: repeat
13: return rectangles
End RectangleTilingwithoutObstacle
```

Fig. 2. Rectangle Tiling Algorithm

The next step is to find a minimal set of sub-rectangles from the rectangles, that is, the set of candidates generated by the RectangleTilingwithoutObstacle algorithm so that the union of all the sub-rectangles in the resulting set covers the target. This problem is as hard as the well-known minimum Set Cover Problem, NP-Complete [18]. For example, consider that each candidate rectangle of size 1x1 is an element and all other candidate rectangles are a set of the unit rectangles. If you can find a minimum set cover in this problem, then you can find a minimum set of candidate rectangles.

A greedy algorithm is known to be an efficient approximation for the Set Cover Problem, so we used it to choose the set of rectangles from the candidates. The SetCovering algorithm selected the sets as follows:

```
Algorithm: SetCovering (rectangles)
1: while (uncovered cell exists)
2: Choose the rectangle that contains the largest number of uncovered cells from the rectangle list. If there are rectangles that have the same area size, give a higher priority to the rectangle that has the least number of turns within the rectangle when using the Boustrophedon Path algorithm.
3: Remove the rectangle from the rectangle list and add it to the sefofrectangle list.
4: endofwhile
5: return setofrectangles
End SetCovering
```

Fig. 3. Set Covering Algorithm

During each iteration, the set that contains the largest number of uncovered elements is chosen. If there are rectangles that have the same area size, the algorithm will assign a higher priority to the one that has fewer turns within it when using the Boustrophedon Path algorithm. In the SetCovering algorithm (Figure 3), the setofrectangles become the target area for coverage.

The next step is to determine the order of visits among the selected rectangles, i.e., the rectangles in setofrectangles. To do this, we measure the *coverage efficiency* of each rectangle. We define the *coverage efficiency* as the ratio of the rectangle size to the *estimated coverage time* required for it to be completely covered. The *estimated coverage time* for a rectangle is defined as the sum of the complete coverage time of rectangle plus the time it takes the robot to travel of the robot from its current location to the entrance of the rectangle.

The coverage time of a rectangle depends on several factors: the size of the rectangle, the speed of the robot, and the number of turns the robot must execute within the rectangle. Additionally, we note that given these factors the coverage time for the same size rectangles could be different due to the number of turns the robot must make to cover it. Even for rectangles that do not have any obstacles (as decomposed rectangles), the number of turns could be different since layouts differ. For example, consider two rectangles which have the same calculated area; one is a square and the other is not. If the robots path is along the length of the rectangle, then the square will require more turns than the non-square rectangle and the total coverage time for the square will be longer than the rectangle. Therefore, the number of turns is an important factor when trying to optimizing coverage time.

Once the *coverage efficiency* for all uncovered rectangles is evaluated, the rectangle that has the greatest coverage efficiency will be selected (denoted as R1 in the getR1 algorithm in Figure 4).

Now, the coverage process can begin for the R1 rectangle. The Boustrophedon Path algorithm, in our opinion, is the best choice for covering the R1 rectangle, albeit, other algorithms could also be applied.

```
Algorithm: getR1(setofrectangles, deadline)
1: Set CoverageArea as the number of cells in the rectangle.
2: RecCoverageTime = MoveTimeForOneCell*NumOfCells + TurnOverhead*NumOfTurns
3: EstimatedCoverageTime = TimeOfTravelToRec + RecCoverageTime
4: if(EstimatedCoverageTime > deadline) then
5:        EstimatedCoverageTime := deadline
6:        LeftTime = deadline - TimeOfTravelToRec
7:        CoverageArea = LeftTime / MoveTimeForOneCell
8: endif
9: CoverageEfficiency = CoverageArea / EstimatedCoverageTime
10: For all the rectangles in the setofrectangles left, find the next R1 that has the maximum
CoverageEfficiency
11: remove the R1 from setofrectangles
12: return R1
END getR1
```

Fig. 4. The getR1 Algorithm

```
Algorithm: coverR1 (rectangle r1)
1: Cover the r1 area by Boustrophedon Path algorithm.
2: The rest of this module would be done while the robot is covering the r1 area.
3: if (new obstacle within r1) then
4:         Avoid the obstacle and update the Map;
6: endif
7: if (new obstacle or new free area from mbr) then
8:         Update the Map;
9:         RECchanged := True;
10: endif
11: if the new obstacle or free area is from Unknown area
12:         MBRchanged = True;
13:         Update the Map;
14: endif
End
```

Fig. 5. The coverR1 Algorithm

```
Algorithm: AmaxCoverage(deadline)
1: Global Map
2: Bool MBRchanged, RECchanged
3: MBRchanged :=FALSE, RECchanged :=FALSE
4: mbr := call MinimumBoundingRectangle()
5: rectangles := call RectangleTilingwithoutObstacle(mbr)
6: setofrectangles := call SetCovering(rectangles)
7: r1 : = call getR1(setofrectangles, deadline)
8: while(r1 is not null and deadline is not exhausted)
9:         move to the initial position of r1
10:        call CoverR1(r1)
11:        if(MBRchanged == True) Then
12:            MBRchanged := False
13:            goto line 4
14:        endif
15:        if(RECchanged == True) Then
16:            RECchanged := False
17:            goto line 5
18:            r1 : = call getR1(setofrectangles)
19:        endif
20: endofwhile
END AmaxCoverage
```

Fig. 6. The AmaxCoverage Algorithm

After the robot covers the current rectangle, R1, the algorithm calculates the next R1 and covers it, this process continues until R1 is null. However, as we previously stated, the map of the environment is dynamic; it can change if the robot senses new information. For example, if the robot is covering the R1 rectangle, and encounters a new obstacle or senses a new free area that was previously known as unknown area (i.e., it was not mapped), and then the robot will update the new information on the map. The strategy for completing this is as follows.

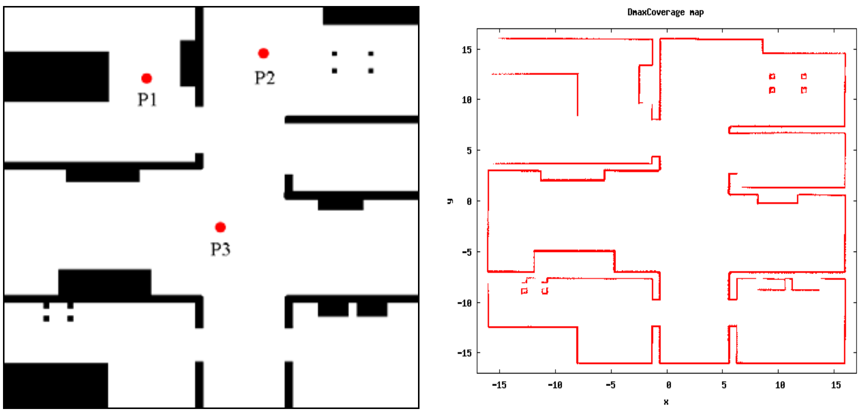
If a new obstacle is found in the R1 area, the robot will avoid it and update the map. If a new obstacle or free area is found in the MBR area, the rectangle tiling process should be recalculated. If either of them are found outside the MBR area, that is, in an unknown area, the MBR needs to be recalculated since the current MBR is no longer valid for the current map. Figure 5 shows the algorithm for how the robot updates and handles the new information while it is covering the R1 area. Figure 6 gives a summary of the AmaxCoverage algorithm.

4 Experimental Results and Analysis

In this section, we evaluate the performance of the AmaxCoverage algorithm. Through the experiments, we show how the AmaxCoverage algorithm is implemented and its performance results within an unknown environment.

The AmaxCoverage algorithm was implemented using Player/Stage, versions 2.1.1 and 2.10. We used the Pioneer 2 DX robot with the SICK laser (LMS200) sensor for the experiments. To build a real-time map for the simulated indoor environment, we utilized the FastSLAM algorithm.

Several maps were used for the evaluation of various indoor layouts. Figure 7(a) is a representative example. The dark-black-colored areas represent walls and obstacles and the three red circles, denoted as P1, P2, and P3, represent the starting points of the robot in our experiments, these were used to verify the stability of the AmaxCoverage algorithm's performance. The white area represents the free area, i.e., area to be cleaned. To evaluate the performance of the algorithm, the robot was set with an average velocity of 0.4m/s and a delay time of 0.2 m/s per turn.

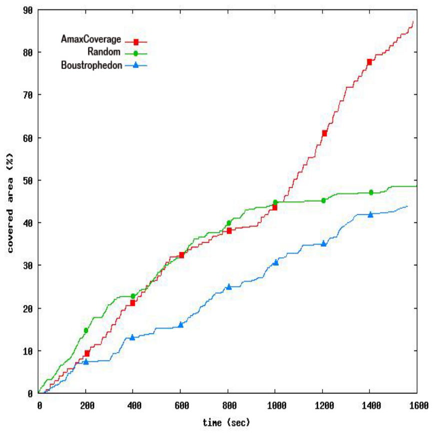


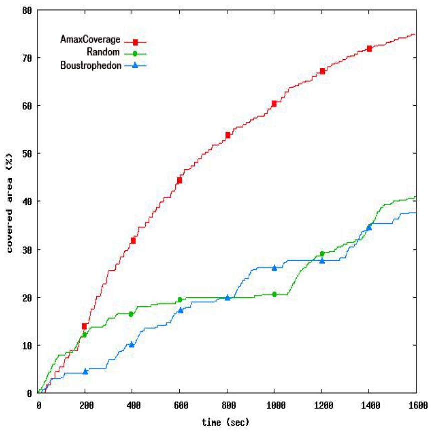
(a) Sample Area (left) (b) Map Constructed after 30 Minutes by AmaxCoverage(right)

Fig. 7. Sample Area and Map constructed by AmaxCoverage algorithm

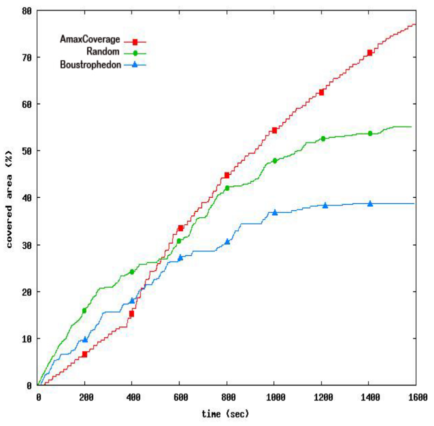
Since the robot does not have any *a priori* information about the environment, it needs to build a map using the FastSLAM algorithm starting from one of the initial positions.

Figure 8 shows the performance evaluation results of the AmaxCoverage algorithm compared with the Boustrophedon Path and Random algorithms. The graphs show the percentage of the completed area over time. At the start of the experiments, the AmaxCoverage algorithm is comparable to the Random and worse than the Boustrophedon due to the planning process, which includes rectangle tiling and set covering. However, in the latter phases of the experiments, the AmaxCoverage algorithm reduces cleaning time by almost 50% compared with the other algorithms regardless of the initial starting point.





(a) Results from initial position ‘P1’ (b) Results from initial position ‘P2’



(c) Results from initial position ‘P3’

Fig. 8. Cleaning completion rate

Another advantage of the AmaxCoverage algorithm is shown in Figure 7(b). This map was constructed after 30 minutes of covering using the AmaxCoverage algorithm. As shown, the map from the AmaxCoverage algorithm is most similar to the original map in Figure 7(a) compared with the results of the Random and Boustrophedon algorithms discussed in Section 1 and shown in Figure 1.

Our experiments suggest the AmaxCoverage algorithm can complete the coverage of an unknown environment much faster if a human cooperates with the robot on the task. Figure 8 shows that the covering of the AmaxCoverage algorithm achieved an 80%–90% completion rate in the same amount of time that the covering of the Boustrophedon algorithm reached only a 40%–50% completion rate. Therefore, if a human working in cooperation with the AmaxCoverage algorithm-based robot, covered the remaining complex areas, i.e., non-free areas, the total covering time could be decreased even more. This is possible because the AmaxCoverage algorithm maximizes the coverage of free areas, which can be covered more quickly than non-free areas, in other words, the areas that a human can, at this time, cover much more efficiently.

5 Conclusion

In this paper, we proposed a new coverage algorithm, AmaxCoverage. The AmaxCoverage algorithm efficiently integrates with a SLAM algorithm for the unknown environment in real time. For covering the target area, the algorithm gives priority to large mostly empty spaces with few obstacles compared to complex areas. Our approach allows the algorithm to achieve high coverage efficiency and when working cooperatively with a human will lead to more efficient total coverage of an unknown environment when compared to the Random and Boustrophedon algorithms.

Our future research includes on how to detect and account for dynamic obstacles, such as people, pets, etc. Additionally, we are working on extending the types of grid map the AmaxCoverage algorithm can use, e.g., topological maps.

References

1. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)
2. May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)
3. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
4. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)

5. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
6. Zelinsky, A., Jarvis, R.A., Byrne, J.C., Yuta, S.: Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot. In: Proceedings of International Conference on Advanced Robotics, Tokyo, Japan, pp. 533–538 (November 1993)
7. Choset, H., Pignon, P.: Coverage Path Planning: the boustrophedon cellular decomposition. In: Proceedings of the International Conference on Field and Service Robotics, Canberra, Australia (December 1997)
8. Neumann de Carvalho, R., Vidal, H.A., Vieria, P., Riberio, M.I.: Complete Coverage Path Planning Guidance for Cleaning Robots. In: Proceedings of the IEEE Int. Symposium on Industrial Electronics, vol. 2 (1997)
9. Wong, S.C., MacDonald, B.A.: A topological coverage algorithm for mobile robots. In: Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, Las Vegas, Nevada (October 2003)
10. Yoon, S.H., Park, S.H., Choi, B.J., Lee, Y.J.: Path Planning for Cleaning Robots: A Graph Model Approach. In: Proceedings of the International Conference on Control, Automation and Systems, Cheju National University, Jeju, Korea, October 17-21, pp. 2199–2202 (2001)
11. Agmon, N., Hozon, M., Kaminka, G.A.: Constructing Spanning Trees for Efficient Multi-Robot Coverage. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida (May 2006)
12. Hazon, N., Mieli, F., Kaminka, G.A.: Towards Robust On-line Multi-Robot Coverage. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida (May 2006)
13. Doh, N.L., Kim, C., Chung, W.K.: A Practical Path Planner for the Robotic Vacuum Cleaner in Rectilinear Environments. IEEE Transactions on Consumer Electronics 53(2), 519–527 (2007)
14. Williams, K., Burdick, J.: Multi-robot Boundary Coverage with Plan Revision. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, Florida (May 2006)
15. Batalin, M.A., Sukhatme, G.S.: Coverage, Exploration and Deployment by a Mobile Robot and Communication Network. In: Zhao, F., Guibas, L.J. (eds.) IPSN 2003. LNCS, vol. 2634, pp. 376–391. Springer, Heidelberg (2003)
16. Schwager, M., Slotine, J.-J., Rus, D.: Decentralized, Adaptive Control for Coverage with Networked Robots. In: Proceedings of the 2007 IEEE International Conference on Robotics and Automation, Roma, Italy, April 10-14, pp. 10–14 (2007)
17. Taik, O.Y.: Survey on Cleaning Robot Consumer. In: Proceedings of 2nd Annual Workshop of Korea Robotics Society (2005)
18. Bailey, T., Durrant-Whyte, H.: Simultaneous Localization and Mapping (SLAM): Part I. IEEE Robotics & Automation Magazine 13(2), 99–108 (2006)
19. Dissanayake, G., Newman, P., Durrant-Whyte, H.F., Clark, S., Csobra, M.: A Solution to the Simultaneous Localisation and Mapping (SLAM) Problem. IEEE Transactions on Robotics and Automation 17(3), 229–241 (2001)
20. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: Fast-SLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In: Proceedings of AAAI National Conference on Artificial Intelligence, pp. 593–598 (2002)

21. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: Fast-SLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Probably Converges. In: Proceedings of International Joint Conference on Artificial Intelligence, pp. 1151–1156 (2003)
22. Stewart, I.: Squaring the Square. Scientific American, 74–76 (July 1997)
23. Feige, U.: A Threshold of ln n for Approximating Set Cover. Journal of the ACM (JACM) 45(4), 634–652 (1998)
24. Fox, D.: Adapting the Sample Size in Particle Filters through KLD-Sampling. International Journal of Robotics Research 22 (2003)