An Efficient Area Maximizing Coverage Algorithm for Intelligent Robots with Deadline Situations

Heung Seok Jeon*

Department of Computer Engineering, Konkuk University School of Natural Science, Chungju-city 380-701, Korea
hsjeon@kku.ac.kr

Abstract

Coverage algorithm is one of the core technologies required for intelligent robots such as cleaning robots, harvesting robots, painting robots and lawn mowing robots. Although many smart coverage algorithms have been proposed, to the best of our knowledge, all of them make the same assumption that they have sufficient time to cover the entire target area. However, the time to completely cover the whole target area may not always be available. Therefore, in this paper, we propose another new coverage scheme, which we call the DmaxCoverage algorithm that decides the coverage path by considering the deadline for coverage. This approach can be beneficial when the time for the coverage is not sufficient to cover the entire target area. Experimental results show that the DmaxCoverage algorithm outperforms previous algorithms for these situations.

Key Words*: Mobile Robot, SLAM, Coverage algorithm*

1. Introduction

Coverage algorithm is one of the fundamental technologies for intelligent mobile robots [1]. Most previous research concerning coverage algorithms have focused on the problem of finding a complete coverage path for the target area. In other words, the common goal of previous approaches is to cover the entire target area and to minimize the elapsed time for the coverage without human intervention. For this reason, a mobile robot using previous algorithms does not guarantee the best performance when the time for coverage is not sufficient to cover the entire area.

Coverage algorithms that are based on random movement are widely used in real world robots. This is mostly because the implementation of the algorithm is simple and does not require expensive sensors. However, these algorithms cannot guarantee that the whole workspace will be covered within the deadline constrained by battery power and time limitation [2-3].

Many smart coverage algorithms have been proposed to guarantee complete coverage. The most powerful ones are those that rely on distance transform for planning paths of the mobile robot [4-5]. However, these schemes have a limitation in that it requires a complete grid map of the environment. That is, these schemes assume they have a complete map of the environment. Thus, distance transform based algorithms cannot be applied to unknown environments.

The fundamental problem of these approaches is that getting the entire map of a region is not always easy. Also, they still need to handle dynamic obstacles and moving objects. In order to cover unknown environments, mobile robots should have the ability to incrementally build a map of its environment. For this reason, we utilize a Simultaneous Localization and Mapping (SLAM) algorithm. The SLAM algorithm performs by simultaneously estimating the position of the robot and building the map of the environment.

* Corresponding Author

Thus, in this paper, we propose a new on-line coverage algorithm called DmaxCoverage, which considers the deadline constraints. Our algorithm is capable of covering as much area as possible within the deadline and assumes that the robot does not possess a map of the environment for coverage in advance. In order to cover an area of an unknown environment, a SLAM algorithm is integrated into DmaxCoverage.

The rest of this paper is organized as follow. In the next section, previous algorithms are reviewed. Our algorithm is described in Section 3. Section 4 shows the test results. Finally, Section 5 concludes this paper.

2. Related Works

Most previous coverage algorithms use the Boustrophedon Path algorithm to cover the workspace. In the Boustrophedon Path algorithm, the robot crosses the full length of the area in a straight line, turns and then traces a new straight line path adjacent to the previous one. By repeating this procedure, the robot is guaranteed to cover the entire target area [2-3].

The Boustrophedon Path algorithm shows very good performance in small areas and in places without obstacles. However, it suffers from performance degradation in complex environments that have many scattered objects. Performance of robots in such environments drops drastically.

Distance transform based coverage algorithms are widely used for complete coverage path planning [4-5]. Coverage path generated by these algorithms have several problems. The fundamental problem is that they are based on the assumption that a complete grid map of the environment is available. That is, a complete grid map is required to complete a coverage path.

The Backtracking Spiral Algorithm (BSA) is an online complete coverage path planning algorithm for mobile robots [6]. Unlike previous algorithms, it does not require a pre-built grid map of an environment. However, it uses an external device such as a ceiling camera for localization of the mobile robot. Furthermore, it has a limitation that it requires a large amount of memory to stack the backtracking points.

To get a map for the target area and to maintain the location of the robot in real time, solutions for the Simultaneous Localization and Mapping (SLAM) problem are required. The SLAM problem asks if it is possible for a mobile robot to be placed at an unknown location in an unknown environment and for the robot to incrementally build a consistent map of an environment while simultaneously determining its location within this map. The SLAM problem has been one of the major issues for the mobile robotics community as it would provide the means to make a robot truly autonomous. While EKF-SLAM [7] and FastSLAM [8] are the two most important solutions, newer alternatives that offer much potential have also been proposed [9]. We have integrated FastSLAM with our DmaxCoverage algorithm that we describe below.

Ten data sets of TM proteins are examined in this study (see Table 1). These contain the sequences of four 4-tms (acetylcholine receptor, GABA receptor, gap junction, and hydrogen ion transporter), two 6-tms (photosystem II and ABC transporter), and four 12-tms (solute carrier, amino acid transporter, sodium dependent transporter, and sugar transporter) protein groups. The data are extracted from SWISS-PROT Release 41.00 [6], according to its functional descriptions in DE, KW and/or CC lines [7]. All extracted sequences are full-length ones.

We focus on these 4-tms, 6-tms, and 12-tms protein groups, because they have a larger number of sequences than other ones registered in SWISS-PROT. These sequences were investigated first by the prediction of signal peptide region with DetecSig [8], and then by the TM topology prediction with ConPred II [9]. These predictions about the signal peptide and the number of TMSs are used for checking the SWISS-PROT descriptions. The length of TMS regions predicted by ConPred II is adjusted to 21 residues.

3. DmaxCoverage Algorithm

When an intelligent robot is placed in a new environment without previous information about the environment, the robot has to somehow decide the target area to be covered. In our approach, this is done by first scanning the local area to build a map of the target area using sensors and SLAM algorithms. Then, the DmaxConverage algorithm computes the Minimum Bounding Rectangles (MBR) (Figure 2: line 2).

The MBR is the minimal rectangle that includes all the free areas and the areas occupied by the obstacles and the walls of the current local map. MBR may be a current snapshot of the map that may dynamically change with time. Then, we decompose the MBR into subareas that do not contain obstacles utilizing the Rectangle Tiling scheme [10]. The Rectangle Tiling scheme is a mathematical technique that finds all possible sub-rectangles within a rectangle.

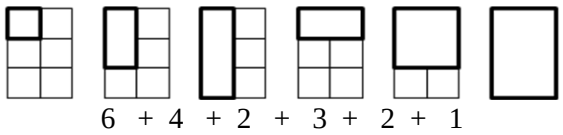
Figure 1 shows examples of rectangle tiling. The very first rectangle with 6 cells is decomposed into 18 subareas. Figure 1(b) shows a rectangle tiling with obstacle (gray cell). Rectangle Tiling removes a rectangle with obstacles from the rectangle list when decomposing a rectangle. The very first rectangle of Figure 1(b) is decomposed into 10 subareas that do not include an obstacle cell.

A rectangle of size m x n contains N(m, n) sub-rectangles, where N(m, n) can be computed as Equation (1). For each bottom left corner with coordinates (i, j), there are (m - i)(n - j) possible top right-corners.

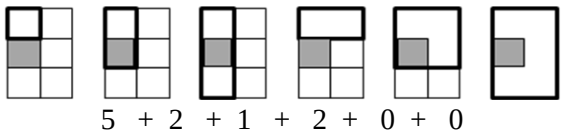
$$N(m,n)=\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(m-i)(n-j)=\tfrac{1}{4}m(m+1)n(n+1). \qquad (1)$$

For example, if an environment size is 10mx10m and robot size is 40cmx40cm, we can make a grid map of 25x25 cells (width 1000cm/40cm, height 1000cm/40cm).

The next step is to find a minimal set of sub-rectangles from the rectangles that result from the Rectangle Tiling (Figure 2: line 5). A greedy algorithm is known to be an efficient approximation algorithm to the Set Cover problem, so we utilize the greedy algorithm to choose the set of rectangles.



(a) Example of rectangle tiling without obstacles



(b) Example of rectangle tiling with one obstacles

Figure 1. Examples of Rectangle Tiling

The next step is to decide the visiting sequence among the selected rectangles, that is, *setofrectangles* (Figure 2: line 6). To determine the visiting sequence, we evaluate the *coverage efficiency* value of each rectangle. We define the *coverage efficiency* as the ratio of the rectangle size to the estimated coverage time for this rectangle. The estimated coverage time for a rectangle is defined as the sum of the coverage time for the rectangle and the

traveling time of the robot to the rectangle. The traveling time is the required time for the robot to move from the current location to the entrance of the next rectangle.

The coverage time for a rectangle depends not only on the size of the rectangle and the speed of the robot, but also on the number of turns that the robot has to make within the rectangle. The coverage time for the same size rectangles could be different because of the different number of turns. Even for the rectangles that does not have any obstacles (as decomposed rectangles), the number of turns could be different as layouts differ. For example, imagine two rectangles of the same area, where one is a square while the other is not. If the robot moves in the rectangle along the longer side, then the square will require more turns than the non-square rectangle. Thus, the total coverage time for the square would be longer than the rectangle. Thus, the number of turns is one of the important factors in estimating the coverage time.

Once the coverage efficiencies for all uncovered rectangles are evaluated, the rectangle R1 that shows the biggest coverage efficiency will be selected. The function getR1 in Figure 2 is the algorithm that selects the R1 rectangle.

Now, the actual coverage process can begin for the R1 rectangle. The selected rectangle R1 is covered by the Boustrophedon Path algorithm [2], which shows very good performance in areas without obstacles (Figure 2: line 10).

```
Algorithm: DmaxCoverage(deadline)
1: Local 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 DmaxCoverage 2.0
```

Figure 2. The DmaxCoverage Algorithm

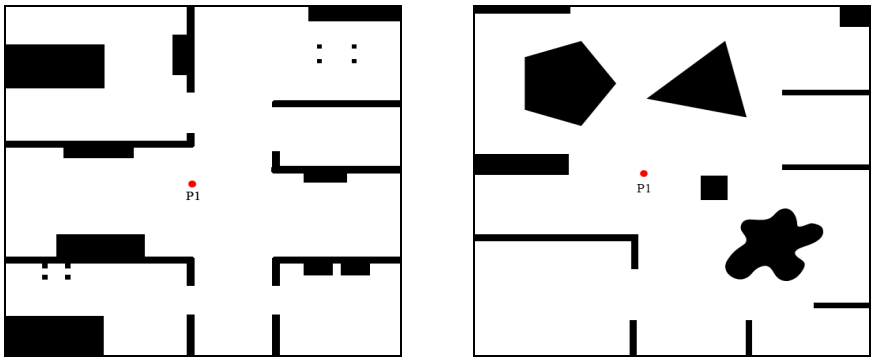
After the robot covers the R1 rectangle, the algorithm repeatedly calculates and covers the next R1 rectangle until the next R1 rectangle is null. However, as we mentioned before, the

map could be changed by new information from the unknown environment. While the robot is covering the R1 rectangle, it can encounter a new obstacle or realize a new free area that was previously known as an unknown area. The robot, then, should update the map with the new information and handle it appropriately, as described as follows.

If a new obstacle is found in the R1 area, the robot just avoids it and updates the map. If a new obstacle or a new free area is found within the MBR area, the rectangle tiling process is repeated. If they are found outside of the MBR area, that is, in an unknown area, the MBR needs to be calculated again because the current MBR is no longer valid for the current map. Figure 2 summarizes the whole DmaxCoverage algorithm.

4. Experimental Results

Simulations using Player/Stage [11] are performed in order to evaluate the ideas presented in this paper. We use the Pioneer 2 DX robot model with Sick laser range finders. We also utilize the FastSLAM algorithm that is integrated into our approach for map building and localization [8]. Figure 3 shows a map of the indoor environment used for experiments.



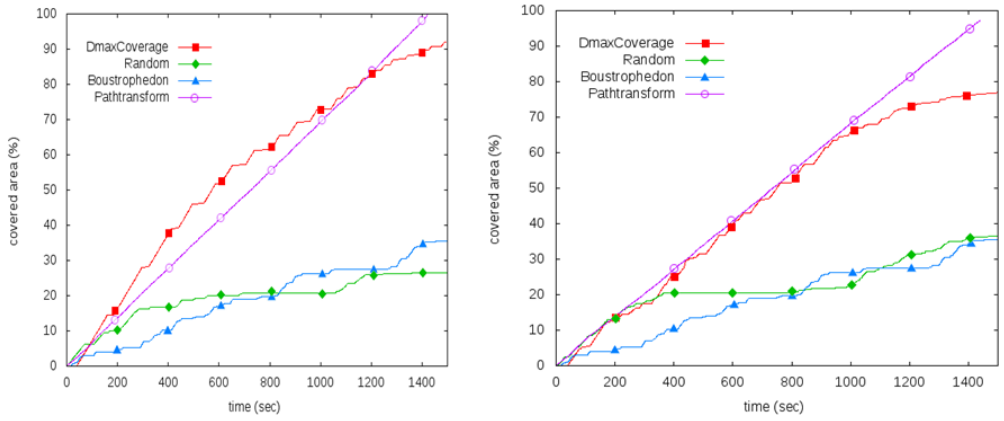
(a) Simple environment (b) Complex environment

Figure 3. Environment Maps with Initial Point P1

4.1. Coverage Evaluation

Several algorithms were used for performance evaluation. Random and Boustrophedon path algorithms are the most commonly used algorithms in commercial cleaning robots. Path Transform is an ideal algorithm to find the complete coverage path [4-5]. However, the Path Transform algorithm assumes that it knows the map of the environment in advance. Therefore, the performance of using this algorithm is used as a comparative metric.

Figure 4 shows the percentage of coverage, that is, the rate of the covered area, per time. We observe that the DmaxCoverage algorithm shows better performance than the Random and Boustrophedon algorithms. Furthermore, for the simple map environment case, the DmaxCoverage algorithm shows better performance than the Path Transform algorithm for most of the time (Figure 4(a)). This is because the turn overhead of DmaxCoverage is 9.5% smaller than the Path Transform algorithm.



(a) Results of simple map environment (b) Results of complex map environment

Figure 4. Coverage Completion Rate

The performance of the algorithm is very close to the performance of the Path Transform algorithm in the complex environment (Figure 4(b)). The DmaxCoverage takes more time between rectangles than in the simple environment. Analysis shows that the robot moves 16% more than in the simple environment.

4.2 Deadline Evaluation

Mobile robots may have deadline conditions such as battery power and time limitation. For example, when the remaining battery of a robot is not sufficient to cover the entire target area and recharging would take non-trivial time such that the robot cannot return and resume cleaning quickly enough. Therefore, in such cases, the cleaning robot may need to cover as much area as possible before the battery runs out.

Figure 5 shows a deadline example. Assuming that the robot starts covering at position S and stops at position E, it needs to choose the next rectangle. However, the next candidate rectangles are R2, R3, and the sizes of the two rectangles are the same. Considering coverage efficiency such as the rectangle size and the traveling time, the robot selects R3 as its next target.

After coverage of R3, R5 and R4 rectangles become the next target candidates. Although R5 is farther away from R3 than R4, R5 can be selected if the coverage efficiency of R5 is larger than that of R4.

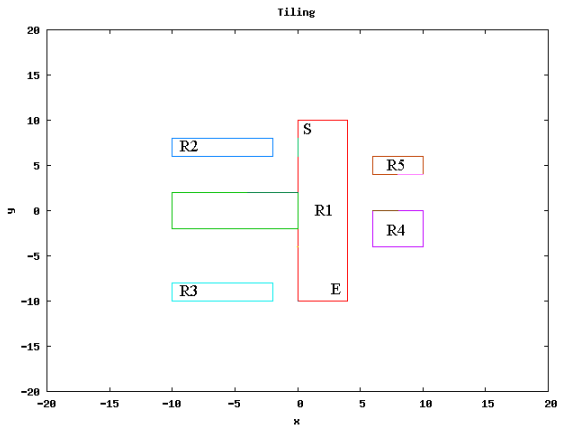


Figure 5. Tiling Map using First Local Map

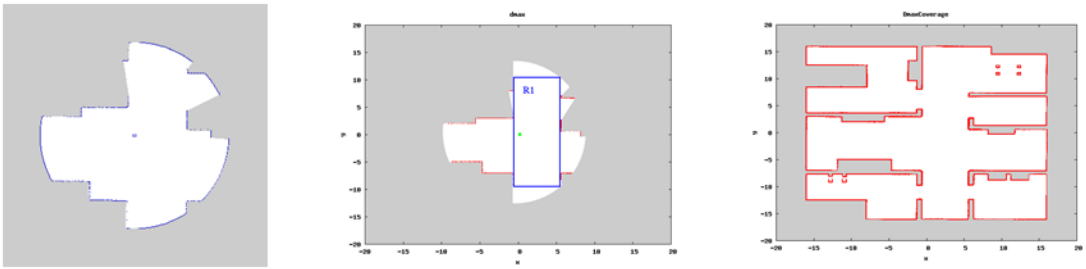
Table 1 shows the performance evaluation results of the DmaxCoverage algorithm itself for various deadlines. The results show that there is roughly a 5% improvement when the deadline is considered over when the deadline is not considered.

Table 1. Covered area (%) of the DmaxCoverage with Various Deadlines

Deadlines (sec.)	Covered Area (%)	
	without deadline	with deadline
700	35.28	37.14
800	37.95	38.10
900	39.12	42.61
1000	44.12	48.312

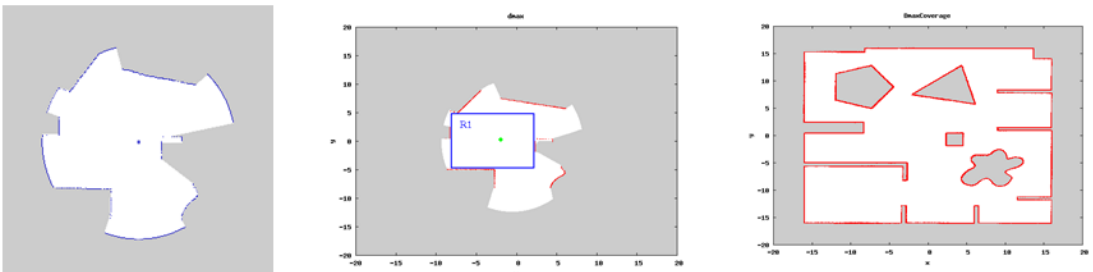
4.3. Mapping Evaluation

When a mobile robot is placed in an unknown environment, it has to build a map of its environment for coverage. To build a map of the environment, we utilize the Fast-SLAM algorithm. Figure 6 and 7 shows the different stages of the map that is built. Figure 6(a) and Figure 7(a) is the initial scanning status in the unknown environment. The mobile robot utilizing the DmaxCoverage algorithm makes a first local map using sensor data that is gathered. Then the robot uses the first local map for Rectangle Tiling and chooses rectangle R1 (Figure 6(b) and 7(b)). The robot will continue to update the map while covering the rectangle. Figure 6(c) and Figure 7(c) shows the result of the map after coverage.



(a) The initial laser data (b) R1 rectangle using first map (c) Result map

Figure 6. Simple Environment Results



(a) The initial laser data (b) R1 rectangle using first map (c) Result map

Figure 7. Complex Environment Results

5. Conclusions

In this paper, we proposed a new coverage algorithm, which we call DmaxCoverage. The DmaxCoverage algorithm is designed for situations where the robot has a deadline at which time it must terminate its coverage. Experimental results show that the DmaxCoverage algorithm shows better performance than previous algorithms. We are currently working on how to detect and handle dynamic obstacles such as people and pets.

References

[1] R. D. Schraft, M. Hägele and H. Volz, "Service robots: the appropriate level of automation and the role of users/operators in the task execution", In Proceedings of the International Conference Systems, Man, and Cybernetics, vol. 4, **(1993)**, pp.163-169.
[2] H. Choset and P. Pignon, "Coverage Path Planning: the Boustrophedon Cellular Decomposition", In Proceedings of the International Conference on Field and Service Robotics, Canberra, Australia, **(1997)**.
[3] E. Gonzalez, A. Suarez, C. Moreno and F. Artigue, "Complementary Regions: a Surface Filling Algorithm", In Proceedings of the International Conference on Robotics and Automation, **(1996)**, pp. 909-914.
[4] A. Zelinsky, R. A. Jarvis, J. C. Byrne and S. Yuta, "Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot", In Proceedings of the International Conference on Advanced Robotics, **(1993)**, pp. 533-538.
[5] Y. Choi, T. Lee, S. Beak and S. Oh, "Online Complete Path Planning for Mobile Robots Based on Linked Spiral Paths Using Constrained Inverse Distance Transform", In Proceedings of the International Conference on Intelligent Robots and Systems, **(2005)**, pp. 5788-5793.
[6] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra and C. Bustacara, "BSA: A complete coverage algorithm", In Proceedings of the International Conference on Robotics and Automation, **(2005)**, pp. 2040-2044.
[7] G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark and M. Csobra, "A Solution to the Simultaneous Localisation and Mapping (SLAM) Problem", IEEE Transactions on Robotics and Automation, **(2001)**, pp. 229-241.
[8] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem", In Proceedings of the AAAI National Conference on Artificial Intelligence, **(2002)**, pp. 593-598.
[9] A. Eliazar and R. Parr, "DP-SLAM 2.0", In Proceedings of the International Conference on Robotics and Automation, **(2004)**, pp. 1314-1320.
[10] I. Stewart, Scientific American, Squaring the Square, vol. 277, **(1997)** July, pp. 94-96.
[11] The Player Project, http://playerstage.sourceforge.net/.