# An Online Complete Coverage Algorithm for Cleaning Robots based on Boustrophedon Motions and A* search

Abdallah Ntawumenyikizaba
Dept. of Computer Engineering,
Kyung Hee University
1-Seocheon, Giheung, Yongin,
Gyeonggi, 449-701, South Korea
abdallah@khu.ac.kr

Hoang Huu Viet
Dept. of Computer Engineering,
Kyung Hee University
1-Seocheon, Giheung, Yongin,
Gyeonggi, 449-701, South Korea
viethh@khu.ac.kr

TaeChoong Chung
Dept. of Computer Engineering,
Kyung Hee University
1-Seocheon, Giheung, Yongin,
Gyeonggi, 449-701, South Korea
tcchung@khu.ac.kr

***Abstract*— This paper presents an online complete coverage algorithm for cleaning robots based on boustrophedon motions combined with A* search algorithm. In our approach, while performing a boustrophedon motion to cover an unvisited area in the workspace, the robot detects and stores backtracking points. To execute the next boustrophedon motion, A* search is employed as a backtracking mechanism which guides the robot to the nearest backtracking point. Experimental results prove that the proposed algorithm ensures the complete coverage of the workspace in the finite moving steps of the robot. Furthermore, our proposed approach is efficient in terms of the covered path length and the number of boustrophedon motions.**

***Keywords- A* algorithm; boustrophedon motion; cleaning robots; complete coverage***

## I. INTRODUCTION

Cleaning robots are among service robots, where are used in our everyday life for cleaning task. In cleaning process, in order to obtain a superior cleaning performance, the robot should have the capability to reach and clean all available area in workspace. Complete Coverage Path Planning (CCPP) is the determination of a path that a robot must take in order to pass over each point in an environment. To efficiently accomplish complete coverage navigation it requires to take into account some criteria and constraints. Firstly, the moving trajectory of cleaning robot must ensure the coverage of all accessible area as well as the ending point should be reachable. Secondly, the cleaning robot has no prior knowledge about environment and the numbers of obstacles are finite and unknown. The environment is finite and closed, and the starting position of the robot is not fixed. Thirdly, a cleaning robot should minimize the cleaning time and overlapping paths. Fourthly, with no constraints on the size and position of obstacles, sensors have to detect them correctly. Fifthly, the complete coverage navigation plan should be composed of sequential and continuous operations, and simple motion paths are preferred. And lastly, the complete coverage navigation algorithm must be preferably systematic and structured, avoiding complex mixtures of cases and conditions.

In recent years, many studies on the complete coverage task of cleaning robots have been proposed. In the case of known environments, the CCPP problem can be solved using methods such as random path planning [1,2], genetic algorithms [3], neural networks [4], exact cellular decomposition [5,6,7], spanning trees [8], spiral filling paths [9,10,11], and ant colony [12,13]. In the case of unknown environments, most commercial robots use inefficient random path planning algorithms [14] and very few inexpensive contact collision sensors to get complete coverage.

In this paper, we propose a novel approach to the online complete coverage of unknown environments that can be applied to autonomous cleaning robots. In our approach, a robot achieves the complete coverage by performing simple boustrophedon motions and using a backtracking mechanism based on A* search algorithm to link boustrophedon paths. The experimental results prove that the proposed approach ensures the complete coverage of the unknown environment in the finite moving steps of the robot. Furthermore, our proposed method is efficient in terms of the path length and the number of boustrophedon motions.

The rest of this paper is organized as follows: Section 2 describes a short review of A* algorithm related to our approach. Section 3 presents our approach to complete coverage problem. The simulation results are discussed in Section 4. Finally, we conclude our work in Section 5.

## II. A* SEARCH

The A* search [15] is a best-first search algorithm that finds the shortest path from the start node to the goal node in a graph. A* search uses an estimate function for each node $s$ of the graph: $f(s) = g(s) + h(s)$, where $g(s)$ is the actual distance to reach node $s$ found so far, and $h(s)$ is the *heuristic estimate* of the distance from node $s$ to the goal node. So the function $f(s)$ represents an estimate of the total cost of the path from the start node through node $s$ to the goal node.

The pseudocode of A* search is shown as in Algorithm 1 [16]. A* search maintains two lists, an *open* list and a *closed*

list. The *open* list is a priority queue that contains nodes to be considered for expansion. The *closed* list contains nodes that have already been expanded and ensures that each node is expanded only once. Besides, the *parent* list is used to extract the path from the start node $s_{start}$ to the goal node $s_{goal}$ after A* ends. Initially, the starting node $s_{start}$ and its cost function $f(s)$ are inserted into *open* list, and the *parent* list of node $s_{start}$ contains node $s_{start}$. At each iteration, the node $s$ of the top of *open* list is taken out by procedure *open.Pop()*, and is checked whether it is the goal node $s_{goal}$. If so, then A* search ends and the shortest path between the start node and the goal node is reported. Otherwise, A* adds node $s$ to *closed* list. Next, if each neighboring node $s'$, *nghb(s)*, of node $s$ does not belong to *closed* list, A* calculates $g(s')$ and the parent of node $s'$ by considering the cost path $g(s)$ from the start node $s_{start}$ to node $s$ and from $s$ to $s'$ in a straight line, $c(s,s')$, resulting in a length of $g(s) + c(s',s)$. It inserts the g-value and parent of node $s'$ into the *open* list if this new path is shorter than the shortest path $g(s')$ which is the path length from the start node to node $s'$ found so far. The search process continues until the goal node is chosen, or *open* list is empty. In the latter case, A* stops with no solution.

**Algorithm 1**: A* search algorithm

```
1:  Inputs: s_start, s_goal;
2:  Output: path;
3:  g(s_start) := 0;
4:  parent(s_start) := s_start;
5:  open := ∅;
6:  f(s_start) := g(s_start) + h(s_start);
7:  open.Insert(s_start,f(s_start));
8:  closed := ∅;
9:  while (open ≠ ∅) do
10:     s := open.Pop();
11:     if (s = s_goal) then /* extract the solution */
12:        path := s;
13:        while (parent(s) ≠ s_start) do
14:           s := parent(s);
15:           path := s ∪ path;
16:        endwhile
17:        break;
18:     endif
19:     closed := closed ∪ {s};
20:     for each s' ∈ nghb(s) do
21:        if s' ∉ closed then
22:           if s' ∉ open then
23:              g(s') := ∞;
24:              parent(s') := null;
25:           endif
26:           if (g(s) + c(s,s') < g(s')) then
27:              g(s') := g(s) + c(s,s');
28:              parent(s') := s;
29:              if s' ∈ open then
30:                 open.Remove(s');
31:              endif
32:              f(s') := g(s') + h(s');
33:              open.Insert(s',f(s'));
34:           endif
35:        endif
36:     endfor
37: endwhile
```

## III. THE PROPOSED METHOD

This section describes our method to solve the complete coverage of cleaning robots in unknown environments. A boustrophedon motion is a sequence of simple back-and-forth motion as shown in Fig. 1. Based on the boustrophedon motion, it can be seen that the robot can move from the current position to one of its four adjacent positions based on the four directions including East (*E*), North (*N*), West (*W*), and South(*S*), except directions that lead the robot into obstacles. As a result, the robot's moves generate a uniform grid where cells are of the size of the robot in its workspace. Hereafter, a position of the robot is identified as a cell of grid.

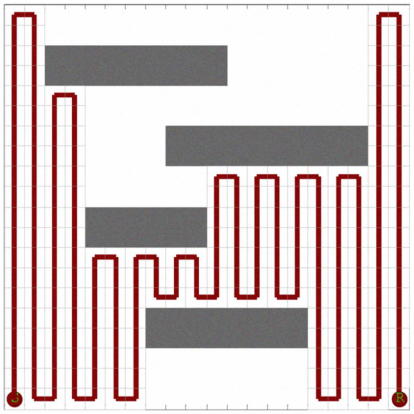

Figure 1. A boustrophedon motion and the uniform grid is generated by robot's moves. The positions **S** and **R** are the starting and ending positions of the boustrophedon motion, respectively.

Since a single boustrophedon motion cannot cover the entire workspace, multiple boustrophedon motions need to be performed to ensure the complete coverage as shown in Fig. 2. Fig. 2 shows that if each boustrophedon covers a region, called *boustrophedon region*, then the workspace is the union of non-intersecting boustrophedon regions. The smallest boustrophedon region is a cell which is of the size of the robot.

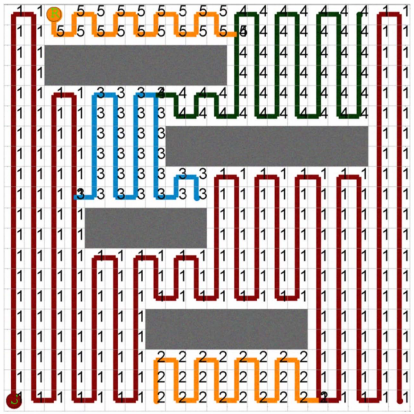

Figure 2. Five regions are made by five boustrophedon motions which are labeled from 1 to 5.

The key idea in our approach is that the robot's free workspace is broken down into boustrophedon regions while cleaning process, where each boustrophedon region is covered by a boustrophedon motion. To backtracking uncovered region,

the A* search is employed to determine the nearest backtracking point or cell and to avoid obstacles while moving to the nearest backtracking point. Once the robot covers all boustrophedon regions, it has covered the entire free region of the workspace. To achieve this idea, we first give two formal definitions as follows:

*Definition 1*: Backtracking points are points or cells that have at least one uncovered neighboring cell.

*Definition 2*: The critical point or ending point of a boustrophedon motion is the point or cell that its neighboring cells have been covered or are obstacles.

Since backtracking points are candidates of the starting point of the next boustrophedon motion, they must be detected and stored during the execution of boustrophedon motions.

Let *model* be the grid-like model of the workspace which is constructed incrementally as the robot moves. Let $s_{start}$ be the starting point and $s_{cp}$ be the critical point of each boustrophedon motion. Let $bp_{list}$ be a list of the backtracking points and $s = (x_s,y_s)$ be the current position of the robot. Algorithm 2 depicts the execution of a boustrophedon motion. An each iteration, the robot locates at cell $s = (x_s,y_s)$ and observes its neighboring cells in direction order of *N*, *S*, *E*, and *W*. If exists a free neighboring cell, the robot will cover it immediately. When a cell is covered, it is marked as the value of 1 or a "*virtual obstacle*" and it is not accessible to the robot while executing the boustrophedon motion. In contrast, cells where obstacles are detected are marked as the value of 2 or a "*real obstacle*". While performing the boustrophedon motion, backtracking points are detected and stored in the backtracking list $bp_{list}$, the model of the workspace is constructed incrementally. The boustrophedon motion ends if the current cell is the critical point.

**Algorithm 2**: Boustrophedon motion algorithm

```
1:  Inputs: s_start, bp_list, model;
2:  Outputs: s_cp, bp_list, model;
3:  s := s_start;
4:  model(s) := 1;
5:  while (s is not critical point) do
6:    if (no obstacle in the north direction) then
7:      y_s := y_s + 1;
8:    else
9:      if (no obstacle in the south direction) then
10:       y_s := y_s - 1;
11:     else
12:       if (no obstacle in the east direction) then
13:         x_s := x_s + 1;
14:       else
15:         x_s := x_s - 1;
16:       endif
17:     endif
18:   endif
19:   endif
20:   model(s) := 1; /* a virtual obstacle /*
21:   if (s is a backtracking point) then
22:     bp_list := bp_list ∪ {s};
23:   endif
24: endwhile
25: s_cp := s;
```

For the first boustrophedon motion, the starting point is the initial position of the robot. At the end of the first boustrophedon motion procedure, where the critical point and the backtracking list are detected, the robot must estimate the best backtracking point as the starting point of the second boustrophedon motion. In general, for a new boustrophedon motion, the starting point must be determined in the boustrophedon list of previous backtracking points. The selection of the best backtracking point can be based on different criteria such as the nearest point to the current critical point using the Euclidean distance, the Manhattan distance, or even the shortest path using a distance propagation algorithm which only already covered cells (*i.e.*, virtual obstacles) are considered to compute the length of the path from the critical point to determine the backtracking points.

After having the starting point of the new boustrophedon motion, the robot has to move from the current position (*i.e.*, the critical point $s_{cp}$ of the current boustrophedon motion) to the starting point of the new boustrophedon motion while avoiding obstacles. The easiest way to achieve the collision-free movement is to get back to the starting point based on the covered cells. However, the movement obtained by this way can be unrealistic looking. Therefore, we propose that A* search algorithm is employed to determine the shortest collision-free path between the critical point and the starting point of the next boustrophedon motion.

**Algorithm 3**: BA* algorithm

```
1: model := ∅;
2: bp_list := ∅;
3: repeat
4:   a) Perform a boustrophedon motion from s_start, return the
     critical point s_cp, the backtracking list bp_list, and the model.
5:   b) Determine the nearest backtracking point as the starting
     point s_start of the next boustrophedon motion from s_cp
     based on the backtracking list bp_list, and the model.
6:   c) Find the shortest collision-free path from s_cp to s_start
     based on A* and model.
7:   d) Robot move from s_cp to s_start based on the path obtained
     from A* algorithm.
8: until (bp_list = ∅);
```

When A* search is applied to find the shortest path from the starting cell to the goal cell on grids [16,17], A* has the ability to expand a cell in one part of the free space and then immediately expand a cell in another part of the free space of the grid to determine the minimal cost function *f(s)*. Therefore, A* search is an offline algorithm. In contrast, our approach employs A* search as an online algorithm because the robot occupies only at the critical point and finds the shortest collision-free path to the candidates of the start points based on only already visited cells which are known to be free ones as they have already been covered by the robot. In other words, at the critical point the robot determines a complete solution based on the previous visited cells before it moves to the selected starting point. On the other hand, in the boustrophedon motion procedure as shown in Algorithm 2, the robot only observes the workspace in four neighboring cells of its position and takes an action to cover an uncovered cell. Therefore, the

boustrophedon motion procedure is an online procedure. Based on the boustrophedon motion procedure and A* search algorithm, we propose the online BA* algorithm as Algorithm 3.

In BA* algorithm, A* search is used to determine the shortest collision-free path from the critical point to the starting point of the next boustrophedon motion. To reduce the path length obtained from A* search, the set of neighboring cells, *nghb(s)*, of cell *s* is considered to be eight neighboring cells instead of four neighboring cells as in the boustrophedon motion procedure.

## IV. EXPERIMENTAL RESULTS

In this section, we describe simulations implemented in Matlab software on computer to evaluate the efficiency of our BA* algorithm.

The first two simulations are implemented to determinate the successful coverage rate of a cleaning robot in a workspace consisting of predefined obstacles. The robot is initially placed at the bottom left corner of the workspace. Fig. 3 and Fig. 4 show the trajectories of these simulations. The point *R* shows the ending point of trajectories. In Fig. 3, twelve boustrophedon motions obtained from the boustrophedon motion procedure and the remaining red paths obtained from A* algorithm are connected to form the trajectory covering the workspace. In Fig. 4, eleven boustrophedon motions obtained from the boustrophedon motion procedure and the remaining red paths obtained from A* algorithm. In these two simulations, the nearest backtracking point of the critical point is determined based on the shortest path using the distance propagation of A* algorithm. The simulation results prove that BA* algorithm succeeds in controlling the robot to cover completely its workspace.

Next, we make simulations to evaluate the length of the path and the number of boustrophedon motions required to cover completely the workspace. We compare the methods of determination of the nearest backtracking point to the current critical point using the Euclidean distance, the Manhattan distance, and the shortest path using the distance propagation of A* search algorithm. The workspaces of these simulations are the same as shown in Fig. 3 and Fig. 4. The robot is initially placed at the bottom left corner of the workspace. The simulation results are shown in Table 1 in which the number in the bracket "()" indicates the number of boustrophedon motions. For the first workspace, the path length and the number of boustrophedon motions are equal for three methods. Meanwhile, if the nearest backtracking point is determined based on A* search, the path length and the number of boustrophedon motion are smallest. Fig. 4 shows the trajectories obtained by using A* search. Fig. 5 and Fig. 6 show the trajectories obtained by using the Euclidean distance and Manhattan distance, respectively, to choose the nearest backtracking point as the starting position of the next boustrophedon motions. It can be seen that at the critical point of the second boustrophedon motion, the staring point of the next boustrophedon motion chosen by A* search is better than that chosen by the other methods.

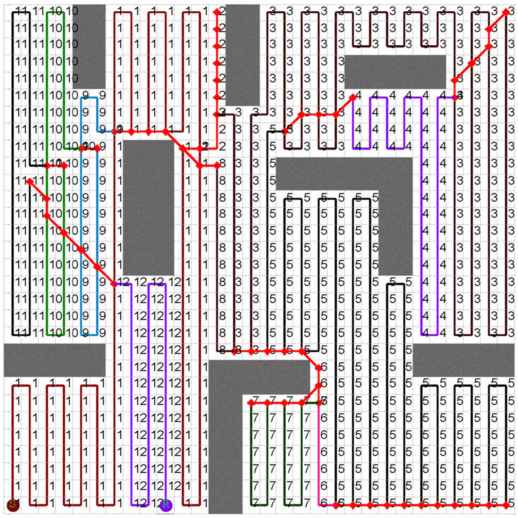

Figure 3. The trajectoty is obtained by BA* algorithm based on A*. Twelve boustrophedon regions are covered and labeled from 1 to 12.

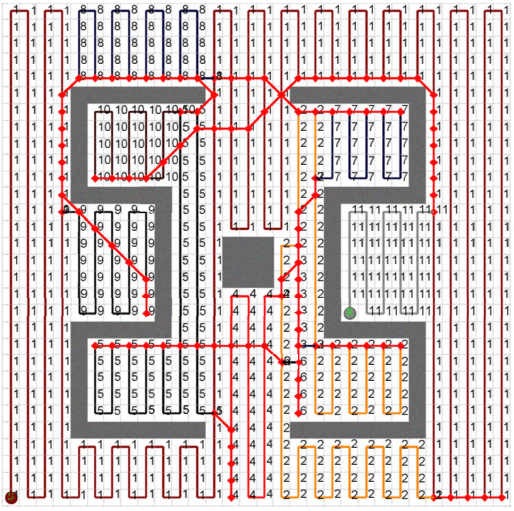

Figure 4. The trajectoty is obtained by BA* algorithm based on A*. Eleven boustrophedon regions are covered and labeled from 1 to 11.

TABLE I. THE COVERED PATH LENGTH

| Distance metric / Workspace | Euclidean distance | Manhattan distance | The shortest path using A* |
|---|---|---|---|
| Fig. 3 | 822.799 (12) | 822.799 (12) | **822.799 (12)** |
| Fig. 4 | 972.8406 (14) | 959.7696 (12) | **920.1127 (11)** |

From simulation results in Table 1, some observations can be remarked as follows:

(*i*) All the simulations terminate after finite steps. It means that BA* algorithm is the complete coverage algorithm.

(*ii*) The length of the covered path and the number of boustrophedon motions depend on the method of determination the nearest backtracking point to the current critical point.

Using the shortest path based on the distance propagation of A* algorithm to choose the starting point of the next boustrophedon motion gives the shortest covered path as well as the smallest number of boustrophedon motions.

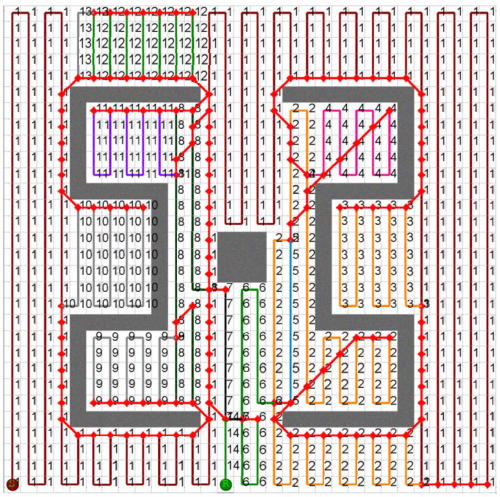

Figure 5. The trajectoty is found based on Euclidean distance.

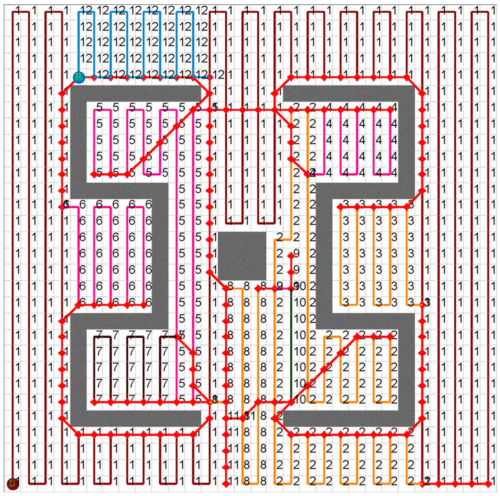

Figure 6. The trajectory is found based on Manhattan distance.

## V. CONCLUSIONS

This paper presents a BA* online complete coverage algorithm for cleaning robots in unknown workspaces based on boustrophedon motions and A* search algorithm. Since a single boustrophedon motion cannot cover the entire workspace, multiple boustrophedon motions need to be performed to ensure the complete coverage. In our approach, while covering an unvisited area based on a boustrophedon motion, the robot detects and stores backtracking points. To cover the next unvisited area, A* search is employed as a backtracking mechanism which guides the robot to the nearest backtracking point. Experimental results prove that BA* algorithm guarantees the complete coverage of the workspace in the finite moving steps of the robot. Moreover, BA* algorithm is a relatively simple but effective approach for the complete coverage problem of cleaning robots. In the future, we plan to extend the proposed approach to the complete coverage problem for multi-cleaning robots.

## ACKNOWLEDGMENT

This work was supported by a grant from the NIPA(National IT Industry Promotion Agency) in 2012. (Global IT Talents Program) and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (2010-0012609).

## REFERENCES

[1] T. Balch, “The case for randomized search,” In Proc. of Sensors and Motion, IEEE International Conference on Robotics and Automation, San Francisco, CA , May 2000.

[2] T. Palleja, M. Tresanchez, M. Teixido, J. Palacin, "Modeling floor-cleaning coverage performances of some domestic mobile robots in a reduced scenario," Robotics and Autonomous Systems, vol. 58, no. 1, pp. 37-45, 2010.

[3] M. Dlouhy, F. Brabec, P. Svestka, “A Genetic Approach to the Cleaning Path Planning Problem,” In Proc. of the16th European Workshop on Computational Geometry, Eilat, Israel, March 2000.

[4] S. X. Yang, C. Luo, “A Neural Network Approach to Complete Coverage Path Planning,” IEEE Trans. on Systems, Man, And Cybernetics—Part B: Cybernetics, vol. 34, no. 1, pp. 718-724, 2004.

[5] H. Choset, P. Pignon, "Coverage Path Planning: The Boustrophedon Cellular Decomposition," In Proc. of International Conference on Field and Service Robotics, Canberra, Australia, 1997.

[6] E.U.Acar, H. Choset, A.A. Rizzi, P.N. Atkar, D. Hull, "Morse Decompositions for Coverage Tasks," The International Journal of Robotics Research, vol.21, no.4, pp. 331-344, 2002.

[7] R. Mannadiar, I. Rekleitis, “Optimal Coverage of a Known Arbitrary Environment,” In Proc. of IEEE International Conference on Robotics and Automation, Alaska, USA, 2010, pp. 5525-5530.

[8] Y.Gabriely, E.Rimon, “Spanning-tree based coverage of continuous areas by a mobile robot,” Annals of Mathematics and Artificial Intelligence, vol.31, no. 4, pp.77-98, 2001.

[9] Y. Gabrielly, E. Rimon, “Spiral-STC: an on-line coverage algorithm of grid environments by a mobile robot”, in Proc. of the IEEE Int. Conf. on Robotics and Automation, Washington, DC, USA, 2002, pp. 954-960.

[10] E. González, P. Aristizábal, and M. Alarcón, “Backtracking Spiral Algorithm: A Mobile Robot Region Filling Strategy,” in Proc. ISRA 2002, 2002, pp. 261-266.

[11] E. González, O. Álvarez, Y. Díaz, C. Parra, C. Bustacara, “BSA: A Complete Coverage Algorithm,” in Proc. of the IEEE Int. Conf. on Robotics and Automation, Barcelona, Spain, 2005, pp. 2040-2044.

[12] Z. Chibin, W. Xingsong, D. Yong, “Complete Coverage Path Planning Based on Ant Colony Algorithm,” In Proc. of the 15th International conference on Mechatronics and Machine Vision in Practice, Auckland,New-Zealand, 2008. pp. 357-361.

[13] S. Koenig, Y. Liu, "Terrain Coverage with Ant Robots: A Simulation Study," In Proc. of the International Conference on Autonomous Agents, Montreal, Quebec, Canada, 2001, pp. 600-607.

[14] J. Palacín, T. Palleja, I. Valgañón, R. Pernia, J. Roca, "Measuring coverage performances of a floor cleaning mobile robot using a vision system," In Proc. of the IEEE Int. Conf. on Robotics and Automation, Barcelona, 2005, pp. 4236-4241.

[15] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

[16] A. Nash, K. Daniel, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," In Proc. of the AAAI Conference on Artificial Intelligence, 2007, pp. 1177–1183.

[17] P. Yap, "Grid-Based Path-Finding," Lecture Notes in Artificial Intelligence, vol. 2338, pp. 44-55, 2002.