Journal of Intelligent & Robotic Systems

with a special section on Unmanned Systems

ISSN: 0921-0296 (Print) 1573-0409 (Online)

Description

The Journal of Intelligent and Robotic Systems bridges the gap between theory and practice in all areas of intelligent systems and robotics. It publishes original, peer reviewed contributions from initial concept and theory to prototyping to final product development and commercialization.

On the theoretical side, the journal features papers focusing on intelligent systems engineering, distributed intelligence systems, multi-level systems, intelligent control, multi-robot systems, cooperation and coordination of unmanned vehicle systems, etc.

On the application side, the journal emphasizes autonomous systems, industrial robotic systems, multi-robot systems, aerial vehicles, mobile robot platforms, underwater robots, sensors, sensor-fusion, and sensor-based control. Readers will also find papers on real applications of intelligent and robotic systems (e.g., mechatronics, manufacturing, biomedical, underwater, humanoid, mobile/legged robot and space applications, etc.).

hide

Browse Volumes & Issues

Impact Factor	Available
1.512	1988 - 2017
Volumes	**Issues**
88	273
Articles	**Open Access**
2,517	47 Articles

Latest Articles

Editorial

From the Editor-in-Chief

Kimon P. Valavanis (October 2017)

» Download PDF (518KB)

OriginalPaper

High Precision Stabilization of Pan-Tilt Systems Using Reliable Angular Acceleration Feedback from a Master-Slave Kalman Filter

Sanem Evren, Firat Yavuz, Mustafa Unel (October 2017)

OriginalPaper

Decentralized Control of Harmonic Drive Based Modular Robot Manipulator using only Position Measurements: Theory and Experimental Verification

Bo Dong, Keping Liu, Yuanchun Li (October 2017)

» See all articles

Stay up to Date

Article abstracts by RSS

Register for journal updates

Find a Volume or Issue

Find

Share

Journal Title
Journal of Intelligent & Robotic Systems

Coverage
Volume 1 / 1988 - Volume 88 / 2017

Topics
» Control, Robotics, Mechatronics
» Electrical Engineering
» Artificial Intelligence (incl. Robotics)
» Mechanical Engineering

Journal of Intelligent & Robotic Systems
All Volumes & Issues

Volume 87, Issue 2, August 2017

ISSN: 0921-0296 (Print) 1573-0409 (Online)

In this issue (10 articles)

JOURNAL SEARCH

Search Terms: 0921-0296

Total journals found: 1

THE FOLLOWING TITLE(S) MATCHED YOUR REQUEST:

Journals 1-1 (of 1)

FORMAT FOR PRINT

JOURNAL OF INTELLIGENT & ROBOTIC SYSTEMS

Monthly ISSN: 0921-0296

SPRINGER, VAN GODEWIJCKSTRAAT 30, DORDRECHT, NETHERLANDS, 3311 GZ

Coverage

Science Citation Index Expanded

Current Contents - Engineering, Computing & Technology

Journals 1-1 (of 1)

FORMAT FOR PRINT

Search Terms:

Search type: Title Word

Database: Master Journal List

SEARCH

J Intell Robot Syst
DOI 10.1007/s10846-017-0485-x

B-Theta*: an Efficient Online Coverage Algorithm for Autonomous Cleaning Robots

SeungYoon Choi · SeungGwan Lee · Hoang Huu Viet · TaeChoong Chung

Received: 1 September 2016 / Accepted: 15 January 2017

Abstract We propose a novel approach to deal with the online complete-coverage task of cleaning robots in unknown workspaces with arbitrarily-shaped obstacles. Our approach is based on the boustrophedon motions, the boundary-following motions, and the Theta* algorithm known as B-Theta*. Under control of B-Theta*, the robot performs a single boustrophedon motion to cover an unvisited region. While performing the boustrophedon motion, if the robot encounters an obstacle with a boundary that has not yet been covered, it switches to the boundary mode to cover portions along the obstacle boundary, and then continues the boustrophedon motion until it detects an ending point. To move to an unvisited region, the robot detects backtracking points based on its accumulated knowledge, and applies an intelligent backtracking mechanism thanks to the proposed Theta* for multi-goals in order to reach the next starting point. Complete coverage is achieved when no starting point exists for a new boustrophedon motion. Computer simulations and experiments on real workspaces show that our proposed B-Theta* is efficient for the complete-coverage task of cleaning robots.

Keywords Boustrophedon motion · Boundary-following motion · Cleaning robot · Complete coverage · Theta* algorithm

S. Y. Choi · T. C. Chung (✉)
Department of Computer Engineering, Kyung Hee University, 1732, Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 446-701, Korea
e-mail: tcchung@khu.ac.kr

S. Y. Choi
e-mail: sychoi84@khu.ac.kr

S. G. Lee
Humanitas College, Kyung Hee University, 1732, Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 446-701, Korea
e-mail: leesg@khu.ac.kr

H. H. Viet (✉)
Department of Information Technology, Vinh University, 182-Le Duan, Vinh City, Nghe An, Vietnam
e-mail: viethh@vinhuni.edu.vn

1 Introduction

The complete-coverage task of service robots has become the main concern of the robotics research community due to its wide range of applications for lawn mowing, mine hunting, floor cleaning, harvesting, etc. [5, 21], as well as its challenges in terms of energy saving, time saving, and the ability to deal with both known and unknown workspaces. In known workspaces, the complete-coverage task is actually the path-planning task of a robot to be able to visit all the reachable positions in the workspace while minimizing the coverage path length. A number of existing methods address the coverage task in known workspaces, namely genetic algorithms [10], neural networks [21, 32], cellular decomposition [1, 4, 7,

Published online: 27 January 2017

22], spanning trees [13, 14], spiral filling paths [15, 16], and the ant colony method [3, 18]. However, these methods are offline algorithms, which require a full map or a complete prior knowledge of the workspace. By contrast, in real-world scenarios the map of a workspace is usually unknown to the robot, which means that the locations, sizes, and shapes of obstacles are also unknown. Even when the robot has a map, it must know exactly its initial position and heading angle in the map. This requirement causes difficulties and inconveniences for practical use. Moreover, maps do not always reflect reality, considering the presence of objects that usually move, such as chairs, stools, rocks, bikes, cars, etc. Therefore, in unknown workspaces, robots require online coverage algorithms instead of offline coverage algorithms, which are insufficient when a map is unavailable.

From this point of view, we focus on developing a technique for the online complete-coverage task in unknown settings. This technique addresses most of the disadvantages of the previous works. The online complete-coverage problem has been considered in recent years by several works. Oh et al. [24] propose a triangular-cell-based map representation to improve the rectangular-cell-based map, in which the navigation directions of the robot is increased from eight to twelve. This approach reduces the navigation path length while still fulfilling the coverage mission. However, the drawbacks of this technique are the difficulties in deciding the optimal size of the triangular cells, and that the backtracking path is still long because of the resulting broken-line path instead of a smooth line-of-sight path. Therefore, a triangular-cell-based map representation is inadequate for real-world scenarios. Wong et al. [30, 31] propose a topological coverage algorithm to deal with the complete-coverage task in unknown environments. The algorithm uses natural landmarks such as walls or corridors in the environment to construct a planar graph representing a decomposition of reachable surfaces into subregions, so that each sub-region can be covered by a zigzag pattern. When the robot finishes covering the current sub-region, a breadth-first search on the planar graph is performed to find the closest uncovered sub-region. The robot then follows the edge to reach the selected sub-region. This method does not provide an optimal coverage path, because the backtracking path linking the sub-regions is based only on the edges of the graph, ignoring the shortcut paths inside the sub-regions. In other words, this method is inefficient for the online complete-coverage problem, particularly when the workspace is large. Viet et al. [29] propose a BA* algorithm based on boustrophedon motions and the A* search [17]. At each ending point the robot plans k paths to all k backtracking points, and then chooses the shortest collision-free path as the backtracking path. Therefore, BA* is inefficient in terms of time complexity.

Existing works have always assumed that a robot can clean all the dirt at the edges of obstacles or the boundaries of workspaces. This assumption is not satisfied in the real world when existing methods pay much attention to the coverage task in an object-free zone. The hit-and-turn mechanism of a robot cannot cover edges completely as a man does when cleaning a floor or mowing a lawn. Meanwhile, the necessity of complete coverage is the highest criterion. This criterion is serious in a common example where an autonomous robot cleans the floor for a year and always leaves dirt at some specific locations. The boustrophedon-based methods [1, 4, 7, 24, 29, 31] cannot satisfy this criterion either, because they cover the workspace with simple back-and-forth motions. More specifically, they always provide stair-shaped coverage zones around curved edges or curved boundaries. This neglect is unacceptable, particularly for cleaning robots when the dirt usually accumulates at the edges and boundaries of obstacles. Another consideration is that most state-of-the-art vacuum cleaners in the market today are designed with only a few bump sensors to gather information from the workspace, making them even less suitable for online methods. Therefore, cleaners are usually equipped with the limited algorithm of randomly going straight and then turning, hopefully to achieve complete coverage over a sufficiently long time period [25, 27]. In this paper we propose a complete and efficient approach to deal with the online complete-coverage task of autonomous cleaning robots in completely unknown workspaces with arbitrarily-shaped obstacles. This approach is based on the boustrophedon motions, the boundary-following motions, and the Theta* algorithm called B-Theta*. The boustrophedon cleaning mode drives the robot to go back-and-forth along a path similar to that of an ox plowing a field [7]. Our approach, applied to an actual cleaning robot, needs only a few touch sensors to recognize obstacles correctly during the robot's navigation time.

This approach requires two regular conditions: the workspace must be closed, and the accessible regions must be connected and reachable by the robot from any initial position. Under control of B-Theta*, the robot takes three steps to accomplish its coverage mission. First, the robot works in the boustrophedon mode, performing a boustrophedon motion to cover an unvisited area. While in this mode, if the robot encounters an obstacle with a boundary that has not yet been covered, the robot switches to the boundary mode to cover the portions along the obstacle boundary, and then continues the boustrophedon motion until it detects an ending point. Second, the robot recalls its knowledge so far to determine the backtracking points, and then plans the shortest backtracking path to the backtracking points using the proposed Theta* for multi-goals. Finally, the robot takes this backtracking path to perform the next boustrophedon motion. The coverage task finishes when the list of backtracking points is empty. The proposed approach guarantees that the workspace is covered completely by the sequential and continuous operations of the robot, and that the accessible area at the obstacle edges and the workspace boundary are cleaned efficiently. Computer simulations and experiments on real workspaces show that our B-Theta* is efficient for the complete-coverage task of cleaning robots in terms of the coverage rate and the coverage path length in unknown workspaces with arbitrarily-shaped obstacles.

The rest of this paper is organized as follows: Section 2 gives the background of our approach. Section 3 presents our approach to the online complete-coverage problem. Section 4 discusses the simulations and evaluations. Section 5 describes our experiments on *iRobot Create* in real workspaces. Finally, Section 6 provides our conclusions.

2 Background

2.1 Boustrophedon Motion Approaches

The trapezoidal decomposition is the most widely-known method of the boustrophedon motions for solving the coverage path-planning problem for cleaning robots [20]. This method is an exact cellular decomposition technique in which the accessible area of the robot is decomposed into non-overlapping cells. The cells are formed via three types of events: IN, OUT, and MIDDLE. These events happen while a vertical line, called a *slice*, sweeps from left to right through a bounded workspace. This method assumes that all obstacles in the workspace are polygonal. An IN event is formed when the slice intersects a vertex of an obstacle, in which the current cell is closed and two new cells are opened. An OUT event is formed when the slice intersects a vertex of an obstacle, in which the two current cells are closed and one new cell is opened. A MIDDLE event is formed when the slice intersects a vertex of an obstacle, in which the current cell is closed and a new cell is opened. With such a cellular decomposition, each cell of the trapezoidal decomposition is either a trapezoid or a triangle. As a result, the coverage in each cell can be achieved with a boustrophedon motion, as shown in Fig. 1a.

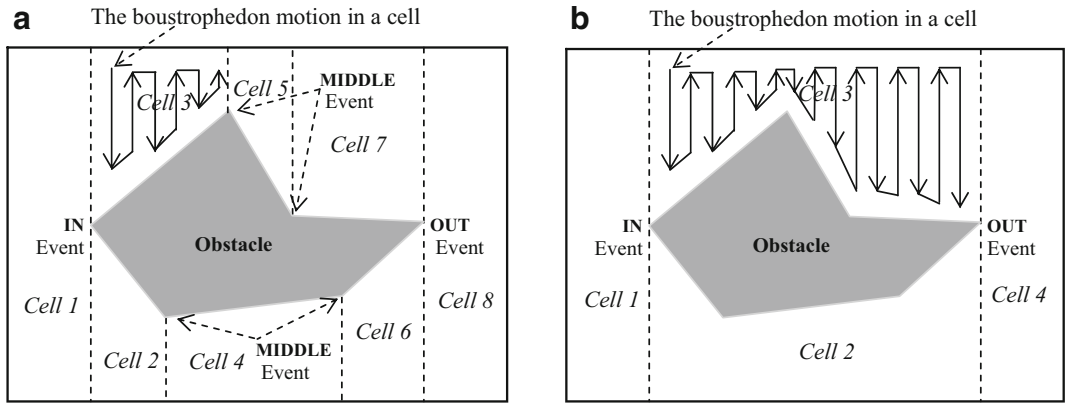


Fig. 1 **a** The trapezoidal decomposition of a workspace; **b** The boustrophedon decomposition of a workspace

If each cell is represented by a vertex and the adjacent cells are connected to form the edges of a graph, then the coverage-path planning problem is reduced to determining a walk through the graph that each vertex has visited at least once, i.e., the traveling salesman problem. Unfortunately, the trapezoidal decomposition generates numerous small cells that lengthen the coverage path. Repartitioning of the cells to achieve a shorter coverage path is the main improvement of the boustrophedon cellular decomposition (BCD) approach [4, 7]. The BCD approach defines cells with only IN and OUT events to reduce the number of decomposed cells, as depicted in Fig. 1b. BCD is an offline approach in which the coverage task of the robot is decomposed into two stages. The first stage is the determination of a walk through the adjacent graph, whose vertices represent the decomposed cells and whose edges connect the adjacent cells. The second stage is the actual coverage-path planning for the robot using the vertices of the walk. The robot starts at the first cell corresponding to the first vertex of the walk. If the cell has not yet been covered, it is swept by a boustrophedon motion; otherwise, it is passed through to the next cell corresponding to the next vertex of the walk. These covering and passing actions are repeated until the cell corresponding to the final vertex of the walk is reached.

BCD is an exact cellular decomposition method for coverage-path planning. The coverage path is planned only when the map of the workspace is known beforehand; thus, BCD cannot be applied for the coverage task of cleaning robots in unknown workspaces. Furthermore, BCD is not complete coverage, because the boustrophedon motions cannot cover the portions along the curved boundaries of the workspace and obstacles.

2.2 Theta* Algorithm

Theta* [8, 23] is a variant of the A* search [17] for any-angle path planning. Given a starting vertex and a goal vertex, Theta* also defines a *heuristic estimate* f-value $f(s) = g(s) + h(s)$, which is an estimate of the shortest path length from the starting vertex via vertex s to the goal vertex, where $g(s)$ is the length of the shortest path from the starting vertex to vertex s that has been found so far, and $h(s)$ is an estimate of the distance from vertex s to the goal vertex. Inherited from the A* search, Theta* is optimal if $h(s)$ is an *admissible heuristic*, that is, if $h(s)$ never overestimates the cost to reach the goal [19, 26].

For grids, if the heuristic $h(s)$ is the straight-line distance from vertex s to the goal vertex, then the A* search guarantees an optimal solution path for the robot from the starting vertex to the goal vertex [8, 23, 33]. However, the solution path is constrained by the limited number of connections between the adjacent vertices. Therefore, the solution path is not equivalent to the true shortest path. Moreover, the robot usually performs a large number of heading changes while moving along the solution path. Unlike the A* search, in which the parent must be a successor, Theta* allows the parent of a vertex to be any vertex, thereby improving the solution path thanks to a smoothing process. Theta* inserts the smoothing task by checking the line-of-sight paths into the iterations of the searching process. Theta* takes a starting vertex s_s and a goal vertex s_g as inputs to find the true shortest path connecting them through a sequence of vertices. As a version of the A* search, examining all of the candidate vertices that might be able to construct the shortest path requires that Theta* maintains and updates the three lists of vertices during the searching progress: *open*, *closed*, and *parent*. The *open* list stores all of the detected neighboring vertices of the vertices that are already expanded. Any vertex in the *open* list is eliminated from the list if the vertex is expanded. The *closed* list memorizes all vertices that are already expanded. By referring to this list, Theta* guarantees that a vertex is not examined twice. The *parent* list stores the parent-child relations between the vertices. This list is used to construct the solution path between two input vertices when Theta* ends. Theta* is described in detail in [8, 23].

3 Proposed B-Theta* Algorithm

This section describes our method to solve the online complete-coverage problem of cleaning robots in unknown workspaces with arbitrarily-shaped obstacles.

3.1 Representing the Robot

The cleaning robot and obstacles in its workspace are three-dimensional objects, but the movement of the robot is constrained in a two-dimensional plane $\mathcal{R}^2$. Therefore, without loss of generality, the robot and

obstacles can be projected onto the ground and considered as two-dimensional objects in the workspace $\mathcal{W} = \mathcal{R}^2$. Assuming that the robot is modeled by a circle with a radius r, and that the robot is supported by a precise localization system for determining its location in the workspace $\mathcal{W}$, the *configuration* q of the robot is defined as

$$q = [x, y, \theta]^T, \tag{1}$$

where (x, y) is the center position, and θ is the heading angle of the robot in the fixed frame $F_w = (OXY)$ of the workspace $\mathcal{W}$, as shown in Fig. 2 [11, 20]. In the current configuration $q = [x, y, \theta]^T$, if the robot rotates at an angle α, the next configuration $q' = [x', y', \theta']^T$ is determined as

$$q' = [x', y', \theta']^T = [x, y, \theta + \alpha]^T. \tag{2}$$

If $\alpha > 0$, the robot rotates counterclockwise, and vice versa. If the robot moves forward at some distance d, the next configuration $q' = [x', y', \theta']^T$ is determined as

$$q' = [x', y', \theta']^T = [x + dcos(\theta), y + dsin(\theta), \theta]^T. \tag{3}$$

When the robot works in $\mathcal{W}$, it cannot access all the regions of $\mathcal{W}$ because of obstacles. A configuration q of the robot is said to be valid if the robot can access the position specified by q in $\mathcal{W}$. The set of all valid configurations of the robot is defined as its accessible region in $\mathcal{W}$. The configuration q in $\mathcal{W}$ that the robot cannot access is an obstacle. We also consider the unreachable space beyond the workspace boundaries as obstacles.

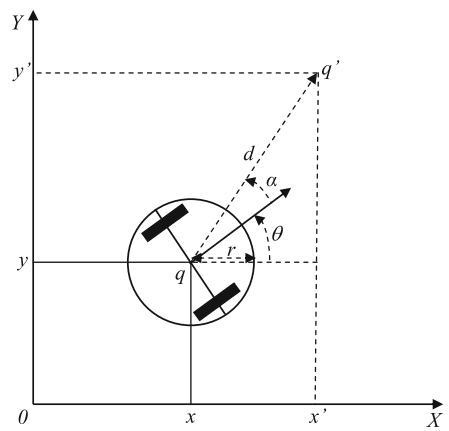


Fig. 2 The configuration of the robot in the fixed frame $F_w = (OXY)$ of workspace $\mathcal{W}$

3.2 Constructing the Coverage Mechanism

The coverage mechanism is constructed based on boundary-following motions and boustrophedon motions. The boundary-following motions allow the robot to cover the portions along the obstacle boundaries. The boustrophedon motions allow the robot to cover the remaining accessible regions in the workspace.

3.2.1 Constructing Boundary-Following Motions

The boustrophedon-based approaches [1, 7, 20, 24, 30, 31] for the complete-coverage problem in workspaces with arbitrarily-shaped obstacles do not provide complete coverage, because the robot moves parallel to the coordinate axes of the fixed framework F_w while performing the coverage task; thus, the portions along the obstacle boundaries are not covered, and dirt is often concentrated in these portions. Therefore, covering along the obstacle boundaries becomes important in the cleaning robot application. From this point of view, we insert boundary-following motions into boustrophedon motions to overcome the shortcomings of the solutions using boustrophedon motions. The boundary-following motion has been used for the path planning problem in Bug algorithms [6], but it has not been implemented for cleaning robots. In our approach, while performing a single boustrophedon motion in covering an unvisited region, if the robot reaches the boundary of an obstacle or the workspace and the boundary has not yet been covered, the robot switches to the boundary mode to cover the portions along this boundary. In the boundary mode, the robot can cover the boundary in the left-detection boundary mode (LDBM), which goes along the boundary so that the boundary is always on the left of the robot, or the right-detection boundary mode (RDBM), which is the opposite of the LDBM. The robot needs to determine whether to use the LDBM or RDBM as soon as it encounters the obstacle in order to reduce the turning angle and coverage path length. Specifically, the robot uses its memory to check the positions at its left and right sides. If the left side has been covered and the

right side has not yet been covered, the robot switches to the RDBM. If the left side has not yet been covered and the right side has been covered, the robot switches to the LDBM. Otherwise it chooses randomly between the LDBM and RDBM.

In the boundary mode, the robot needs to memorize the covered boundary of obstacles to avoid duplicate coverage. To do so, we decompose the boundary-following motion into steps and store positions of steps in a list called the obstacle boundary model $\mathcal{B}$. Each element of model $\mathcal{B}$ is a square cell bounding the robot that is generated when the robot occupies a new position by moving forward along the obstacle boundary a distance equal to its radius. The boundary model $\mathcal{B}$ is thus constructed incrementally while the robot covers the boundary of obstacles. Algorithm 1 describes the flowchart of the LDBM. Specifically, the robot uses model $\mathcal{B}$ to recognize the covered positions, and uses only the sensors on its front and left sides to identify the obstacle. In each iteration of the algorithm the robot keeps turning left as long as the left sensor identifies the obstacle, and then moves forward along the obstacle boundary a distance equal to its radius. The square cell bounding the robot is then generated and added to model $\mathcal{B}$. If the robot hits an obstacle at the front, it turns right so that the left sensor keeps detecting the obstacle. The algorithm terminates when the robot arrives at the position at which it has switched to the LDBM. The RDBM algorithm enables the robot to perform the boundary mode in the opposite direction of the LDBM.

Figure 3 shows an example of the boundary mode, where each circle represents the robot and the straight line attached to the dot at the circle's center represents the moving direction of the robot. Moreover, two straight lines attached to the dot at the circle's center indicate that the robot has two configurations at the same position. The first configuration is the same direction as that of the previous configuration. The second configuration is the direction pointing to the next position. Starting at position $S1$, the robot performs the first boustrophedon motion until it encounters the left obstacle, as shown in Fig. 3a. Given that both the left and right sides of the robot have not yet been covered, the robot randomly switches to the RDBM. When the robot encounters the position at which it has switched to the RDBM, it changes direction and then continues the boustrophedon motion until it reaches the boundary of the workspace, as shown in Fig. 3a. Similarly, both the left and right sides have not yet been covered, so the robot randomly switches to the LDBM. After covering the portions along the boundary of the workspace, the robot switches back to the boustrophedon mode to cover the accessible area until it encounters the right obstacle, as shown in Fig. 3b. Given that the left side has not yet been covered and the right side has been covered, the robot switches to the LDBM.

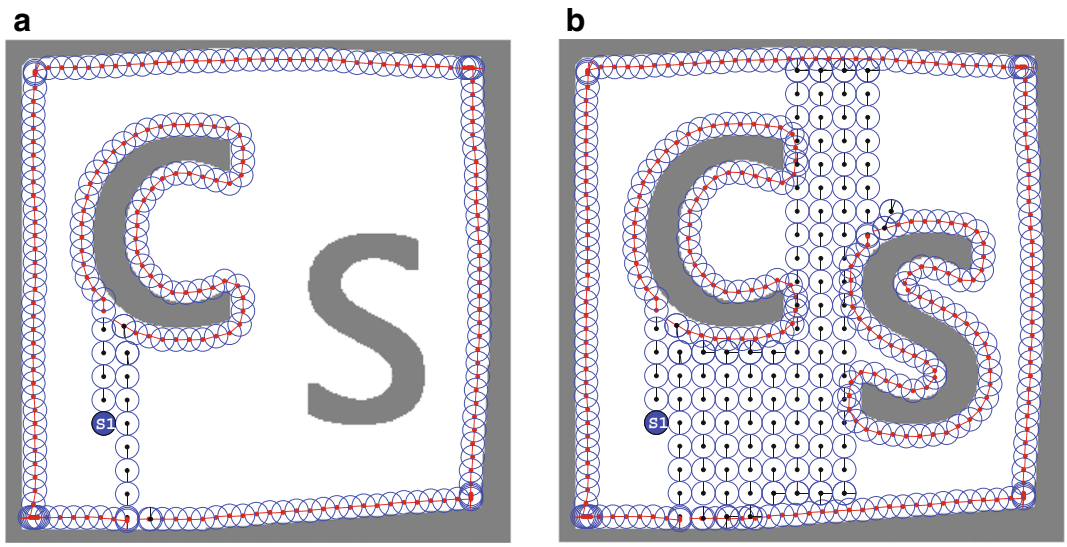
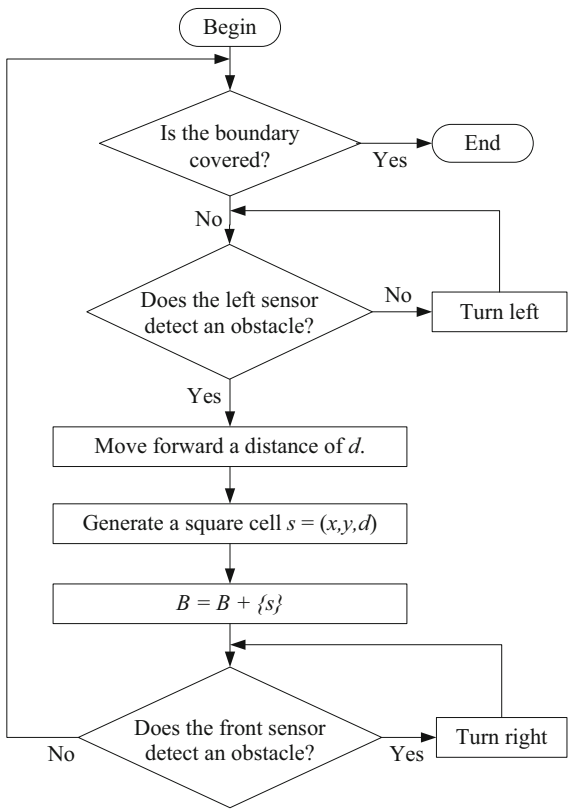


Fig. 3 The right-detection boundary mode (RDBM) and the left-detection boundary mode (LDBM)

3.2.2 Constructing Boustrophedon Motions

To achieve an online complete coverage, our approach is to construct non-overlapped regions incrementally so that their union is the remaining accessible part of the workspace. Each region is covered by a boustrophedon motion, and is therefore called a *boustrophedon region*. The smallest boustrophedon region is a square cell whose side length is equal to the robot's diameter.

Algorithm 1 The flowchart of the left-detection boundary mode



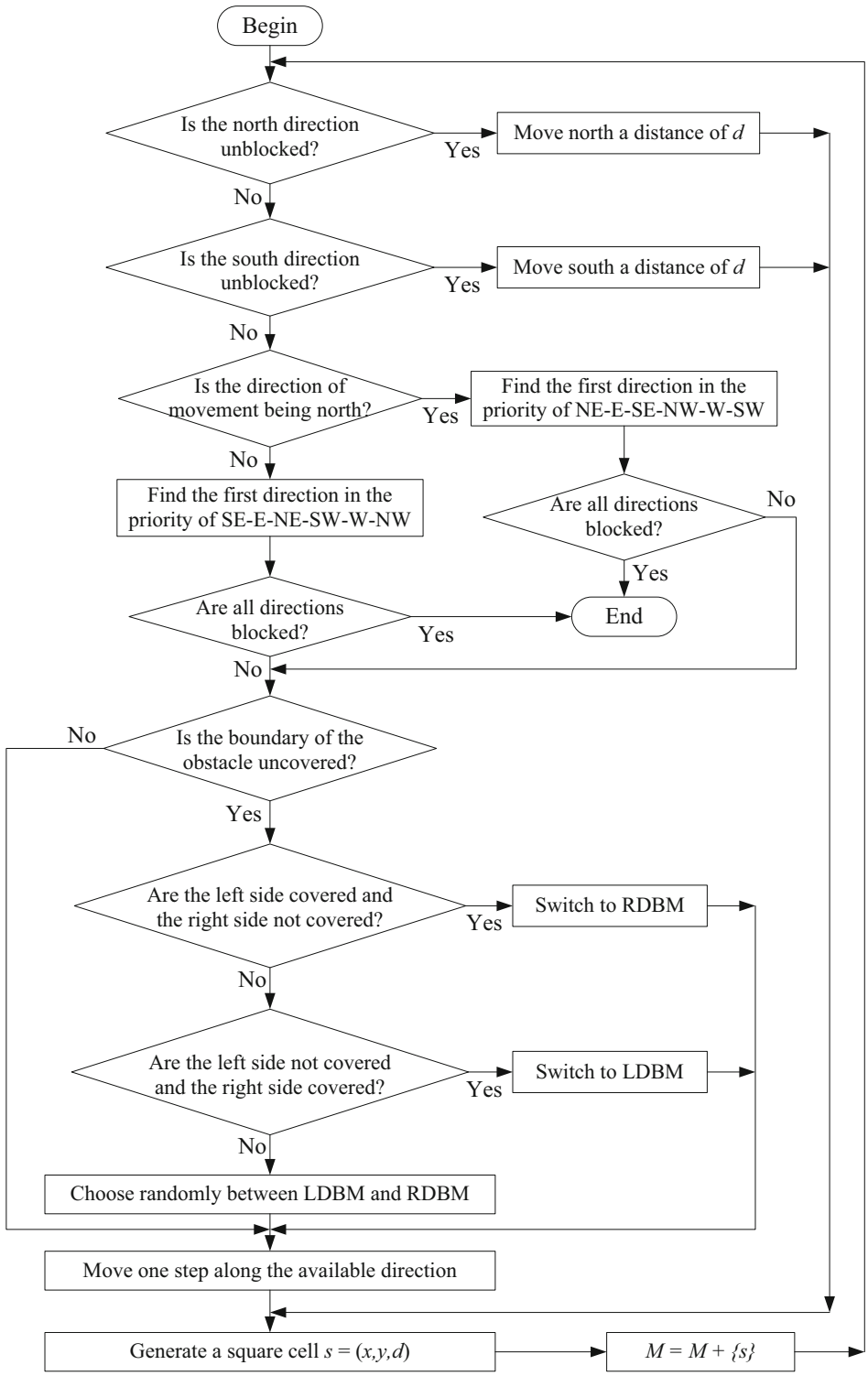
In the boustrophedon mode, the robot mimics the plow tracks in a field. Typically, the robot goes north (or south) until it detects an obstacle, turns around, traverses the next adjacent column of the accessible area, and so on. We compose the boustrophedon path with non-overlapping cells to memorize the covered positions. Each cell is a square with a side length of the robot's diameter d centered on the boustrophedon path. The set of these cells is stored in a list called model $\mathcal{M}$. A new cell s of model $\mathcal{M}$ is generated when the robot occupies an uncovered position while going to a cell along north (N), east (E), south (S), or west (W) at a distance of d, or along south-east (SE), north-east (NE), north-west (NW), or south-west (SW) at a distance of $\sqrt{2}d$. In other words, $\mathcal{M}$ represents a partial grid on the workspace, and is constructed incrementally as the robot explores the workspace given by

$$\mathcal{M} = \mathcal{M} \cup \{s\}, \tag{4}$$

where s is a square cell bounding the robot. $\mathcal{M}$ is used to build a backtracking mechanism that guides the robot to the next uncovered region while avoiding obstacles.

Algorithm 2 is used for the robot to perform a boustrophedon motion. In this algorithm, the robot uses the sensors on its front, left, and right sides to detect obstacles, and uses its memory to identify the covered cells. In other words, the robot checks a *blocked position*, which is either an obstacle or a covered cell, by using not only its sensors, but also its knowledge accumulated in models $\mathcal{M}$ and $\mathcal{B}$. At any position in the boustrophedon mode, the robot moves only one step along the direction leading to an uncovered position. Specifically, if the robot moves north, it keeps checking north and moves forward in iterations until it encounters a blocked position. If the robot moves south, the north cell is already covered by checking its memory, so it keeps checking south and moves forward in iterations until it encounters a blocked position. When the robot encounters a blocked position while moving north, it uses its memory to check blocked directions, including S, NW, W, and SW (or S, NE, E, and SE), and uses its sensors to check an unblocked direction, prioritizing NE, E, or SE (or NW, W, or SW) to find an available direction. In contrast, when the robot encounters a blocked position while moving south, it uses its memory to check blocked directions, including N, SW, W, and NW (or N, SE, E, and NE), and uses its sensors to check an unblocked direction, prioritizing SE, E, or NE (or SW, W, or NW) to find an available direction. In addition, if the robot hits an obstacle while moving north or south, it has to check whether the boundary of the obstacle is uncovered by recalling the memorized boundary model $\mathcal{B}$. If the boundary is uncovered, the robot switches to the boundary mode to cover the portions along the boundary, and then moves to an uncovered cell adjacent to the last cell before switching to the boundary mode.

Algorithm 2 The flowchart of the boustrophedon motion algorithm



The mechanism described in Algorithm 3 enables the robot to construct a boustrophedon path until it reaches an *ending point*, which is a cell whose neighboring cells are all blocked. The robot's sensors are incapable of returning information regarding an occupied location; thus, when the robot moves a step of length equal to d or $\sqrt{2}d$ to the next uncovered position, the next valid configuration $q' = [x', y', \theta']^T$ is determined by Eqs. 2 or 3.

Figure 4a shows an example of a single boustrophedon motion constructed by Algorithm 2. The robot starts at position $S1$ to perform the first boustrophedon motion. While performing the boustrophedon motion, the boundary mode is inserted into the boustrophedon mode when the robot encounters the left obstacle, the boundary of the workspace, and the right obstacle at the first, second, and tenth vertical line, respectively. The robot finishes the first boustrophedon when the ending point $E1$ is detected. Figure 4b shows the boustrophedon path, $\mathcal{M}$, and $\mathcal{B}$ obtained after the robot finishes the first boustrophedon motion.

3.3 Constructing the Backtracking Mechanism

The backtracking mechanism is designed to control the robot from the ending point of the current boustrophedon motion to the next starting point of an uncovered region while avoiding obstacles. The backtracking mechanism consists of determining the next starting point of an uncovered region, and the shortest backtracking path from the ending point to the next starting point.

3.3.1 Determining the Next Starting Point

In most cases the robot cannot cover a workspace with obstacles completely by a single boustrophedon motion. Constructing the coverage path that covers an unvisited region by a boustrophedon motion and then moving to another necessitates that the *starting point* for the next boustrophedon motion is determined. Considering that the workspace is unknown in advance to the robot, the starting point of the next boustrophedon motion must be detected from positions that have been covered. This indicates that a starting point is a *backtracking point* from which the robot can start a future boustrophedon motion. Basically, a backtracking point can be any element of the model $\mathcal{M}$ for which at least one of its eight neighboring cells is uncovered. If so, numerous backtracking points that can become the potential starting points must be maintained, making the decision of the best candidate inefficient. In addition, many backtracking points on the edges of the covered regions can result

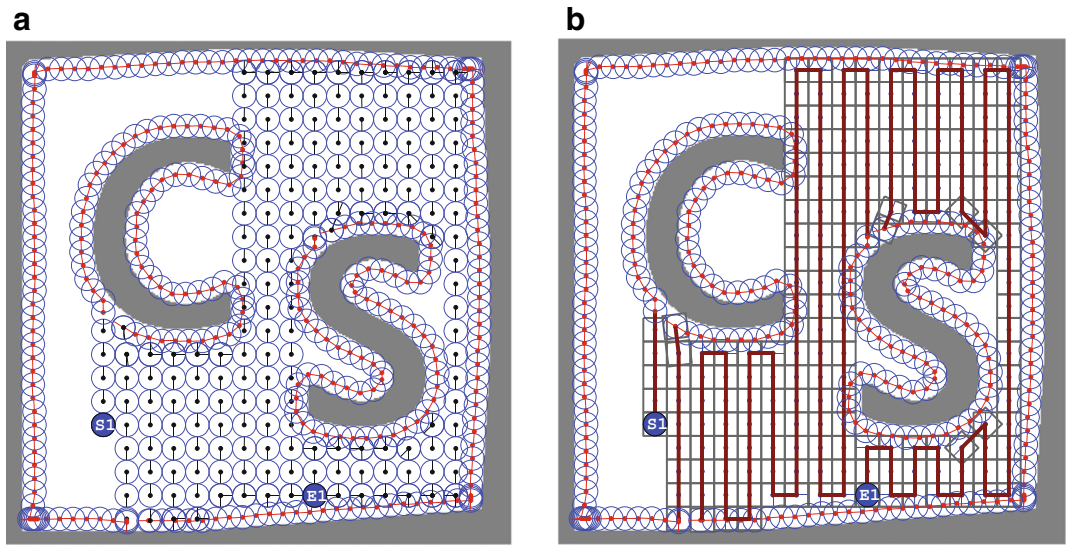


Fig. 4 **a** The region covered by the boustrophedon motion, where $S1$ and $E1$ are the starting and ending positions of the first boustrophedon motion; **b** The boustrophedon path, $\mathcal{M}$, and $\mathcal{B}$ after the robot has finished the first boustrophedon motion

in many boustrophedon motions, and thus lengthen the coverage path. These shortcomings lead to our proposal to reduce the total number of backtracking points. Our approach uses eight neighboring cells of cell s to determine the existence of the starting point candidates. Figure 5 defines all cases in which cell s is decided to be a backtracking point, where each black cell denotes a *blocked position* and each white cell denotes an *uncovered position*. Each grid cell represents a "*don't care*" status (i.e., it could be blocked or uncovered). Specifically, letting $\mathcal{N}(s) = \{s_1, s_2, \cdots, s_8\}$ be the set of eight neighboring cells of cell s in the eight directions E, NE, N, NW, W, SW, S, and SE, respectively, for any two cells $s_i \in \mathcal{N}(s)$ and $s_j \in \mathcal{N}(s)(i, j = 1, 2, \cdots, 8)$, we define the function

$$b(s_i, s_j) = \begin{cases} 1, & \text{if } (s_i \text{ is uncovered) and } (s_j \text{ is blocked);} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

and the sum function

$$\begin{aligned} \Sigma(s) = {} & b(s_1, s_2) + b(s_1, s_8) + b(s_5, s_4) + b(s_5, s_6) \\ & + b(s_3, s_2) + b(s_3, s_4) + b(s_7, s_6) + b(s_7, s_8), \end{aligned} \tag{6}$$

where each component function $b(s_i, s_j)$ corresponds to each sub-figure in Fig. 5. The cell s is defined to be a backtracking point

$$\text{if } \Sigma(s) \geq 1. \tag{7}$$

Under this condition, the backtracking points are only the cells at the corners of the boustrophedon regions. Because a starting point on the edge of a boustrophedon region creates more fragments than a starting point at the corner of a boustrophedon region, Eq. 7 reduces both the total number of backtracking points and the number of boustrophedon regions. This characteristic is one of the main advantages of our approach, because fewer boustrophedon motions result in a shorter coverage path.

A set of backtracking points must be detected from the accumulated knowledge and stored in a *backtracking list* to determine the next starting point. When the robot arrives at the ending point of the current boustrophedon motion, the backtracking list $\mathcal{L}$ of the robot is determined by

$$\mathcal{L} = \{s | s \in \mathcal{M} \text{ and } \Sigma(s) \geq 1\}, \tag{8}$$

and the next starting point is determined among the candidates in the backtracking list $\mathcal{L}$ by measuring the length of the paths from the ending point to the points in $\mathcal{L}$ on the model $\mathcal{M}$. The candidate that provides the shortest path is chosen for the next starting point. In other words, the starting point s_{sp} of the next boustrophedon motion is defined as

$$s_{sp} = \arg\min_{s_k \in \mathcal{L}} (f(s_{ep}, s_k)), \tag{9}$$

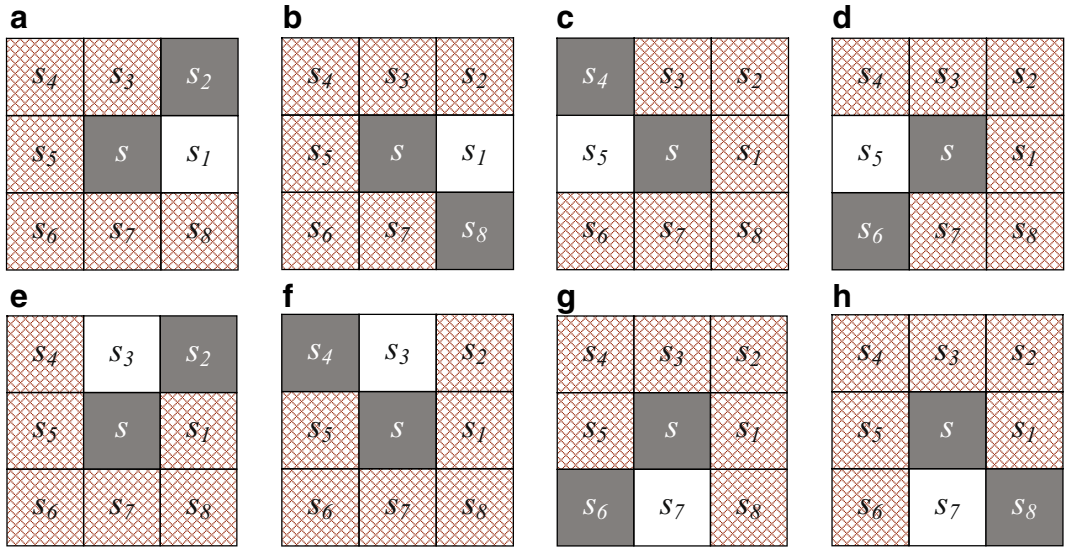


Fig. 5 Conditions of the backtracking points

where $f(s_{ep}, s_k)$ is a distance-based cost function that defines the path length between the ending point s_{ep} and an element $s_k \in \mathcal{L}$.

3.3.2 Planning the Backtracking Path

In this work, planning the backtracking path for the robot is actually determining a collision-free path from the ending point of the current boustrophedon motion to the next starting point. At the ending point, the robot can obtain the backtracking path by backtracking along the old boustrophedon regions until it reaches the next starting point. However, this method not only lengthens the coverage path, but also forms numerous heading changes of the robot. Using the model $\mathcal{M}$ built so far, the backtracking path from the ending point to the next starting point can be determined by the graph search algorithms, such as depth-first search, breadth-first search, and Dijkstra's algorithm [9], or the heuristic algorithms, such as A* search [17], A*PS [2], and Theta* [8, 23]. The graph search algorithms explore the entire search space to find the solution path, whereas the heuristic algorithms do not. As a result, the heuristic algorithms are more efficient than the graph search algorithms. Specifically, given a consistent heuristic $h(s)$ such as the straight-line distance, the A* search expands only the vertices whose total cost $f(s) = g(s) + h(s)$ is less than c^*, which is the cost of an optimal solution path [19, 26]. Both A*PS and Theta* are variants of the A* search that have a smoothing process to improve the quality of the solution path. A*PS smooths the solution path obtained by the A* search in a post-smoothing step, whereas Theta* inserts the smoothing task into the iterations of the searching process. Theta* finds shorter paths than A*PS with a run time comparable to that of the A* search on grids [8, 23]. Therefore, Theta* is employed in this work to plan the backtracking path for the robot.

By default, Theta* plans the shortest collision-free path from a starting point to a single goal point based on the heuristic estimate $f(s) = g(s) + h(s)$. Evidently, if Theta* is applied directly to determine the backtracking path, at each ending point the robot has to determine all k backtracking paths from the ending point to all k candidates in the backtracking list $\mathcal{L}$, and then chooses the shortest one. In other words, Theta* is inadequate to determine the optimal backtracking path because it increases the time complexity. Figure 6a shows a typical example for finding the shortest path from the starting point S to one goal $G_i (i = 1, 2, \cdots, 10)$. Goals G_1, G_2, G_3, and G_4 are

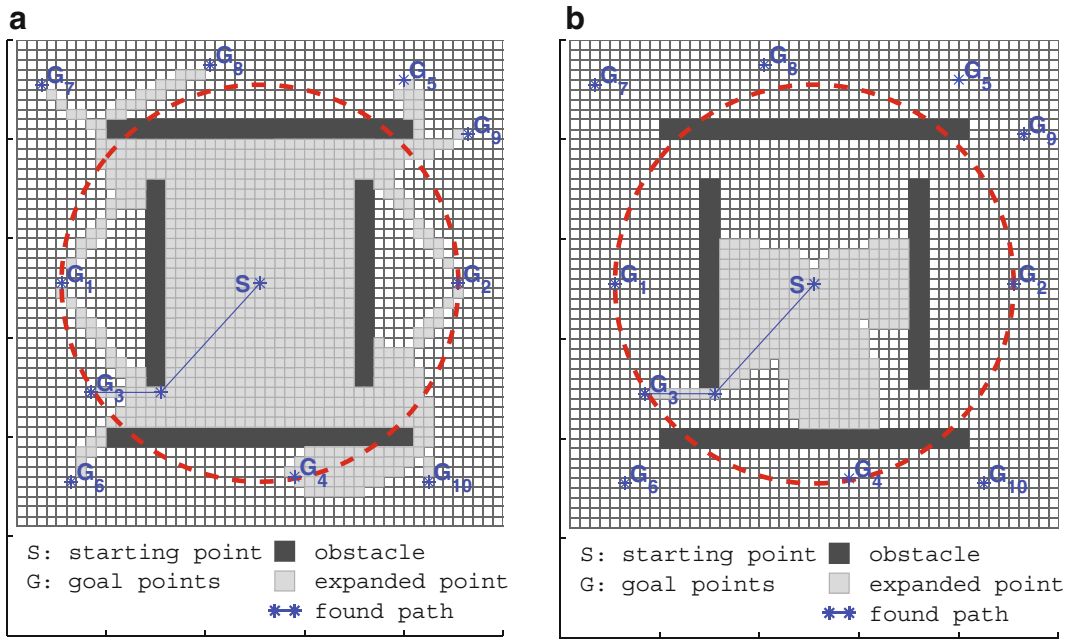


Fig. 6 Solution paths and areas explored by **a** Theta* and **b** Theta* for multi-goals

placed on a circle with center S. To do so, Theta* has to expand a large area to find the paths to all the goals and then choose the shortest path, as shown in the dark area. Obviously, the larger the number of goals or the farther away from the starting point they are, the more inefficient Theta* is. To overcome this limitation, we propose an efficient variant of Theta*, called Theta* for multi-goals, to adapt the purpose of finding the shortest collision free path to a goal among goals. Therefore, we propose an efficient variant of Theta*, called *Theta* for multi-goals*, to adapt the purpose of finding the shortest collision-free path to a goal among goals. We define an estimate of the distance from cell s to the goal cells $s_k \in \mathcal{L}$ as follows:

$$h(s) = \min_{s_k \in \mathcal{L}} (c(s, s_k)), \tag{10}$$

where $c(s, s_k)$ denotes the straight-line distance between two cells s and s_k. Evidently, with the proposed heuristic estimation $h(s)$, Theta* for multi-goals always chooses the goal cell s_k that is nearest to cell s when it explores the search space. As a result, the goal cell s_k that provides the shortest path to the ending point s_{ep} is chosen for the next starting point s_{sp}, or Eq. 9 is exactly met when Theta* for multi-goals ends. Since the backtracking path is constructed based only on the regions that have been accessed by the robot (i.e., the model $\mathcal{M}$ built so far), Theta* for multi-goals never stops without a solution for the backtracking path. Moreover, considering that Theta* for multi-goals ends when a solution is reached, its number of explored points is always less than that explored by a bunch of Theta* for a single goal, as illustrated in Fig. 6b. Two important notes are related to $\mathcal{M}$ here. First, the model $\mathcal{M}$ represents a partial grid, and stores a consecutive sequence of visited cells. Thus, the visited neighboring cells of cell s are determined by

$$\mathcal{N}(s) = \{s' \in \mathcal{M} | c(s, s') \leq \sqrt{2}d\}, \tag{11}$$

where $c(s, s')$ denotes the straight-line distance between two cells s and s', and d is the robot's diameter. Second, a line of sight between s and s' in the model $\mathcal{M}$ exists if a straight line from s to s' exists such that all of the cells on that line belong to the model $\mathcal{M}$. Figure 7 shows an example in which the backtracking paths obtained by the A* search and Theta* for multi-goals are constructed based on the model $\mathcal{M}$.

Let $\mathcal{P} = [s_1, s_2, \cdots, s_n]$ be the path found by Theta* for multi-goals on the model $\mathcal{M}$, where s_1 is the ending point (i.e., $s_1 = s_{ep}$) of the current boustrophedon motion and s_n is the starting point (i.e., $s_n = s_{sp}$) of the next boustrophedon motion. When the robot with the configuration $q_i = [x_i, y_i, \theta_i]^T$ is at cell $s_i = (x_i, y_i, d)$ $(i = 1, 2, \cdots, n-1)$, the

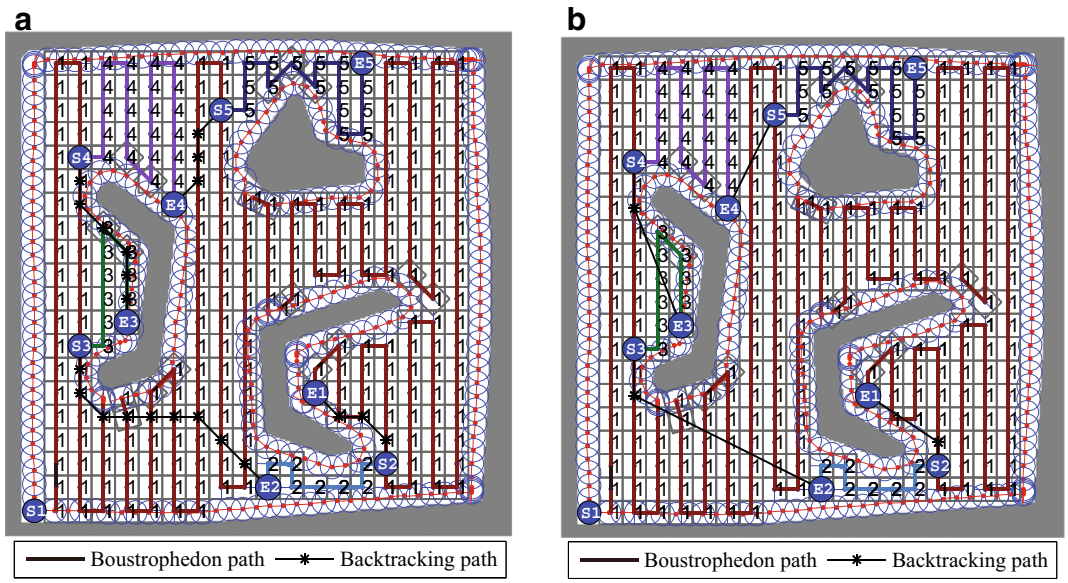


Fig. 7 **a** The backtracking paths found by A* search on the model $\mathcal{M}$; **b** The backtracking paths found by Theta* for multi-goals on the model $\mathcal{M}$, where Si and Ei $(i = 1, 2, \cdots, 5)$ are the starting and ending points of each boustrophedon motion

direction of the robot required to travel to the next cell $s_{i+1} = (x_{i+1}, y_{i+1}, d)$ is

$$\beta_i = \arctan \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \text{ where } \beta_i \in (-\pi, \pi]. \quad (12)$$

The robot needs to turn an angle of α_i, and then move a distance l_i to travel to s_{i+1} as

$$\alpha_i = \beta_i - \theta_i, \text{ where } \alpha_i \in (-\pi, \pi], \text{ and}$$
$$l_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}. \quad (13)$$

When the robot arrives at the starting point $s_n = s_{sp}$, it has to adjust its direction so that a new boustrophedon can be performed. Specifically, the robot has to adjust its heading angle and then move to one of the neighboring positions in the priority directions of N or S. If the direction N is blocked, the robot finds the first available direction in the priority of NE, E, SE, NW, W, and SW. If the direction S is blocked, the robot finds the first available direction in the priority of SE, E, NE, SW, W, and NW. In other words, the robot has to rotate an angle of

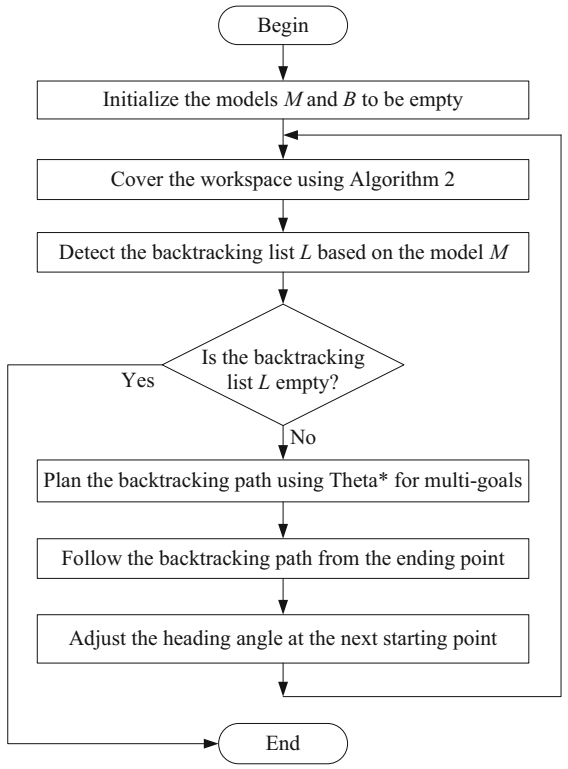
$$\alpha_n = \gamma - \theta_n, \alpha_n \in (-\pi, \pi], \quad (14)$$

where $\theta_n = \beta_{n-1}$ is the heading angle of the robot at the configuration q_n, and $\gamma = k\pi/4$ with $k = -3, -2, \cdots, 4$ depending on whether the direction of the next unblocked position is SW, S, SE, E, NE, N, NW, or W.

3.4 B-Theta* Algorithm

This section presents a B-Theta* algorithm for the online complete-coverage task of an autonomous cleaning robot in unknown workspaces with arbitrarily-shaped obstacles. B-Theta* is shown in Algorithm 3. Initially the robot has no prior knowledge about its workspace, indicating that models $\mathcal{M}$ and $\mathcal{B}$ in the robot's memory are empty. The robot fulfills the coverage mission from its initial position until no backtracking point is detected. B-Theta* is an online algorithm because the robot does not have the full map of the workspace in advance, and has only the local information gathered from its sensors to fulfill the complete-coverage mission. The robot does not have information about the entire workspace, and is forced to choose the optimal backtracking point from its memory, that is, the model $\mathcal{M}$ built so far, as shown in Eq. 9. Thus, the complete-coverage path may not be the global optimization in terms of length. However, the backtracking path is the true shortest path, and the overlap of the lengthwise boustrophedon motions is the minimum; therefore, the coverage path (constrained by the completeness) can be the shortest path. Besides, if the workspace is closed and the accessible regions of the robot are connected and can be reached by the robot from any initial position, then the B-Theta* algorithm provides complete coverage. This is because if an uncovered region exists, it must be connected to at least a covered cell in the model $\mathcal{M}$. In the model $\mathcal{M}$, B-Theta* always gives a backtracking path for the robot to reach the next starting point, and ends when no starting point is detected. Therefore, this method guarantees that the robot visits the uncovered region to cover it.

Algorithm 3 The flowchart of B-Theta* algorithm



4 Simulations and Evaluations

In this section we conduct computer simulations and evaluate the efficiency of the proposed B-Theta* by comparing it with BCD [4, 7] and BA* [29].

4.1 Simulations

This section presents simulations implemented in the Matlab environment on a computer with a 2.4 GHz Core i5-2430M CPU and 4 GB of RAM. The simulations are designed to evaluate the performance of B-Theta* for a cleaning robot in unknown workspaces with arbitrarily-shaped obstacles. The input workspaces are binary images with 300 × 300 pixels, in which each pixel is marked to indicate whether it belongs to an obstacle or not based on its value of one or zero, respectively. We also assume that the robot is modeled by a circle with a radius of $r = 7$ pixels.

The first four simulations aim to verify the complete coverage of B-Theta*. Figure 8a shows the final trajectory of the robot in the first simulation, where each tiny circle along the boundaries of the obstacles represents the boundary-following paths, and the paths labeled 1 to 5 represent the boustrophedon paths. The robot starts at $S1$ to perform the coverage task. After performing the LDBM to cover the portions along the boundary of the workspace, the robot performs the first boustrophedon motion until it encounters the left obstacle while moving south along the 2^{nd} vertical line. The robot then switches to the LDBM to cover the portions along the boundary of the left obstacle, and continues the first boustrophedon motion until it encounters the middle obstacle while moving south along the 8^{th} vertical line. The robot switches again to the LDBM to cover the portion along the boundary of the middle obstacle, and continues the first boustrophedon motion until it reaches the ending point. At the ending point, the robot determines the backtracking list $\mathcal{L}$ based on the current model $\mathcal{M}$, and then plans the backtracking path using Theta* for multi-goals. The robot then follows the backtracking path and performs the second boustrophedon motion. The robot continues to cover the remaining accessible area until it arrives at the ending point $E5$ of the 5^{th} boustrophedon motion, where no backtracking point is detected. If the diameter of the robot is d and the coverage rate is defined as the ratio of the covered pixel number to the accessible pixel number, then the coverage path length is $456.76d$, and the coverage rate is 100 %. For the next three simulations, the robot starts at random positions $S1$ in the workspaces. Figure 8b shows the result of the second simulation. The robot performs seven boustrophedon motions to finish the coverage task at the ending position $E7$. The coverage path length is $461.15d$, and the coverage rate is 100 %. Figure 9 shows the result of the third and fourth simulations. The coverage path length and the coverage rate of the robot in the workspace shown in Fig. 9a are $421.19d$

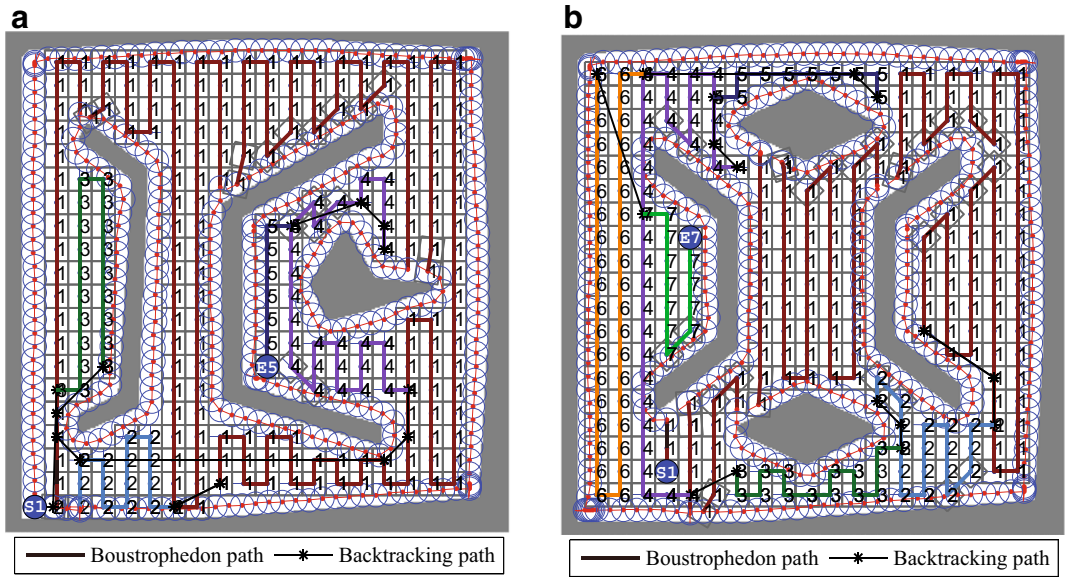


Fig. 8 The coverage paths achieved by B-Theta*. Boustrophedon paths are labeled 1 to 5 and 1 to 6 for cases **a** and **b**, respectively. The remaining paths are the backtracking paths

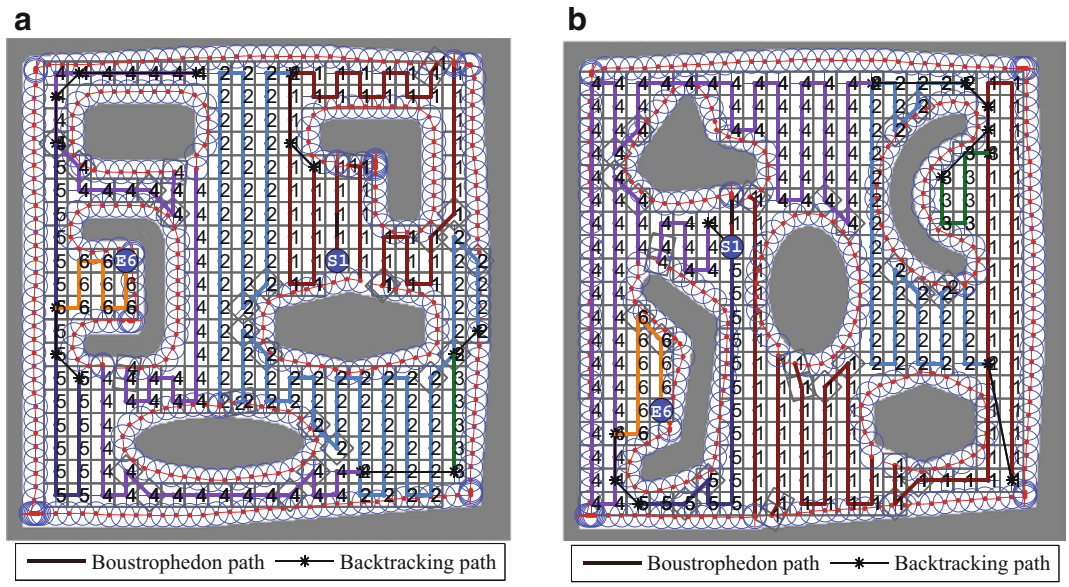


Fig. 9 The coverage paths obtained using B-Theta*. Boustrophedon paths are labeled 1 to 6. The remaining paths are the backtracking paths

and 100 %, respectively, while those of the robot in the workspace shown in Fig. 9b are 438.54d and 100 %, respectively.

We emphasize that Theta* for multi-goals finds the next starting point among the candidates in the backtracking list $\mathcal{L}$ and the backtracking path at the same time. The coverage path for which the best backtracking points are determined by Theta* for multi-goals is always shorter than or equal to that for which the best backtracking points are determined by the Euclidean distance-based cost function. This characteristic is due to the fact that the Euclidean distance measures the straight line distance from the ending point to all candidates in the backtracking list $\mathcal{L}$, and chooses the nearest backtracking point without considering whether the straight line goes through the obstacles. The next two simulations demonstrate this statement. Figure 10a and b describe the final trajectory of the robot for which the best backtracking points are determined by Theta* for multi-goals and the Euclidean distance-based cost function, respectively. Positions Si and $Ei\,(i = 1, 2, ..., 6)$ denote the starting and ending points of the i^{th} boustrophedon motion. When the robot arrives at $E1$, the point $S2$ chosen by Theta* for multi-goals and the Euclidean distance-based cost function is the same. When the robot arrives at $E2$, the point $S3$ measured by the Euclidean distance-based cost function is closer to $E2$ than the point $S3$ measured by Theta* for multi-goals. However, the path from $E2$ to $S3$ determined by Theta* for multi-goals is much shorter than that to $S3$ determined by the Euclidean distance. The coverage path length for which the best backtracking points are determined by Theta* for multi-goals is 458.86d (see Fig. 10a), and that for the Euclidean distance-based cost function is 481.51d (see Fig. 10b). Evidently, in workspaces with U-shaped or S-shaped obstacles, the coverage path for which the best backtracking points are determined by Theta* for multi-goals is much shorter than that for which the best backtracking points are determined by the Euclidean distance-based cost function.

Overall, the six simulations above show that B-Theta* successfully controls the robot in covering completely unknown workspaces with arbitrarily-shaped obstacles, even when the robot starts at a random position in its accessible area.

4.2 Evaluations

4.2.1 B-Theta and BCD Comparison*

This section evaluates B-Theta* by comparing it with the BCD method [4, 7] reviewed in Section 2.1. The criteria of comparison are the successful coverage rate, the total number of decomposed cells, and the coverage path length. Even though our B-Theta* is an

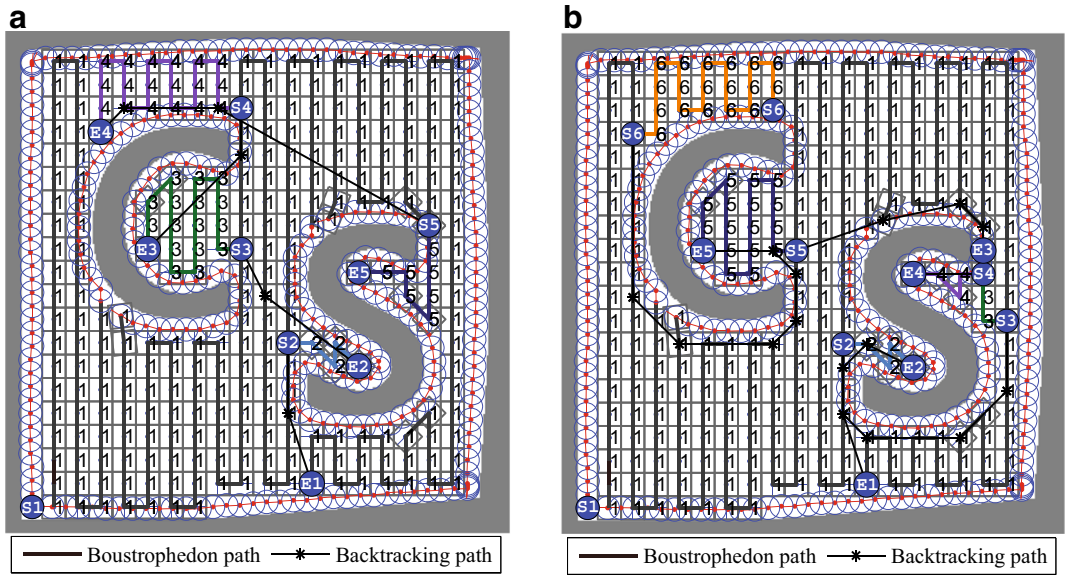


Fig. 10 The coverage paths for which the best backtracking points are determined by **a** Theta* for multi-goals and **b** the Euclidean distance-based cost function

online method and BCD is an offline method, BCD is chosen for comparison with B-Theta* for three main reasons: *(i)* both BCD and B-Theta* use the boustrophedon motion technique to solve the complete-coverage problem; *(ii)* BCD is a popular technique that yields a complete coverage path based on an exact cellular decomposition method that can achieve the shortest coverage path; and *(iii)* BCD can minimize the number of decomposed cells.

BCD is implemented in the workspaces as follows. For instance, BCD decomposes the workspace shown in Fig. 8a into the cells shown in Fig. 11a, and the adjacent graph is then constructed as shown in Fig. 11b, where the vertex v_i represents the cell

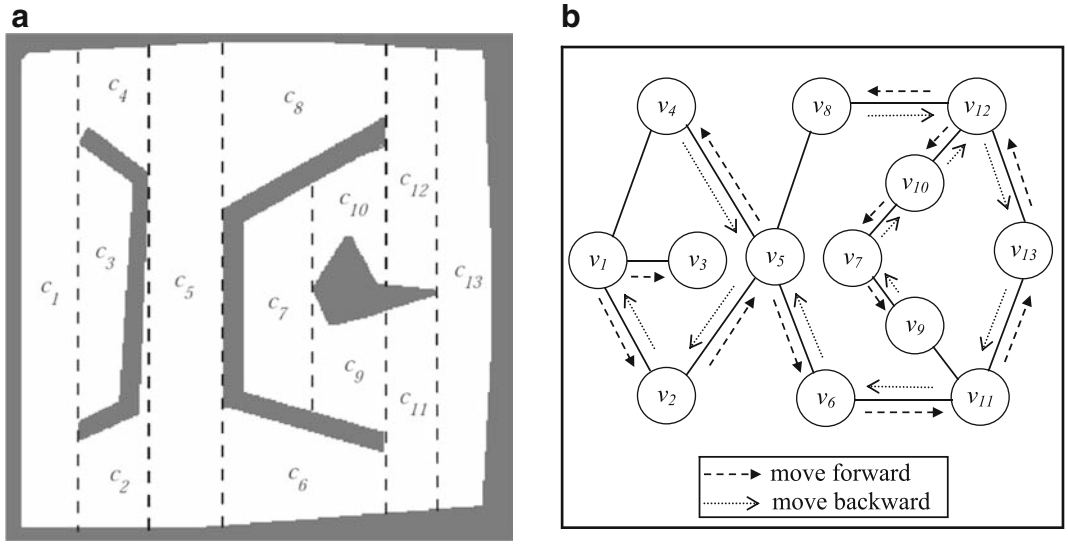


Fig. 11 **a** BCD of the workspace in Fig. 8a; **b** Adjacent graph describes BCD and the walk of the graph starting at vertex v_1

$c_i (i = 1, 2, \cdots, 13)$ of the decomposition. Starting from vertex v_1, the walk is $\mathcal{V} = [v_1, v_2, v_5, v_6, v_{11}, v_{13}, v_{12}, v_8, v_{12}, v_{10}, v_7, v_9, v_7, v_{10}, v_{12}, v_{13}, v_{11}, v_6, v_5, v_4, v_5, v_2, v_1, v_3]$, which is illustrated in Fig. 11b. The robot covers an unvisited cell with a single boustrophedon motion in the order of $c_1, c_2, c_5, c_6, c_{11}, c_{13}, c_{12}$, and c_8, and then backtracks to cover cells c_{10}, c_7, c_9, c_4, and c_3. Figure 12a shows the coverage path obtained from walk $\mathcal{V}$. The coverage path length is $501.65d$, the coverage rate is 95.24 %, and the number of decomposed cells is 13. Similarly, Fig. 12b shows the coverage path of the workspace in Fig. 8b. The coverage path length is $441.33d$, the coverage rate is 94.24 %, and the number of decomposed cells is 11.

In comparing with BCD, we design 16 workspaces to implement simulations as follows. The first eight workspaces are binary images of 300 × 300 pixels, and the remaining workspaces are binary images of 400 × 400 pixels. Each workspace contains five or six obstacles: one each of L-shaped, U-shaped, triangular-shaped, and elliptical-shaped obstacles, and one or two bar-shaped obstacles. The workspaces are generated by placing the obstacles randomly in terms of position and angle to generate diverse situations. The area of each obstacle is 3 to 6 % of the workspace area. The robot is modeled by a circle with a radius of $r = 7$ pixels. For each workspace we perform one pair of simulations for B-Theta* and BCD, in which the initial position of the robot is the same for both B-Theta* and BCD and is placed randomly in the accessible area of each workspace. Figure 13 shows the coverage rate achieved by the 16 simulations with B-Theta* and BCD. The coverage rate achieved by B-Theta* is 100.00 % for all workspaces, while the coverage rate achieved by BCD is only 90.77 to 96.88 %. Although BCD covers more than 90 % of the workspaces, it cannot cover the portions along the boundaries of the workspace and obstacles, where the dirt is more concentrated than the other regions. Figure 14 compares the number of boustrophedon regions achieved by B-Theta* and the number of decomposed cells achieved by BCD. The number of decomposed regions achieved by B-Theta* is always smaller than that achieved by BCD. Specifically, the number of decomposed regions achieved by B-Theta* is 33.33 % (for workspace ID 11) to 64.71 % (for workspace ID 3) smaller than that achieved by BCD. As a result, this advantage contributes significantly to the shortening of the coverage path length achieved by B-Theta*. Figure 15 compares the coverage path lengths achieved by B-Theta* and BCD. In each workspace the coverage path length achieved by B-Theta* is decomposed into two parts. The lower part represented by the brown color shows the total length of the boustrophedon paths and the other part represented by the yellow color shows the total length

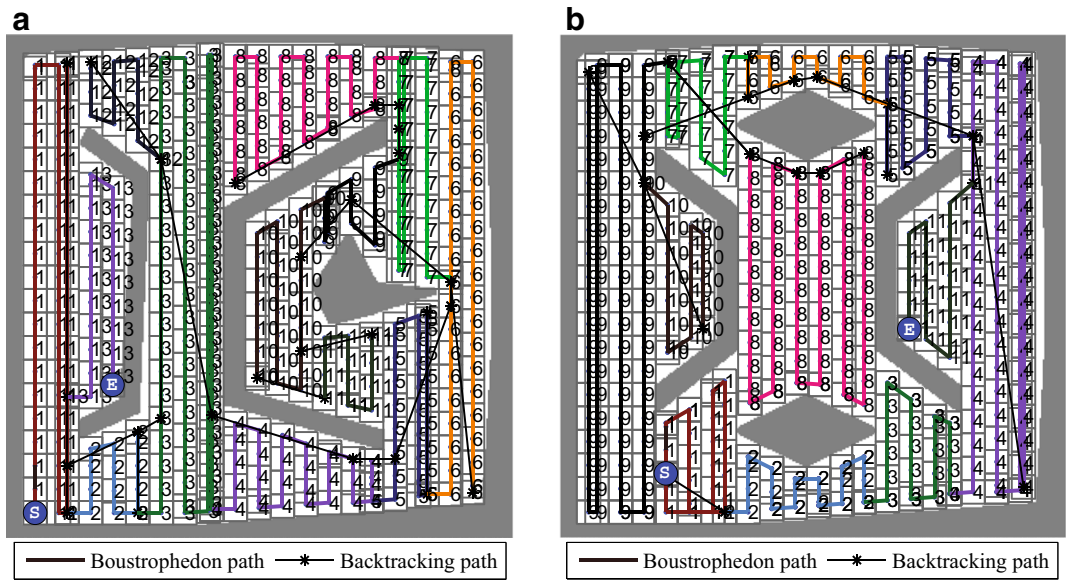


Fig. 12 The coverage paths achieved by BCD in the workspaces in Fig. 8a and b

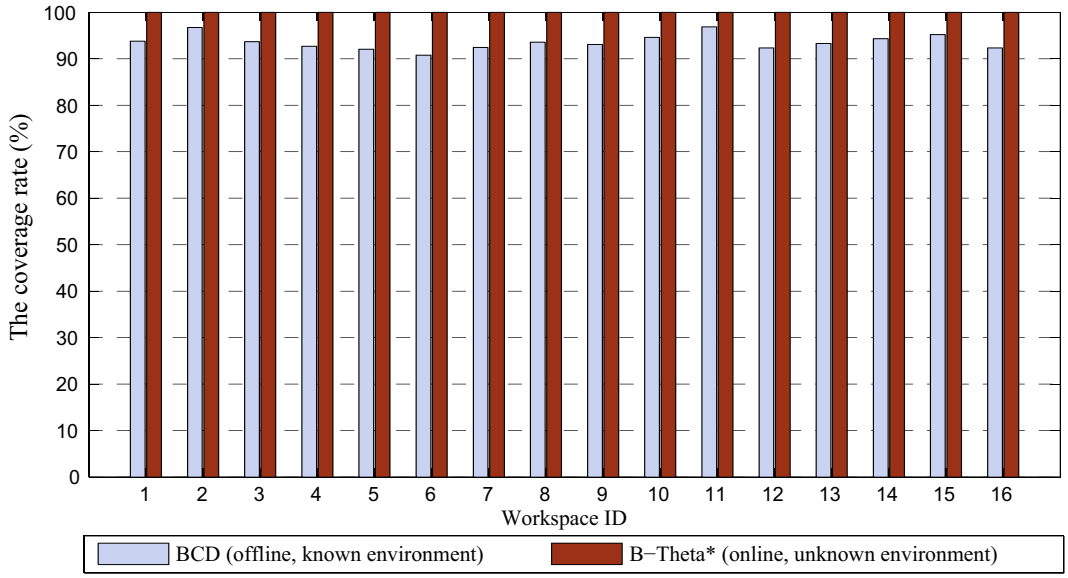


Fig. 13 The coverage rate achieved by BCD and B-Theta* in the workspaces

of the paths along the obstacle boundaries. The total length of the boustrophedon paths achieved by B-Theta* is 18.88 % (for workspace ID 16) to 31.63 % (for workspace ID 3) shorter than the coverage path length achieved by BCD. However, the coverage path length achieved by B-Theta* is 4.60 % (for workspace ID 1) to 14.05 % (for workspace ID 13) longer than that achieved by BCD, because B-Theta* enables the robot to cover portions along the obstacle boundaries. Although the total length of paths along the obstacle boundaries is still long, the coverage path length achieved by B-Theta* is also approximately the same as that achieved by BCD.

According to the simulation results, the several main evaluations of B-Theta* can be summarized as follows:

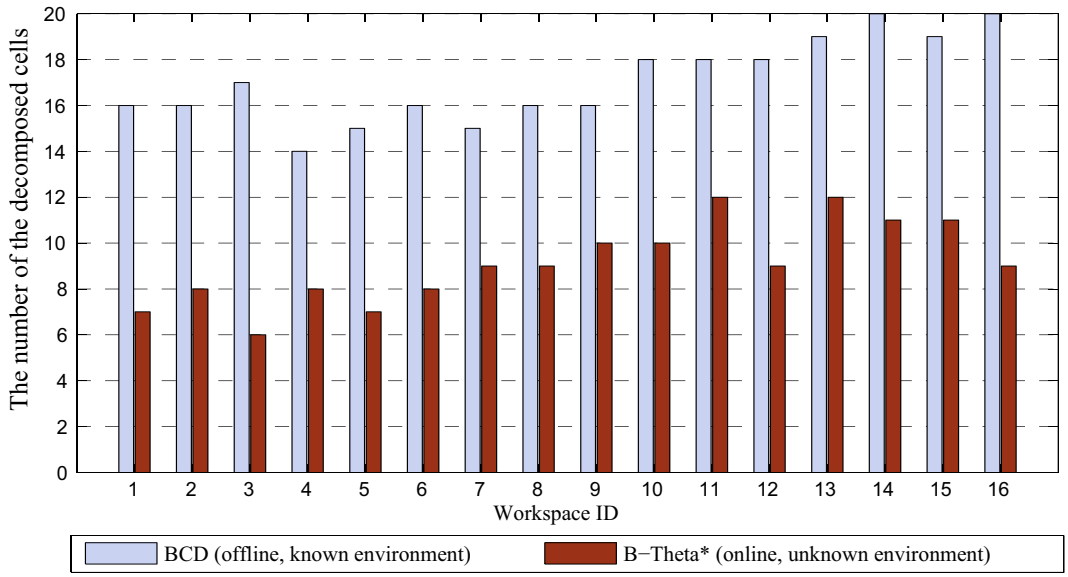


Fig. 14 The number of decomposed cells achieved by BCD and the number of boustrophedon regions achieved by B-Theta* in the workspaces

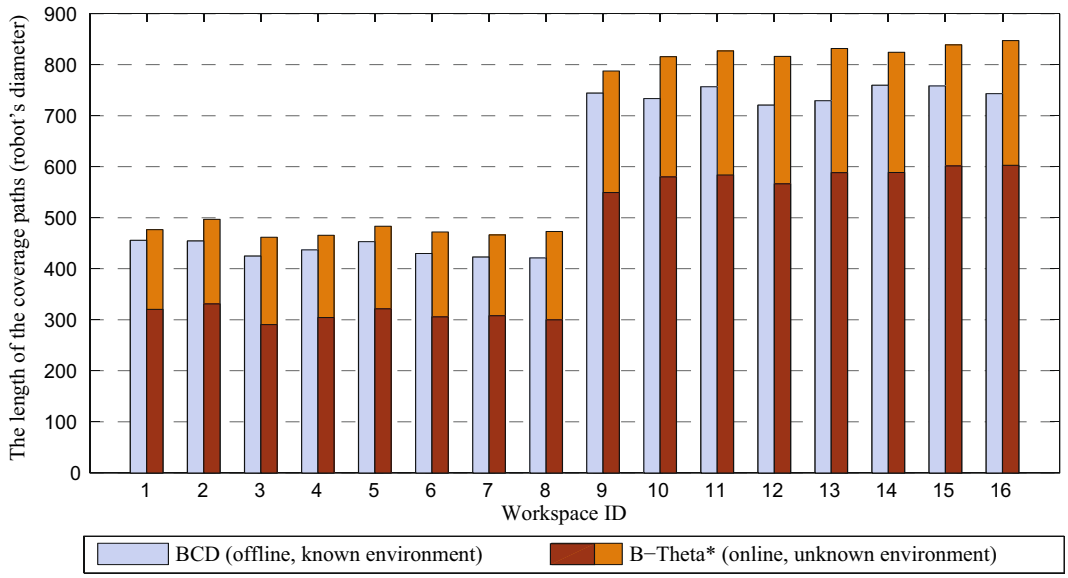


Fig. 15 The coverage path length achieved by BCD and B-Theta* in the workspaces

(a) Under control of B-Theta*, the coverage rate of the robot is 100 %, which means that B-Theta* provides complete coverage. By contrast, BCD does not provide complete coverage, because the robot uses only boustrophedon motions to cover the workspaces. Specifically, the robot cannot cover the small portions along the obstacle boundaries, because the boustrophedon motions only consider the heading angle of the robot in the directions of N, E, S, and W when the robot encounters obstacles. Although the uncovered portions along the obstacle boundaries are small compared with the entire workspace, these portions are still important because dirt is often concentrated and accumulated there. BCD is efficient in terms of coverage when the obstacles are rectangular. However, BCD already cannot cover 100 % of the workspace in this case, let alone cover cases in which arbitrarily-shaped obstacles are present.

(b) The total number of decomposed regions achieved by B-Theta* is much smaller than that achieved by BCD. B-Theta* decomposes the accessible area of the workspaces into regions based on the boustrophedon motions, whereas BCD decomposes the accessible area of the workspaces into cells based on IN and OUT events determined by the changes in the connectivity of a slice. Therefore, a boustrophedon region of B-Theta* can contain more than one cell formed by BCD (e.g., the first boustrophedon region in Fig. 8a contains cells $c_1, c_4, c_5, c_8, c_{12}, c_{13}, c_{11}$, and c_6 in Fig. 11a). This implies that the total number of decomposed regions achieved by B-Theta* is smaller than that achieved by BCD. As a result, the length of the backtracking path shrinks appreciably, so the coverage path is shortened.

(c) The coverage path achieved by B-Theta* may be a little longer than that achieved by BCD, because B-Theta* covers the portions along the boundaries of the obstacles, whereas BCD does not. However, a complete-coverage criterion is the first priority for any cleaning robot. In addition, the longer path achieved by B-Theta* compared to that achieved by BCD can be compensated by three reasons. First, the backtracking mechanism of B-Theta* to the next uncovered region backtracks through the shortest collision-free path, whereas the backtracking mechanism of BCD to the next uncovered region backtracks through the visited regions. Second, BCD can repeat or overlap the lengthwise motions to move to the first counterclockwise unvisited cell (e.g., the lengthwise motions are overlapped at the end of the 1^{st} or the 3^{rd} boustrophedon motion in Fig. 12a). By contrast, in B-Theta* the robot moves to the empty neighboring cell instead of repeating the lengthwise motion. Lastly, the total number of decomposed

regions achieved by B-Theta* is much smaller than that achieved by BCD. These three advantages contribute significantly to decreasing the coverage path length.

Evidently, we can conclude that B-Theta* dominates BCD in terms of the necessity of prior knowledge about the workspaces (online or offline) and the coverage rate.

4.2.2 B-Theta and BA* Comparison*

This section evaluates B-Theta* by comparing it with our previous BA* algorithm [29]. As in the comparison with BCD, the criteria of comparison for B-Theta* with BA* are the successful coverage rate, the total number of boustrophedon regions, and the coverage path length. BA* is chosen to compare with B-Theta* because it is an online algorithm and is superior to BCD in terms of the total number of decomposed regions and coverage path length.

We reuse the 16 workspaces described in Section 4.2.1 to compare B-Theta* with BA*. We perform one pair of simulations for B-Theta* and BA* for each workspace. The initial position of the robot is the same for both B-Theta* and BA* in each pair of simulations, and is placed randomly in the accessible area of each workspace. Figures 16, 17, and 18 show the results of the simulations. Figure 16 shows the coverage rate for each workspace. B-Theta* achieves a coverage rate of 100 % for all simulation workspaces. In contrast, BA* achieves a coverage rate of only 90.55 % (for workspace ID 6) to 95.56 % (for workspace ID 5). Like BCD, under control of BA* the robot cannot cover the entire workspace, because it uses only boustrophedon motions to cover the workspaces. Figure 17 shows the total number of boustrophedon regions achieved by B-Theta* and BA* for each workspace. The total number of boustrophedon regions achieved by B-Theta* is smaller than or equal to that achieved by BA*. Finally, Fig. 18 shows the coverage path length achieved by B-Theta* and BA* for each workspace. The coverage path length achieved by B-Theta* consists of two parts. The first part is the total length of the boustrophedon paths, and the second part is the total length of the paths along the obstacle boundaries. The total length of the boustrophedon paths achieved by B-Theta* is always shorter than the coverage path length achieved by BA*. However, the coverage path length achieved by BA* is shorter than that achieved by B-Theta*, because BA* does not enable the robot to cover portions along the obstacle boundaries. Specifically, the

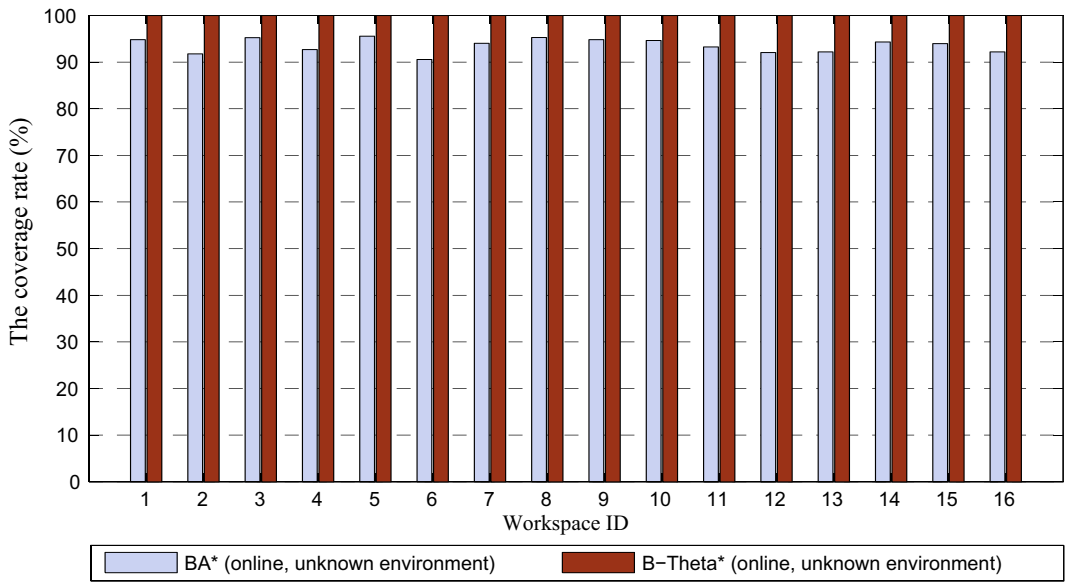


Fig. 16 The coverage rate achieved by BA* and B-Theta* in the workspaces

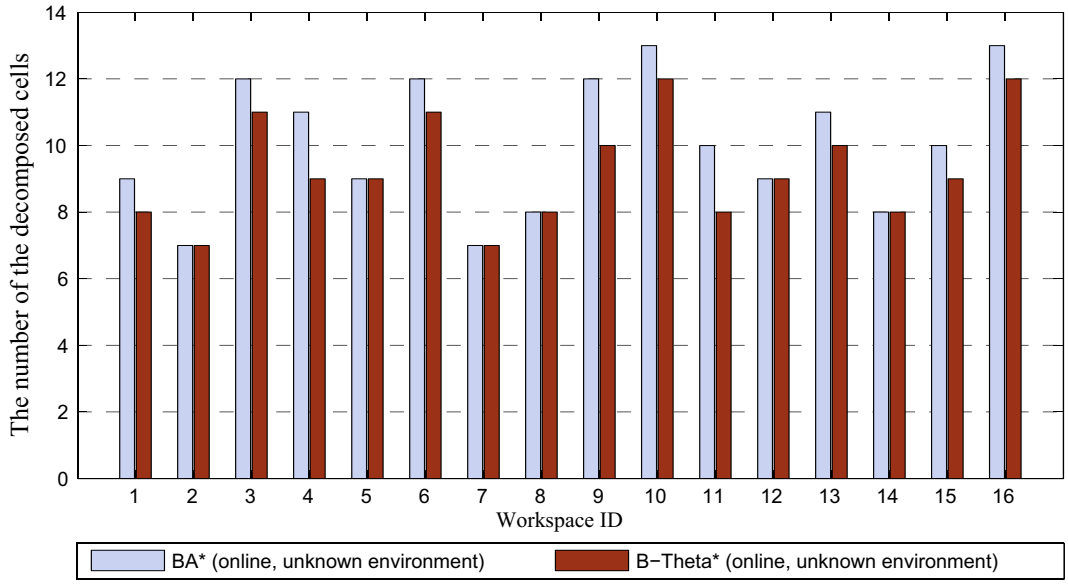


Fig. 17 The number of boustrophedon regions achieved by BA* and B-Theta* in the workspaces

coverage path length achieved by BA* is 6.58 % (for workspace ID 4) to 14.75 % (for workspace ID 2) shorter than that achieved by B-Theta*.

In summary, BA* dominates B-Theta* in terms of the coverage path length, but B-Theta* dominates BA* in terms of the coverage rate.

4.2.3 B-Theta, BA* and BCD Execution Time Comparison*

Lastly, we compare the robot coverage execution time of the algorithms B-Theta*, BA* and BCD. If the velocity of the robot is the same in all simulations,

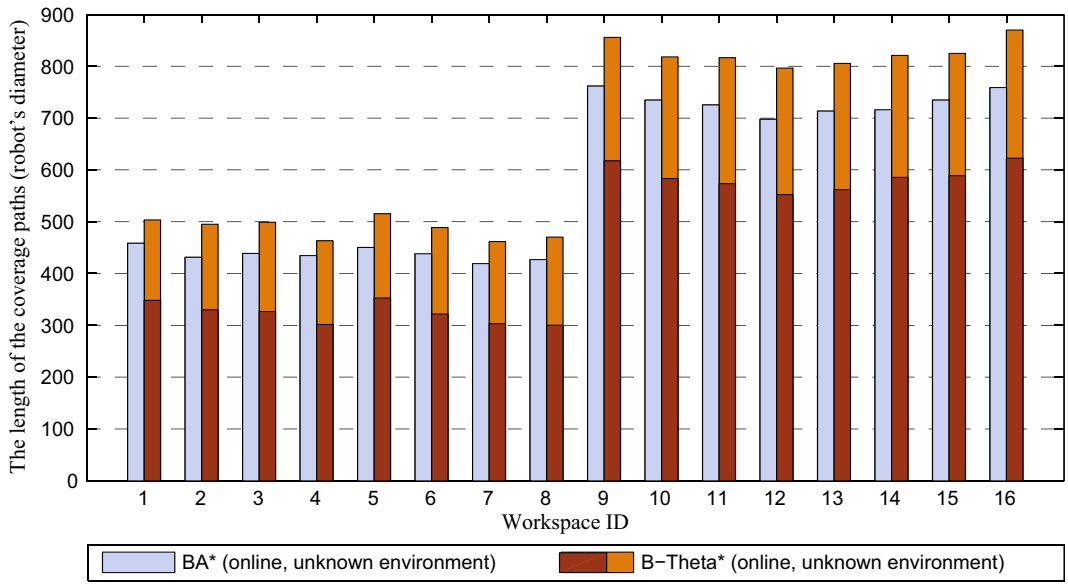


Fig. 18 The coverage path length achieved by BA* and B-Theta* in the workspaces

then the robot coverage execution time depends on the execution time of the algorithms. Figure 19 shows the execution time of B-Theta*, BA* and BCD algorithms for 16 workspaces described in Section 4.2.1. Although the coverage path length found by BA* is shorter than that found by BCD [29] and that found by B-Theta*, the execution time of BA* is longer than that of BCD and that of B-Theta*. This is because BA* has to find all collision-free paths to all backtracking points at each ending point of boustrophedon motion, and then chooses the shortest path as the backtracking path. Meanwhile, B-Theta* finds the shortest backtracking path to all backtracking points using the proposed Theta* for multi-goals and BCD does not use a backtracking mechanism. However, the execution time of B-Theta* is longer than that of BCD because the coverage path found by B-Theta* is longer than that found by BCD.

5 Experiments

This section presents two experiments in actual environments to demonstrate the proposed B-Theta* for autonomous cleaning robots. The experiments are conducted by implementing B-Theta* on *iRobot Create* ♯4400 [28], a programmable Roomba vacuum cleaner designed for education and research. Most Roomba versions do not have the ability to self-execute and perform only via a software interface. The software interface is used to manipulate *iRobot Create*'s behavior and read its sensors through the native low-level numerical commands sent to its serial port using a personal computer (PC). In our experiments the software interface is established wirelessly via *iRobot Create*'s serial port using a Bluetooth connection. A set of compatible Matlab functions in the *iRobot Create* toolbox [12] are used to alter the native low-level numerical commands. These functions perform three main tasks: *(i)* establishing a Bluetooth connection between the PC and the robot; *(ii)* getting the information from the robot's sensors for touch/cliff detection and moving status (i.e., the distance driven and the angle turned); and *(iii)* providing the driving commands to the robot.

B-Theta* is developed and implemented using Matlab programs to control the robot. The programs are coded, stored, and executed on a laptop to drive the robot online during the experiment tests. The CPU of the laptop acts as *iRobot Create*'s processing unit. While moving, the robot can detect obstacles using the three touch sensors on the front, left, and right sides. Although the workspaces in these experiments are not as complicated as those in the simulations, they still feature the characteristics of B-Theta* to avoid a long observation time. Specifically, the robot

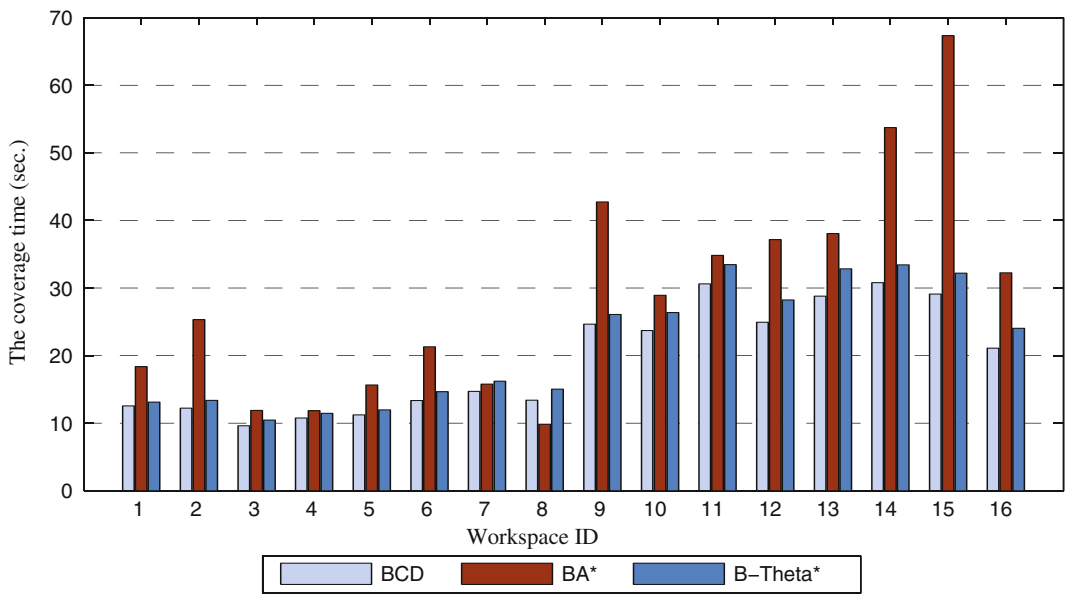


Fig. 19 The execution time achieved by B-Theta*, BA* and BCD in the workspaces

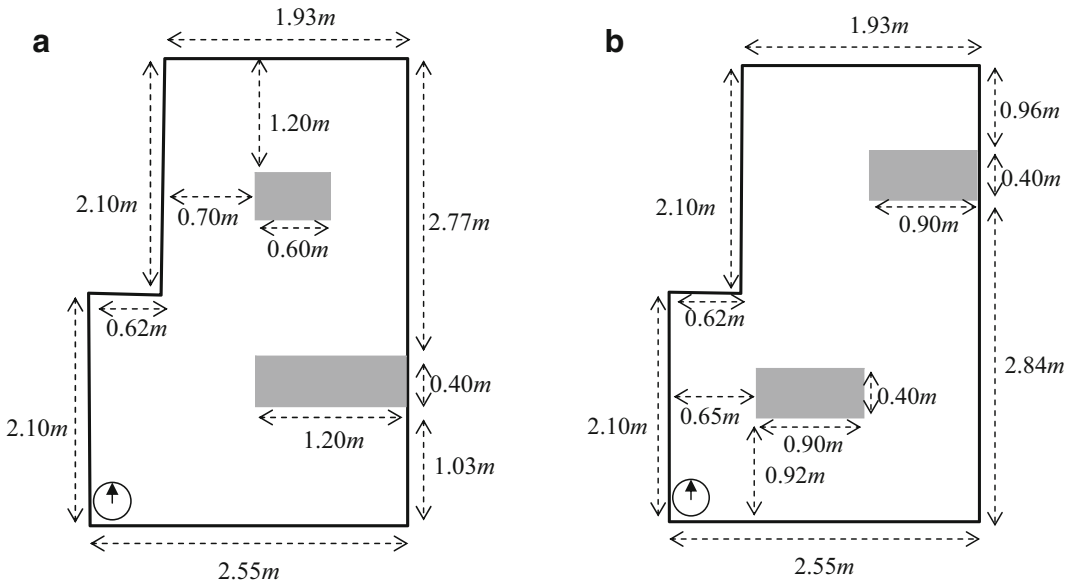


Fig. 20 The workspace of the first **a** and the second experiment **b**

is driven to perform the backtracking paths and cover the workspaces in the boustrophedon mode and the boundary mode. The values of the basic parameters in all of the experiments are as follows. The velocity of the robot is 0.15 m/s, and the distance of each robot's movement is 0.3*m* (i.e., equal to the diameter

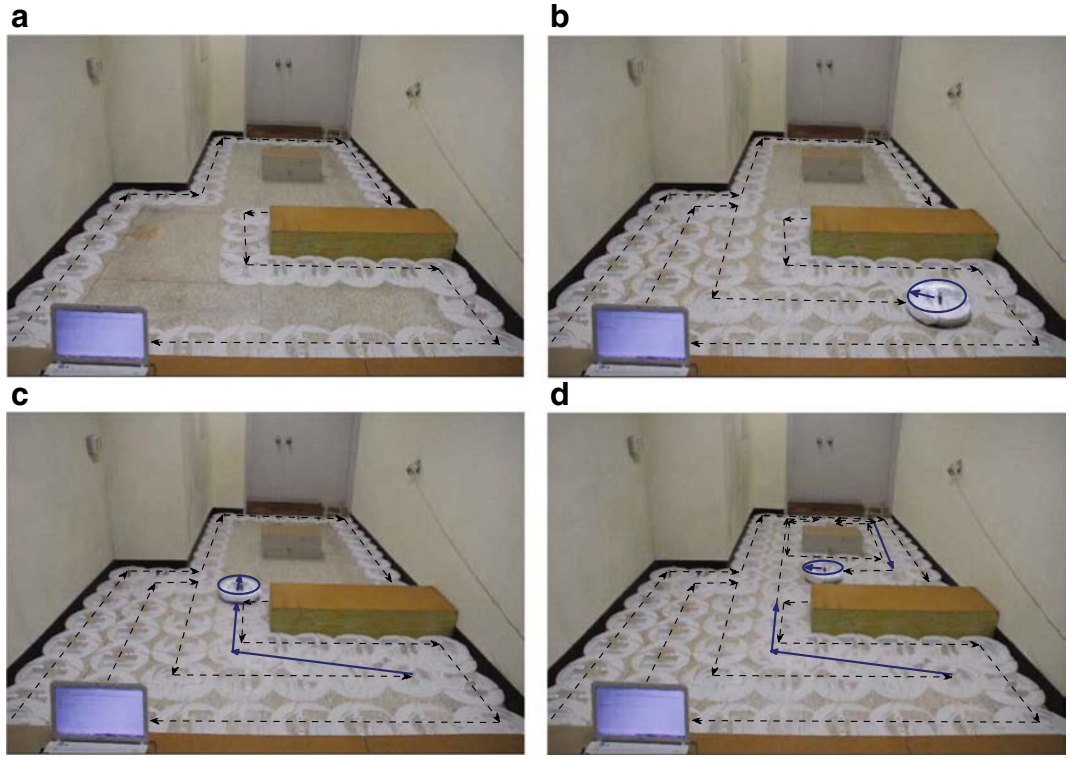


Fig. 21 The trace of *iRobot Create* in the first experiment

of *iRobot Create*). Figure 20a and b show the details of the workspaces. The experiments are recorded using a digital camera at 24 frames/second and then saved to video files. The trace of *iRobot Create* in the video files is extracted by an algorithm that subtracts frames from the background frame to get the robot's positions.

Figure 21 shows the trace of *iRobot Create* in the first experiment, where the dashed arrows describe the boustrophedon motions and boundary-following motions of *iRobot Create*, the solid arrows describe the backtracking path of *iRobot Create* achieved by Theta* for multi-goals, and the circle and the bold solid arrow describe the current position and heading direction of *iRobot Create*. Initially, *iRobot Create* is placed beside the wall at the southwest corner of the workspace. The robot then follows the wall to cover the portions along the boundary of the workspace, as shown in Fig. 21a, after which *iRobot Create* performs the first boustrophedon motion to cover an accessible region of the workspace until it reaches an ending point. At the ending point it changes direction, as shown in Fig. 21b, and then follows the backtracking path found by Theta* for multi-goals to the best backtracking point, as shown in Fig. 21c. The robot performs the next boustrophedon motion until it encounters the obstacle. Finally, the robot covers the obstacle boundary and the remaining region as illustrated in Fig. 21d. The experimental time to cover the workspace is 4.56 min.

Figure 22 shows the trace of the robot in the second experiment. After finishing the wall-following motion of the workspace as in Fig. 22a, *iRobot Create* changes direction and performs the first boustrophedon motion until it encounters the obstacle at the middle of the workspace. The robot then follows the obstacle boundary, as shown in Fig. 22b, after which it continues the first boustrophedon motion until an ending point is detected. At the ending point it plans the backtracking path using Theta* for multi-goals, and then follows this path to the best backtracking point, as shown in Fig. 22c. Finally, it covers the final accessible region, as shown in Fig. 22d. The experimental time to cover the workspace is 5.11 min.

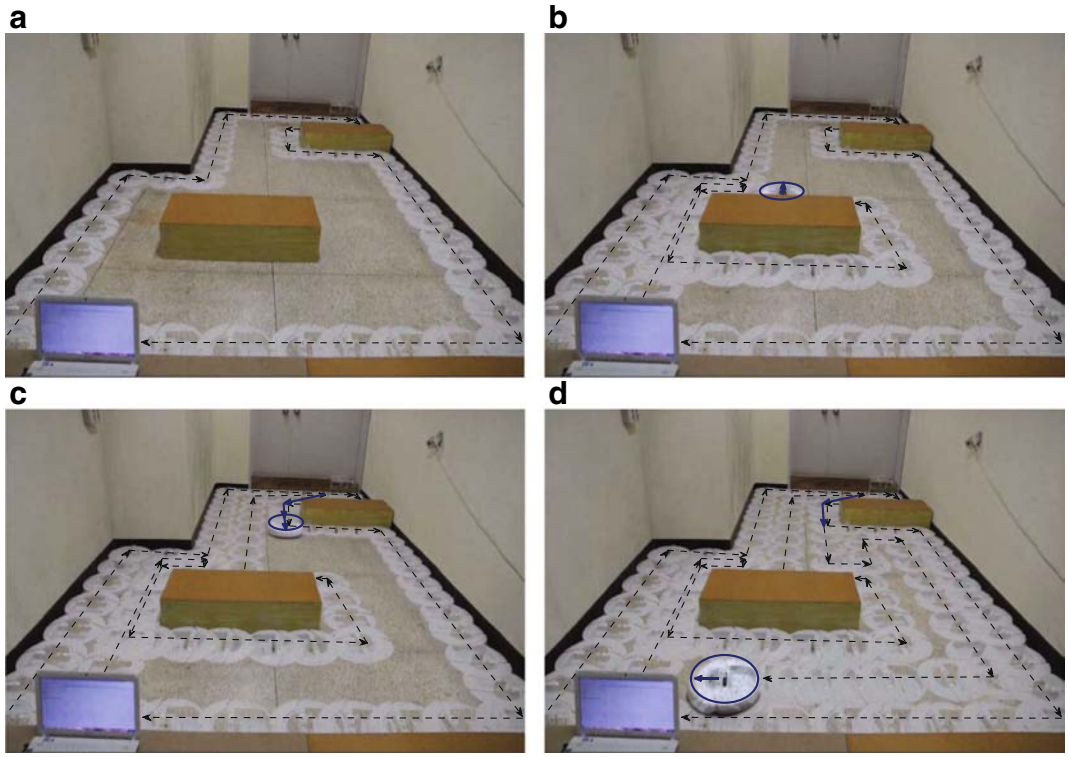


Fig. 22 The trace of *iRobot Create* in the second experiment

The experiments show that although *iRobot Create* is equipped with only a few bump sensors on the front, left, and right sides without any sensors to determine its location in the workspace, B-Theta* works properly under experimental conditions and fulfills the complete-coverage mission. Moreover, B-Theta* is efficient for autonomous cleaning robots in terms of the coverage path length and the coverage rate.

6 Conclusions

In this paper we present an online complete-coverage algorithm for autonomous cleaning robots in completely unknown workspaces with arbitrarily-shaped obstacles based on the boustrophedon motions, the boundary-following motions, and the Theta* algorithm known as B-Theta*. In our approach the robot performs a single boustrophedon motion to cover an unvisited region. While performing the boustrophedon motion, if the robot encounters an obstacle with a boundary that has not yet been covered, it switches to the boundary mode to cover portions along the obstacle boundary, and then continues the boustrophedon mode until an ending point is detected. At the ending point the robot detects the backtracking points based on its accumulated knowledge, plans the shortest backtracking path to the backtracking points based on the proposed Theta* for multi-goals, and then follows the backtracking path to cover the next unvisited region. The coverage mission of the robot finishes when no backtracking point is detected. The computer simulations show that our B-Theta* provides complete coverage in unknown workspaces with arbitrarily-shaped obstacles. Moreover, B-Theta* dominates the BCD approach [4, 7] in terms of the online working manner and the coverage rate. B-Theta* dominates the BA* approach [29] in terms of the coverage rate. The experiments with *iRobot Create* in actual environments show that even when equipped with only a few touch sensors, *iRobot Create* maintains and reaches the desired backtracking points correctly with the exact navigation, and fulfills the online complete-coverage mission. Since B-Theta* is an online algorithm, the robot must construct the coverage path step by step while executing the coverage algorithm. When the robot reaches an ending point, it does not have information about the entire workspace, and is therefore forced to choose the local optimal backtracking point from its memory. As a result, the complete-coverage path achieved by B-Theta* may not be the global optimization in terms of length. Even so, both computer simulations and practical experiments show that our B-Theta* algorithm is an efficient approach to deal with the complete-coverage problem of cleaning robots in actual environments.

Acknowledgments The authors are grateful to the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology (2014R1A1A2057735), IITP (2015-(R0134-15-1033)).

References

1. Acar, E.U., Choset, H., Rizzi, A.A., Atkar, P.N., Hull, D.: Morse decompositions for coverage tasks. Int. J. Robot. Res. **21**(4), 331–344 (2002)
2. Botea, A., Mller, M., Schaeffer, J.: Near optimal hierarchical path-finding. Journal of Game Development **1**(1), 7–28 (2004)
3. Chibin, Z., Xingsong, W., Yong, D.: Complete coverage path planning based on ant colony algorithm. In: Proceedings of the 15Th International Conference on Mechatronics and Machine Vision in Practice, pp. 357–361. Auckland, New-Zealand (2008)
4. Choset, H.: Coverage of known spaces: the boustrophedon cellular decomposition. Auton. Robot. **9**(1), 247–253 (2000)
5. Choset, H.: Coverage for robotics - a survey of recent results. Ann. Math. Artif. Intell. **31**(1–4), 113–126 (2001)
6. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Boston (2005)
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 (1997)
8. Daniel, K., Nash, A., Koenig, S.: Theta*: any-angle path planning on grids. J. Artif. Intell. Res. **39**(1), 533–579 (2010)
9. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
10. Dlouhy, M., Brabec, F., Svestka, P.: A genetic approach to the cleaning path planning problem. In: Proceedings of the 16th European Workshop on Computational Geometry. Eilat, Israel (2000)
11. Dudek, G., Jenkin, M.: Computational Principles of Mobile Robotics, 2nd edn. Cambridge University (2010)
12. Esposito, J.M., Barton, O., Koehler, J., Lim, D.: Matlab toolbox for the create robot (2011). www.usna.edu/Users/weapsys/esposito/roomba.matlab/
13. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. Ann. Math. Artif. Intell. **31**(4), 77–98 (2001)

14. Gabriely, Y., Rimon, E.: Spiral-STC: an on-line coverage algorithm of grid environments by a mobile robot. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 954–960, Washington, DC, USA (2002)
15. Gonzalez, E., Alvarez, O., Diaz, Y., Parra, C., Bustacara, C.: BSA: a complete coverage algorithm. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2040–2044, Barcelona, Spain (2005)
16. Gonzalez, E., Aristizbal, P.T., Alarcn, M.A.: Backtracking spiral algorithm: a mobile robot region filling strategy. In: Proceeding of the 2002 International Symposium on Robotics and Automation, pp. 261–266, Toluca, Mexico (2002)
17. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)
18. Koenig, S., Liu, Y.: Terrain coverage with ant robots: a simulation study. In: Proceedings of the International Conference on Autonomous Agents, pp. 600–607, Montreal, Quebec, Canada (2001)
19. Korf, R.E., Reid, M., Edelkamp, S.: Time complexity of iterative-deepening-A*. Artif. Intell. **129**(1-2), 199–218 (2001)
20. Latombe, J.C.: Robot Motion Planning. Kluwer Academic Publishers (1991)
21. Luo, C., Yang, S.X.: A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments. IEEE Trans. Neural Netw. **19**(1), 1279–1298 (2008)
22. Mannadiar, R., Rekleitis, I.: Optimal coverage of a known arbitrary environment. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 5525–5530, Anchorage, Alaska, USA (2010)
23. Nash, A., Daniel, K., Koenig, S., Felner, A.: Theta*: any-angle path planning on grids. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 1177–1183. Vancouver, Canada (2007)
24. Oh, J.S., Choi, Y.H., Park, J.B., Zheng, Y.F.: Complete coverage navigation of cleaning robots using triangular-cell-based map. IEEE Trans. Ind. Electron. **51**(3), 718–726 (2004)
25. Palleja, T., Tresanchez, M., Teixido, M., Palacin, J.: Modeling floor-cleaning coverage performances of some domestic mobile robots in a reduced scenario. Robot. Auton. Syst. **58**(1), 37–45 (2010)
26. Russel, S.J., Norvig, P.: Artificial Intelligence a Modern Approach, 2nd edn. Pearson Education (2003)
27. Shivashankar, V., Jain, R., Kuter, U., Nau, D.: Real-time planning for covering an initially-unknown spatial environment. In: Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference, pp. 63–68, Florida, USA (2011)
28. The iRobot Create Team: iRobot create owners guide (2006). www.irobot.com/hrd_right_rail/create_rr/create_fam/createFam_rr_manuals.html
29. Viet, H.H., Dang, V.H., Laskar, M.N.U., Chung, T.: BA*: an online complete coverage algorithm for cleaning robots. Appl. Intell. (2012). doi:10.1007/s10489-012-0406-4
30. Wong, S.: Qualitative Topological Coverage of Unknown Environments by Mobile Robots. PhD dissertation, the University of Auckland, New Zealand (2006)
31. Wong, S.C., MacDonald, B.A.: A topological coverage algorithm for mobile robots. In: Proceedings of the International Conference on Intelligent Robots and Systems, pp. 1685–1690 (2003)
32. Yang, S.X., Luo, C.: A neural network approach to complete coverage path planning. IEEE Trans. Syst. Man Cybern. B Cybern. **34**(1), 718–724 (2004)
33. Yap, P.: Grid-based path-finding. Lecture Notes in Artificial Intelligence **2338**(1), 44–55 (2002)

SeungYoon Choi received the B.S. degree in Information and Communication from Korea Nazarene University, Republic of Korea, in 2010, and the M.S. degree in Computer Engineering from Kyung Hee University, Republic of Korea, in 2012, respectively. He is now working toward a Ph.D. degree at Artificial Intelligence Laboratory, Department of Computer Engineering, Kyung Hee University, Republic of Korea. His current research interests include Machine Learning, Optimization, and Robotics.

SeungGwan Lee received the B.S., M.S. and Ph.D. degrees in Department of Computer Engineering from Kyung Hee University, Republic of Korea, in 1997, 1999 and 2004, respectively. He was visiting Professor in School of Computer Science and Information Engineering at Catholic University in 2004 and 2006. He is now an Associate Professor in the Humanitas College at Kyung Hee University since 2006. His research interests include the Artificial Intelligence, Meta-Search Algorithm, Multi-Agents, Ubiquitous Computing, Image Processing, ITS, STEAM Education.

Hoang Huu Viet received the B.S. degree in Mathematics from Vinh University, Nghean, VietNam, in 1994, and the B.S. and M.S. degrees in Computer Science from Hanoi University of Technology, Hanoi, Vietnam, in 1998 and 2002, respectively. He received the Ph.D. degree in Computer Engineering from Kyung Hee University, Republic of Korea in August 2013. He is now a lecturer at Department of Information Technology, Vinh University, VietNam. His current research interests include Artificial Intelligence, Reinforcement Learning, and Robotics.

TaeChoong Chung received the B.S. degree in Electronic Engineering from Seoul National University, Republic of Korea, in 1980, and the M.S. and Ph.D. degrees in Computer Science from KAIST, Republic of Korea, in 1982 and 1987, respectively. Since 1988, he has been with Department of Computer Engineering, Kyung Hee University, Republic of Korea, where he is now a Professor. His research interests include Machine Learning, Meta Search, and Robotics.