CAAI Transactions on Intelligence Technology

Research Article

Sensor-based complete coverage path planning in dynamic environment for cleaning robot

ISSN 2468-2322
Received on 25th December 2017
Revised on 12th February 2018
Accepted on 22nd February 2018
doi: 10.1049/trit.2018.0009
www.ietdl.org

Hong Liu, Jiayao Ma ✉, Weibo Huang

Human Robot Interaction Lab, Shenzhen Graduate School, Peking University, Shenzhen 518055, People's Republic of China
✉ *E-mail: jiayaoma@pku.edu.cn*

Abstract: Using Complete Coverage Path Planning (CCPP), a cleaning robot could visit every accessible area in the workspace. The dynamic environment requires the higher computation of the CCPP algorithm because the path needs to be replanned when the path might become invalid. In previous CCPP methods, when the neighbours of the current position are obstacles or have been visited, it is challenging for the robot to escape from the deadlocks with the least extra time cost. In this study, a novel CCPP algorithm is proposed to deal with deadlock problems in a dynamic environment. A priority template inspired by the short memory model could reduce the number of deadlocks by giving the priority of directions. Simultaneously, a global backtracking mechanism guides the robot to move to the next unvisited area quickly, taking the use of the explored global environmental information. What's more, the authors extend their CCPP algorithm to a multi-robot system with a market-based bidding process which could deploy the coverage time. Experiments of apartment-like scenes show that the authors' proposed algorithm can guarantee an efficient collision-free coverage in dynamic environments. The proposed method performs better than related approaches on coverage rate and overlap length.

1 Introduction

Complete Coverage Path Planning (CCPP) requires robots to pass over every part of the workspace completely in collision free paths. The robot must fill the region with few overlapping paths within a limited time. Comparing with CCPP methods with stationary environments, when more computation is applied to detect the moving obstacle correctly and efficiently, the path in the dynamic environment getting from the last computing might become invalid at any time. Therefore, the algorithm requires higher efficiency to replan in real time. There are few mature methods suitable in a dynamic environment. In this paper, we focus on CCPP algorithm for cleaning robots in the environment with dynamic elements like pets and kids, for example. It is of great importance to replan effectively in order to get a safer and better path. There are many robotic applications using CCPP algorithms [1–3], such as demining robots [4], lawn mowers [5], automated harvesters [6], underwater robots [7], cleaning robots [8] and so on.

Great efforts have been made on CCPP methods in known and stationary environments [9–16]. However, few methods pay attention to dynamic environments with non-stationary obstacles. Luo and Yang [17] proposed a bio-inspired method where the dynamics of each position on the map is topologically organised in a network. However, in the Luo's method, the robot is easily trapped in a situation named deadlock where all the neighbouring locations are neither obstacles nor visited locations. The network inspired by the biologic shunting model is lack of global information to escape from deadlocks quickly.

Both the single-robot based approaches and the approaches using multiple robots have been developed, such as Multi-Robot Spanning Tree Coverage (MSTC) [18], Multi-Robot Forest Coverage [19], Backtracking Spiral Algorithm Cooperative Multi-robot [20] and Boustrophedon and Backtracking mechanism [21]. These approaches reduce the coverage time in general.

In this paper, a priority template and a novel global backtracking mechanism are proposed to coverage task in a dynamic environment. To deal with deadlock problems, where all the neighbouring positions of the robot are neither obstacles nor visited locations, the robot uses a priority template to reduce deadlocks by limiting the uncertainty of directions. Meanwhile, the robot activates the global backtracking mechanism to escape from deadlocks when it is stuck. It determines the best backtracking point using a greedy criterion and plans an optimal path. Our method generates the path by incrementally repairing the path costs as new information discovered. Moreover, the proposed method can be extended to the multi-robot system, different from the single-robot system. Our multi-robot mechanism determines the best backtracking point by starting a market-based bidding process [22] among robots. The bidding process takes into account both the length of the path and the conflict with the others.

The remaining of this paper is organised as follows. Section 2 describes the assumptions, definitions and notations. Section 3 presents the proposed method including four parts, the introduction of short-term memory CCPP model, the global backtracking mechanism dealing with deadlocks, multi-robot extension by the market bidding process and the analysis of the proposed neural-dynamics-based approach. Experiments in various dynamic scenes are implemented in Section 4. Section 5 concludes our work.

2 Assumptions, definitions and notations

Some notations, assumptions and definitions in this paper are described as follows:

2.1 Assumptions

(i) The robot obtains information on a limited range and locates itself accurately or within a tolerable error, uncertainties via its on-board sensors (e.g. radar, laser-scanner and sonar), multi-sensory fusion and simultaneous localisation and mapping are beyond our scope.
(ii) The robot can move to eight potential directions (front, back, left, right, front left, front right, back left, and back right) as a real

cleaning robot works in indoor environments. The algorithm also has the latent capacity to apply on three-dimensional (3D) workspace as long as extending directions with up and down.
(iii) In multi-robot systems, communication between robots guarantees that each robot can share its position with the others and treat the other robots as obstacles.

2.2 Definitions

(i) The point (x, y) indicates the coordinate position of the robot in the workspace. The autonomous mobile robot is able to turn on the spot.
(ii) The 2D Cartesian workspace is modelled as an occupancy grid incrementally as the robot moves. Each grid, being regarded as a neuron, is the same size as the robot.
(iii) There are four states of neurons introduced to the CCPP algorithm, which are visited, unvisited, obstacle, and deadlock. In deadlock state, the neighbours around current position are either visited or obstacles [17]. The position where the robot arrives in a deadlock situation is defined as the deadlock position.
(iv) The backtracking position is defined as the point which has more than one unvisited neighbour, that is to say, it can be a start point of the future coverage.

2.3 Notations

Here are some notions of the proposed method.

$\mathrm{BT}_{\mathrm{list}}$	list of backtracking points
S_{dl}	start deadlock point
G_{bt}	goal backtracking point
R_{dl}	robot who arrives in a deadlock situation
p	candidate point of G_{bt}
$D_{\min}$	minimum distance between the p and the robot

3 Proposed method

The framework of the proposed approach is illustrated in Fig. 1 with a single robot case as an example.

The prior knowledge of the environment is completely unknown, by gathering the local information via sensors. The robot is expected to cover all accessible grids. The first two subsections describe the proposed CCPP algorithm in single robot case. In the bio-inspired motion coverage process, the robot covers unvisited regions by calculating the activities of the neuron network, the network was constructed by the short-memory CCPP method and with a prior template to reduce deadlocks. When a deadlock situation occurs, the backtracking process actives, this process including three steps: (i) searching for candidate backtracking points and updating the backtracking list; (ii) selecting the best backtracking point; (iii) planning the shortest path from a deadlock point to a backtracking point. The coverage process will not finish until all points have visited. What's more, in Section 4, we extend the approach to the multi-robot case with a market-bidding process. Overall, the analysis of the approach in a dynamic environment is given in the last subsection.

3.1 Shunting short-memory-based coverage path planning with a prior template

A human brain could use the short-memory model [23] to deal with the information in a dynamic environment. We adapt the model to path planning problems, especially in the coverage task.

3.1.1 Landscape of neural activities: The core of the algorithm is to propose a neural network for the coverage task. The neural activities could represent the coverage state of the robot. Through the neural activity propagation, the robot will be attracted by the activities of neighbour neurons, in the way that inspired by shunting short-term memory mechanism.

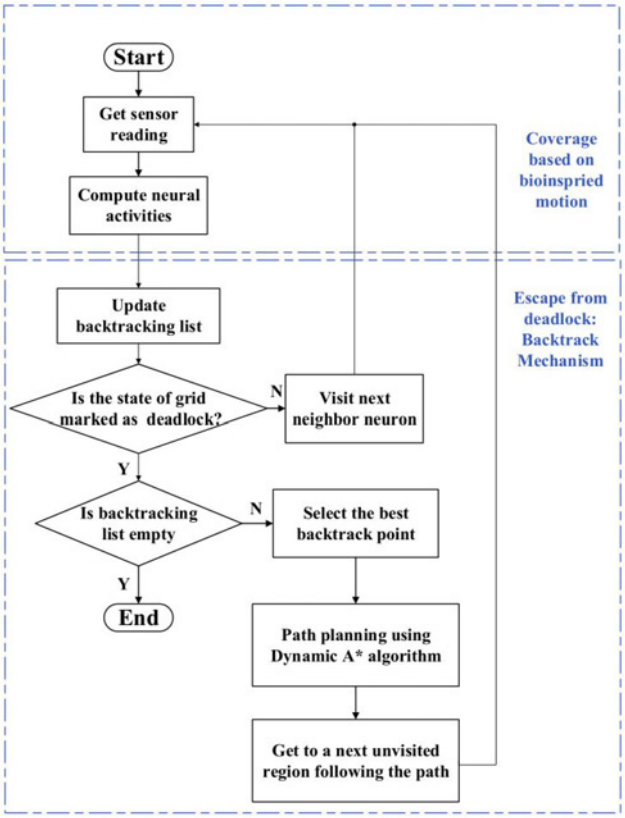

Fig. 1 *Flowchart of the proposed method*

A short-memory model is introduced by Hodgkin and Huxley [23]. It describes how information to transport between the paths of membranes in a biological neural system of humans. This model understands the real-time adaptive behaviour of individuals to complex and dynamic environmental contingencies. In [23], the neural activities across the membranes are described as follows:

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = -Ax_i + (B - x_i)\left([I_i]^+ + \sum_{j=1}^{k} \omega_{ij}[x_j]^+\right) - (D + x_i)[I_i]^- \quad (1)$$

Parameters A, B and D are non-negative constants representing the passive decay rate, and the upper and lower bounds of the neural activity, respectively. k is the number of neural connections from the ith neuron to its neighbours. Here, the term $[I_i]^+ + \sum_{j=1}^{k} \omega_{ij}[x_j]^+$ is the excitatory input and $[I_i]^-$ is the inhibitory input.

We adapt this biological model to the robot coverage path planning task. The architecture of the CCPP neural network is shown in Fig. 2*a*.

x_i and x_j are neural activities of the central and neighbouring neurons, respectively, with a radius r_0. Central neuron represents the current location of the robot, each central neuron has only local lateral with k connections where k represents the number of directions that the robot could move. In this case, k is 8. ω_{ij} is the connection weight between central neural and its neighbour. Especially, in our CCPP neuron network, we define the external input I_i to the ith neuron as follows:

$$I_i = \begin{cases} E, & \text{if it is an unvisited region} \\ -E, & \text{if it is an obstacle region} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where $E \gg B$ is a very large positive constant, which guarantees that unvisited regions attained at the peak of the landscape and obstacles

CAAI Trans. Intell. Technol., 2018, Vol. 3, Iss. 1, pp. 65–72

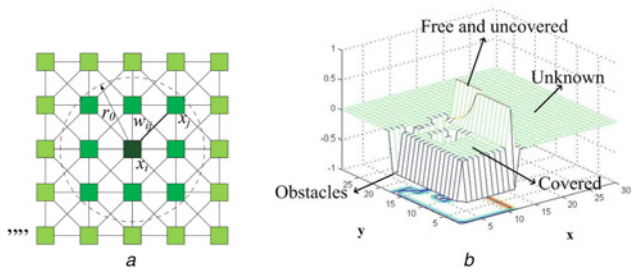

Fig. 2 *Landscape of neural activities*
a Architecture of a neural network with neighbouring neurons
b Neuron activities of robot different states

stay in the valley, as shown in Fig. 2*b*. The connection weight ω_{ij} between the ith and jth neurons can be defined as $\omega_{ij} = f(|\boldsymbol{x}_i - \boldsymbol{x}_j|)$, where $|x_i - x_j|$ represents the Euclidean distance between vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the state space and $f(a)$ can be any monotonically decreasing function, such as a function defined in this paper

$$f(a) = \begin{cases} \mu/a, & \text{if } 0 < a \le r_0 \\ 0, & \text{if } a > r_0 \end{cases} \tag{3}$$

where μ and r_0 are positive constants. The proposed method guarantees the positive neural activity propagates globally, while the negative activity only stays local.

3.1.2 Prior template: In the proposed method, we add a prior template in the process of coverage planning in order to reduce the uncertainty of directions and make the path grow like repeated mowing when the neighbour neurons have the same activities. The regularity in our prior template is the up and down. This template is triggered when the activities of neighbour neurons have more than one in rank one class after updating. In the rand one class, the robot decides to move left or right unless there is no up-and-down direction. This template is very effective in complex environments, making the path more regularly and reduces the number of deadlocks.

3.2 Global backtracking mechanism

We active a global tracking mechanism to escape from the deadlock situation quickly. Our backtracking mechanism mainly lays in two aspects: First, while updating the $\mathrm{BT}_{\mathrm{list}}$, a restriction is imposed according to the spatial characters. Second, a greedy criterion is used to select the best backtracking point, and then robot escapes from deadlocks by dynamic A* algorithm straightly.

3.2.1 Updating the backtracking list: Backtracking points are the points which have more than one unvisited neighbouring neurons, that is to say, the potential backtracking point can be a starting point for next coverage path. Backtracking list is a list of backtracking points which are updated as the robot moves. Once the robot moves, the states of the eight neighbours update. An unvisited point will be marked as backtracking point and added into the backtracking list. As shown in Algorithm 1 (see Fig. 3). Neighbour points are the 8 points around the current point (x, y). State (*position*) denotes the state of a neuron at the position. Activity (*position*) returns the neuron activity of this position. Select (*backtrackinglist*) selects the best point for backtracking list. Move (*current*, *target*) is a point-to-point planner.

As shown in Fig. 4, the current neuron X is coloured in black and the blue dashed circle represents the reception field. Its neighbours which have already been visited are coloured in grey, and occupied by the obstacles are dashed. The neighbours $X3$, $X4$ and $X5$, coloured in white, are free and unvisited. Therefore, the neuron X is a backtracking point candidate, which should be added to the backtracking list. When the robot moves, the current

Input: *CurrentPoint*(x, y)
Output: the new backtracking list
1 **for** *NeighbourPoints around CurrentPoint* **do**
2 **for** *NeighbourPoints around CurrentPoint* **do**
3 Check State;
4 **if** *State(NeighbourPoints)= Unvisited* **then**
5 UpdateBacktrackinglist(AddCurrentPoint);
6 **if** *all State(NeighbourPoints)= Visit or Obstacle* **then**
7 UpdateBacktrackinglist(RemoveCurrentPoint);

Fig. 3 *Algorithm 1: Updating backtracking List*

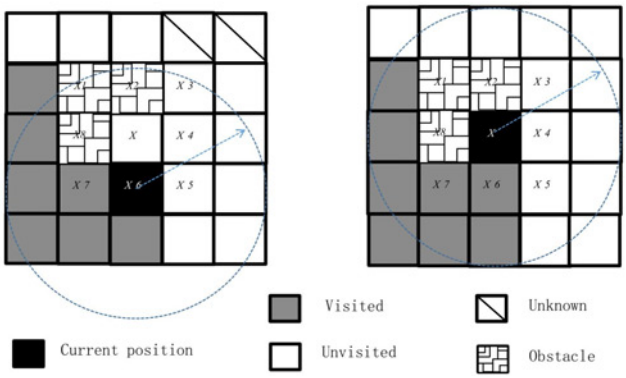

Fig. 4 *Illustration of backtracking list updating*

position effects on the states of eight neighbours, so we also update the state of neighbours instantly. As the top picture to the bottom picture in Fig. 4 shows the robot moves from $X6$ to X, the points X, $X1$, $X7$, $X8$ are removed from the backtracking list, the points $X2$, $X3$, $X4$, $X5$, $X6$, including X are kept in the backtracking list.

3.2.2 Deadlock detection and escaping: We introduce a novel deadlock detection phase to detect the deadlock. It checks the state of the grid around the current grid, when the state is visited or an obstacle, while the activities around the current position are lower than the activity of the current grid, it meets deadlock situation. Note that when the robot has finished the coverage, the robot will also be in a deadlock situation, but the backtrack list is empty this time. This deadlock detection phase is given in Algorithm 2 (see Fig. 5). The phase is always on during the coverage until the end. To choose the best backtracking point, we choose the newest point in the backtracking list as the goal, directly go there using dynamic A* which could provide an optimal path.

3.3 Extend proposed method to multi-robot

In multi-robot systems, one robot needs to move as far as possible away from the other robots to avoid conflict and replanning. However, if the robots are close to each other, the proposed method with a market-based bidding process is performed to ensure collision avoidance.

In single-robot systems when the robot drops into backtracking point selecting process, we aim at choosing the best backtracking point as G_{bt} for the deadlock robot R_{dl}, the most recent point p in the $\mathrm{BT}_{\mathrm{list}}$ is chosen as G_{bt} according to the above discussions. However, while in multi-robot systems, the selection is more complex and can be regarded as a task allocation problem. The market-based architecture is proved to be an effective distributed mechanism for multi-robot task allocation [24], in which each

robot works independently. In our market-based bidding process, each robot selects the best backtracking point which satisfies both conditions: (i) close to the deadlock robot; (ii) far away from the other robots.

The market-bidding process work as follows: the most recent point p is just a candidate. Moreover, G_{bt} should be away from the other robots so that it will not be covered by the others in a short time. Every robot computes its Euclidean distance to p as a tender price. If the tender price of the bidder R_{dl} is lower than any other robots, R_{dl} will win the right to cover p and the region around it. In other words, p is the G_{bt} and the bidding process is complete. On the contrary, if the tender price of R_{dl} is not the lowest price, R_{dl} must select the next recent point in the BT_{list} and start a bidding process again. However, if all the points in the BT_{list} have already been considered and none of them satisfies the above condition, the most recent point in the BT_{list} is chosen as G_{bt}. See Algorithm 3 (see Fig. 6).

Input: backtrack list
Output: the best backtrack point for robot to escape from deadlock

```
for NeighbourPoints around the CurrentPoint do
    Check State;
    if State(NeighbourPoints) = Visited or Obstacle then
        Check Activity;
        if Activity(CurrentPoint) > Activity(NeighbourPoints) then
            State(CurrentPoint) = Deadlock;
            if BacktrackList = EMPTY then
                End CCPP;
                BREAK;
            else
                BestPoint=Select(Backtracklist);
                Move(CurrentPoint, BestPoint);
```

Fig. 5 *Algorithm 2: Deadlock detection and escaping*

Input: Set of robots, BT_{list};
Output: G_{bt} of R_{dl};

```
Choose the most recent point p in the BT_list as a candidate G_bt;
for each robot do
    Compute the distance between the robot and G_bt;
    Find the minimum distance D_min;
    if the distance between R_dl and G_bt equals to D_min then
        return G_bt;The best backtrack point is found;
    else
        Choose the next recent point in the BT_list as G_bt;
return the most recent point p as G_bt;
```

Fig. 6 *Algorithm 3: Bidding process in multi-robot systems*

4 Experiments and discussions

In the experiment section, we analyse the performance of our method on complete coverage path planning, three groups of experiments are conducted to evaluate the time efficiency, coverage rate and overlap rate among methods, two experiments use the single robot, one experiment uses multi-robot. Moreover, we implement the proposed method on the real robot in an apartment-like scene.

4.1 In unknown apartment-like workspace for single robot

The first experiment is implemented to demonstrate the complete coverage in an unknown environment. In the beginning, the robot has no prior knowledge about the workspace. The sensor range is twice the size of the robot. Figs. 7 and 8 show that the proposed method performs well both in easy and complex scenes.

Take Fig. 8*a* as an example. When the robot reaches a point $M(10, 8)$ in Fig. 8*a*-1, the dynamic activity landscape is illustrated in Fig. 8*a*-2 with visited, unvisited, obstacle and unknown regions. The neural activities of unvisited areas have very large values represented by peak, the negative values represented the obstacles. When the robot reaches a point $N(4, 28)$ in Fig. 8*a*-1, i.e. a deadlock situation where the neural activities of eight neighbours are all lower than the current neuron N. In the previous method, it is time consuming that neuron waits for the decay of neighbour neural activities. In the proposed method, the backtracking mechanism is activated here to plan the shortest path directly by using the backtracking list. The robot constructs and updates the backtracking list based on the accumulated information, and selects the best backtrack point $P(4, 22)$ by choosing the most recent point in backtracking list. Then, the dynamic A* algorithm is called to plan a short path. When the robot reaches $P(4, 22)$, a new coverage will begin. The final path is shown in Fig. 8*a*, which illustrates that the proposed method can cover every accessible grid in the workspace.

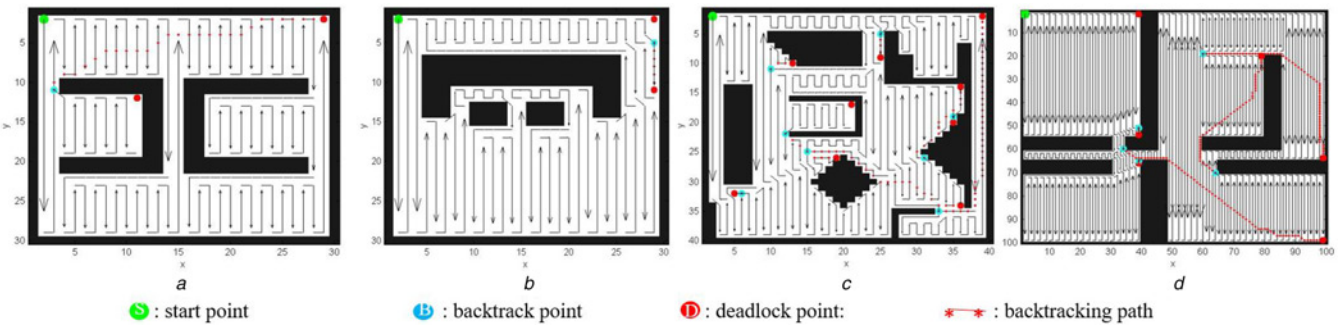

Fig. 7 *Experiment scenes. The back area shows the obstacles, the line with arrowheads represents the coverage paths which calculated by the proposed method. Green point is the start point, blue points are backtracking point, red points are deadlock points, right lines are the backtracking paths. There are four scenes*
a Double *H* space
b Indoor apartment-like scene including a sofa, which can be regarded as an *H* space
c Scene has complex obstacles
d Large scene four times bigger than the others

CAAI Trans. Intell. Technol., 2018, Vol. 3, Iss. 1, pp. 65–72

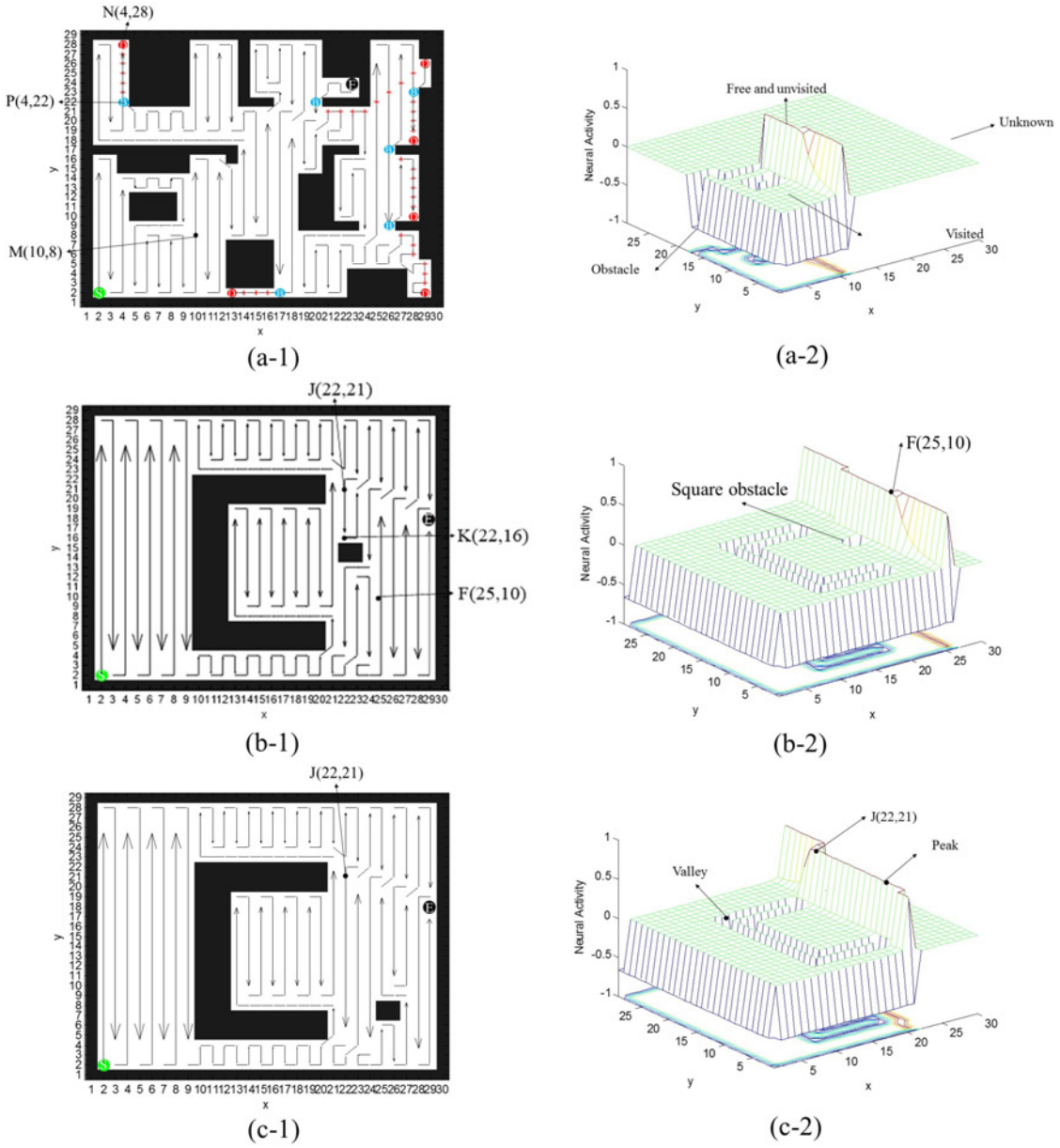

Fig. 8 *Complete coverage path and its neural activity landscapes in the robot current position*
a Proposed complete coverage planning in an unknown environment
a-1 Path generated by the proposed method
a-2 Neural activity landscape when the robot reaches $M(10, 8)$.
b Proposed complete coverage planning in an unknown environment with a non-stationary obstacle
b-1 and *c*-1 Paths with a C-shaped and a non-stationary square obstacle
b-2 and *c*-2 Neural activity landscapes when the robot reaches $F(25, 10)$ and $J(22, 21)$

4.2 In the workspace with non-stationary obstacles

This group of simulation is conducted to verify the effectiveness of the proposed method in the workspace with non-stationary obstacles. A square obstacle can be detected when it appears within a certain range of detection. The obstacle is introduced into the coverage process of simulation *A*, as shown in Figs. 8*b* and *c* and backtracking process in simulation *B*, as shown in Fig. 9. If an obstacle moves to another position which is out of detection, the robot will not cover that region for lack of knowledge. That is beyond the scope of this paper.

Simulation *A* is also performed in the workspace with a C-shaped obstacle and a square obstacle, as shown in Figs. 8*b*-1 and *c*-1. When the robot reaches a point $J(22, 21)$, it has not yet detected the square obstacle. The robot plans the path from $J(22, 21)$ to the bottom side. Once the robot reaches the point $K(22, 16)$, the square obstacle moves in front of the robot (compared Fig. 8*b*-1 with Fig. 8*c*-1).

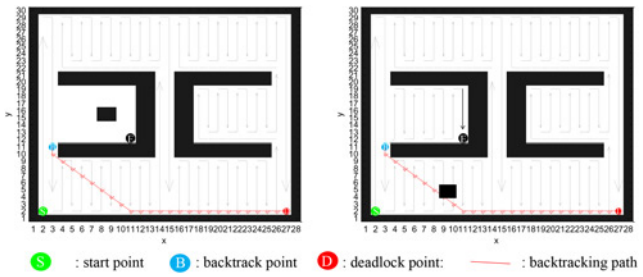

Fig. 9 *Path using the proposed method in unknown environments with a non-stationary obstacle. Left: The path before detecting the squared obstacle. Right: The path after updating the costs*

The robot cannot detect the square obstacle until it is close enough (i.e. with a radius of r_0 shown in Fig. 4). The neural activities of the positions occupied by the square obstacle decay sharply in the valley, as shown in Figs. 8*b*-2 and *c*-2. The robot cannot move forward, and hence turn left to avoid the obstacle.

Simulation *B* tests the situation where the obstacle appears during the backtracking process. When the robot reaches the deadlock point $V(27, 2)$, it chooses the best backtracking point $U(3, 11)$, as shown in Fig. 9. By applying the dynamic A* algorithm, as the robot moving along the path, it acquires sensor measurements and updates the costs. When the robot discovers the square obstacle, it repairs the costs and searches for a new shortest path from the current point to the goal point. The final path is shown in Fig. 9.

4.2.1 Time consumption analysis: To show the time efficiency, we compared proposed methods with backtracking mechanism or without the backtracking mechanism in four dynamic scenes. These scenes are from easy to difficult, as shown in Fig. 7. From Table 1, we can see that the more complex the scene is, the more deadlocks it has the better proposed method performs. In Fig. 7*c*, backtracking mechanism saves 70% time. A backtracking mechanism needs extra little more storage to save backtracking list.

4.2.2 Coverage rate, overlap rate and path length analysis: To verify the efficiency of the proposed algorithm, we conduct two versions of the proposed method (with and without backtracking). Three challenging target scenes are tested, which are shown in Fig. 7. For each scene, 30 different initial poses were randomly selected to get the generalised performance of the approaches. As shown in Table 2, the proposed method performs much better than the method without backtracking in terms of coverage rate and path length. Our backtracking mechanism uses a prior template can get fewer deadlocks. When a deadlock happens, our method including an optimal point-to-point path planning, which we get a shorter path length.

4.3 Experiments on multi-robot

In this subsection, the proposed method is extended to multi-robot systems which distributes the workload to multiple robots and reduces the coverage time overall. In multi-robot systems, one robot needs to move as far as possible away from the other robots to avoid conflict and replanning.

Fig. 10 depicts the complete coverage paths generated by four robots using the proposed method. The green, blue, orange and

Table 1 Time consuming performance of proposed methods, with or without proposed backtracking mechanism

Approach	Escaping time from deadlock(s)		Time-to-completion(s)	
	Easy scene	Hard scene	Easy scene	Hard scene
without backtracking	1.21	10.22	34.77	160.62
with backtracking	0.38	0.51	32.51	144.35
Improvement, %	68.59	95.01	6.49	10.1

Table 2 Quantities measured during the process of three algorithms

Approaches	Quantity			
	Coverage rate, %		Path length, cm	
	Mean	Std.	Mean	Std.
STC	73.8	4.2	420	13
without backtracking	97.6	0.6	670	18
with backtracking	98.9	0.4	641	12

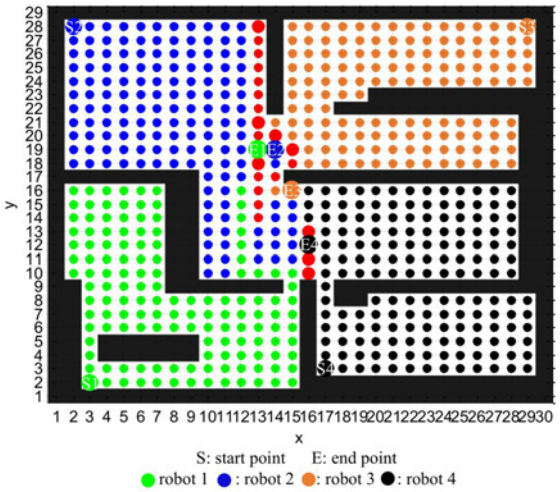

Fig. 10 *Coverage paths performed by the proposed method with four robots. The red points are the revisited points*

black points are covered by robot 1, 2, 3 and 4, respectively. Red points are the points in the backtracking paths which are covered more than once.

We can clearly see that the algorithm is complete which covers all the accessible grids and the workload of each robot is almost balanced. Time taken by the robots is 8.86, 9.25, 9.17 and 9.09 s, respectively. However, for single-robot coverage, it costs 33.92 s. As a result, the coverage time is significantly reduced by deploying multiple robots.

Due to the market-based task allocation, each robot covers its area separately with few conflicts. If some robots fail, all the unvisited grids which are accessible will be covered by the other living robots. Therefore, robustness and completeness can be guaranteed.

To compare the performance of our approach with two versions of the MSTC, non-backtracking MSTC (NB-MSTC) and backtracking MSTC (B-MSTC), the total length of the coverage paths, the workload distribution of the robots and the repetition rate are calculated. Coverage paths are illustrated in Fig. 11 and their length are shown in Table 3. Since the spanning tree algorithm divides the workspace into grids of a size $4D$ where D is the size of the robot, some grids along obstacles do not belong to the graph grid used to construct the tree. As a result, MSTC cannot cover these grids (see Fig. 11), the proposed method does not have this problem due to the grid size of D. Task allocation in our approach is more balanced than two versions of MSTC, as shown in Table 3, as the backtracking mechanism assigns backtracking points to the robots until the task is complete. The coverage time (the maximum coverage time of the robots) using the proposed method is 30.31 s which is less than NB-MSTC (43.16 s) and B-MSTC (34.24 s).

However, due to the backtracking, the repetition rate of our approach is higher than NB-MSTC but lower than B-MSTC. Considering the better performance of the proposed method on the task allocation and coverage rate, the repetition rate is tolerable in real-world applications.

4.4 Tests on real cleaning robot

Experiments in a real environment are also provided. The results are shown in Fig. 12.

The entire approach has been embedded into a laptop while a ceiling camera is used to extract the ground truth of the robot. The indoor home-like environment is 5 m by 5.5 m. The robot contains three infrared sensors to avoid the forward bump. A laser on the robot provides information on the surroundings. The accuracy of the robot poses estimation is a key factor affecting the overall performance of the approach. Odometry system is designed combining the measurements of the encoder and gyro which limits

CAAI Trans. Intell. Technol., 2018, Vol. 3, Iss. 1, pp. 65–72

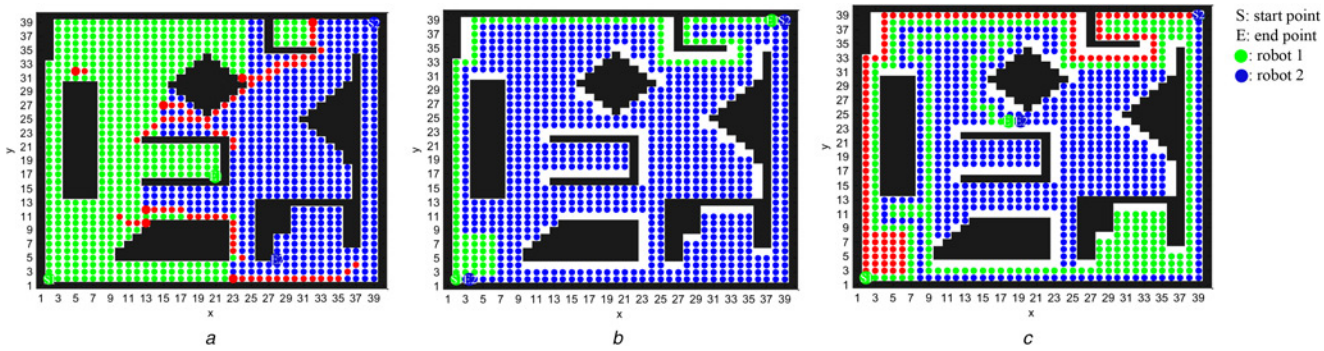

Fig. 11 *Coverage paths using the proposed method and two versions of MSTC in multi-robot systems. The red points are the revisited points*
a BNNB
b NB-MSTC
c B-MSTC

Table 3 Performance comparison between proposed method and two versions of MSTC

Approaches	Number of grids	Coverage rate of R_1, %	Coverage rate of R_2, %	Repetition rate, %
ours	**1126**	**50.1**	**49.9**	**1.9**
NB-MSTC	982	12.6	87.4	0.0
B-MSTC	982	32.3	67.7	7.0

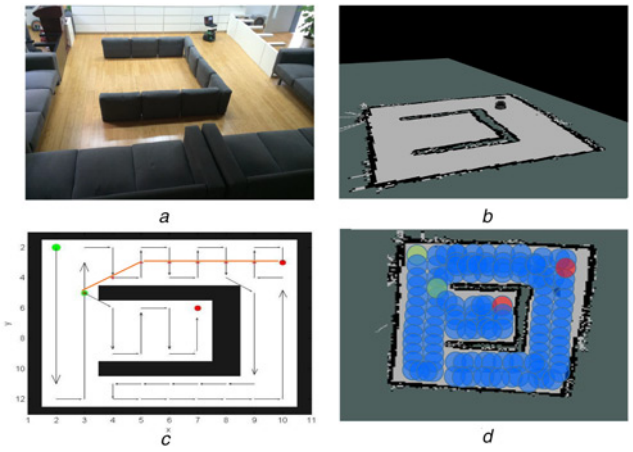

Fig. 12 *Real world experiments*
a Real apartment-like scene
b Robot view of the scene
c Proposed coverage path in the simulation
d Coverage path shows in the sketch map

the estimation error within a tolerable range which is defined as the half size of a grid. The averages of the time-to-completion and coverage rate are 5 min and 92.21%, respectively. These results show that the proposed method is also applicable to cluttered indoor environments.

5 Conclusions

We present here a sensor-based CCPP algorithm which combines a bio-inspired shunting short memory method with a backtracking mechanism and prior template, dealing with a deadlock situation. Instead of waiting for the decay of activities, a backtracking mechanism is activated that it selects the best backtracking point efficiently and plans the shortest path. Experiments show that the proposed method performs more efficiently in most workspaces in terms of the coverage time and the length of coverage paths. Besides, the proposed method is extended to multi-robot systems with a market-based bidding process, and the workloads are deemed more balanced than other multi-robot approaches. For future research, we plan to consider the energy and timing constraints that allow robots to carry limited energy and to complete the coverage task before the deadline.

6 Acknowledgment

This work was supported by the National Natural Science Foundation (NSFC, nos. 61340046, 61673030, U1613209), Natural Science Foundation of Guangdong Province (no. 2015A030311034), Scientific Research Project of Guangdong Province (no. 2015B010919004), Specialized Research Fund for Strategic and Prospective Industrial Development of Shenzhen City (no. ZLZBCXLJZI20160729020003), Shenzhen Key Laboratory for Intelligent Multimedia and Virtual Reality (ZDSYS201703031405467).

7 References

[1] Galceran, E., Carreras, M.: 'A survey on coverage path planning for robotics', *Robot. Auton. Syst.*, 2013, **61**, (12), pp. 1258–1276
[2] Kurabayashi, D., Ota, J., Arai, T., *et al.*: 'Cooperative sweeping by multiple mobile robots'. Proc. Int. Conf. Robotics and Automation (ICRA), Minneapolis, MN, USA, 1996, pp. 1744–1749
[3] Svennebring, J., Koenig, S.: 'Building terrain-covering ant robots: a feasibility study', *Auton. Robots*, 2004, **16**, (3), pp. 313–332
[4] Acar, E.U., Choset, H., Zhang, Y., *et al.*: 'Path planning for robotic demining: robust sensor-based coverage of unstructured environments and probabilistic methods', *Int. J. Robot. Res.*, 2003, **22**, (7-8), pp. 441–466
[5] Hameed, I., Bochtis, D., Sørensen, C.A.G.: 'An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas', *Int. J. Adv. Robot. Syst.*, 2013, **10**, (231), pp. 1–9
[6] Ollis, M., Stentz, A.: 'First results in vision-based crop line tracking'. Proc. Int. Conf. Robotics and Automation (ICRA), Minneapolis, MN, USA, 1996, pp. 951–956
[7] Galceran, E., Carreras, M.: 'Efficient seabed coverage path planning for asvs and auvs'. 2012 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), Vilamoura, Portugal, 2012, pp. 88–93
[8] Hess, J., Beinhofer, M., Burgard, W.: 'A probabilistic approach to high-confidence cleaning guarantees for low-cost cleaning robots'. Proc. Int. Conf. Robotics and Automation (ICRA), Hong Kong, 2014, pp. 5600–5605
[9] Ramaithitima, R., Whitzer, M., Bhattacharya, S., *et al.*: 'Sensor coverage robot swarms using local sensing without metric information'. Proc. Int. Conf. Robotics and Automation (ICRA), Seattle, Washington, USA, 2015, pp. 3408–3415
[10] Galceran, E., Campos, R., Palomeras, N., *et al.*: 'Coverage path planning with realtime replanning for inspection of 3d underwater structures'. Proc. Int. Conf. Robotics and Automation (ICRA), Hong Kong, 2014, pp. 6586–6591
[11] Gabriely, Y., Rimon, E.: 'Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot'. Proc. Int. Conf. Robotics and Automation (ICRA), Washington, USA, 2002, pp. 954–960
[12] Gonzalez, E., Alvarez, O., Diaz, Y., *et al.*: 'BSA: A complete coverage algorithm'. Proc. Int. Conf. Robotics and Automation (ICRA), Barcelona, Spain, 2005, pp. 2040–2044

[13] Viet, H.H., Dang, V.-H., Laskar, M.N.U., *et al*.: 'BA*: an online complete coverage algorithm for cleaning robots', *Appl. Intell.*, 2013, **39**, (2), pp. 217–235
[14] Gabriely, Y., Rimon, E.: 'Spanning-tree based coverage of continuous areas by a mobile robot', *Ann. Math. Artif. Intell.*, 2001, **31**, (1-4), pp. 77–98
[15] Hart, P.E., Nilsson, N.J., Raphael, B.: 'A formal basis for the heuristic determination of minimum cost paths', *IEEE Trans. Syst. Sci. Cybern.*, 1968, **4**, (2), pp. 100–107
[16] Acar, E.U., Choset, H.: 'Sensor-based coverage of unknown environments: incremental construction of Morse decompositions', *Int. J. Robot. Res.*, 2002, **21**, (4), pp. 345–366
[17] 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.*, 2008, **19**, (7), pp. 1279–1298
[18] Hazon, N., Kaminka, G.: 'Redundancy, efficiency and robustness in multi-robot coverage'. Proc. Int. Conf. Robotics and Automation (ICRA), Barcelona, Spain, 2005, pp. 735–741
[19] Zheng, X., Jain, S., Koenig, S., *et al*.: 'Multi-robot forest coverage'. Proc. Int. Conf. on Intelligent Robots and Systems (IROS), Edmonton, Canada, 2005, pp. 3852–3857
[20] Gonzalez, E., Gerlein, E.: 'BSA-CM: a multi-robot coverage algorithm'. Proc. WI-IAT, Milan, Italy, 2009, **3**, pp. 383–386
[21] Viet, H.H., Dang, V.-H., Choi, S., *et al*.: 'BoB: an online coverage approach for multi-robot systems', *Appl. Intell.*, 2015, **42**, (2), pp. 157–173
[22] Dias, M.B., Stentz, A.: 'A comparative study between centralized, market-based, and behavioral multirobot coordination approaches'. Proc. Int. Conf. Intelligent Robots and Systems (IROS), Las Vegas, Nevada, USA, 2003, **3**, pp. 2279–2284
[23] Hodgkin, A.L., Huxley, A.F.: 'A quantitative description of membrane current and its application to conduction and excitation in nerve', *J. Physiol.*, 1952, **117**, (4), p. 500
[24] Dias, M.B.: 'Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments'. Ph.D. thesis, Carnegie Mellon University Pittsburgh, 2004