KICS The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express xxx (xxxx) xxx

www.elsevier.com/locate/icte

Complete coverage path planning scheme for autonomous navigation ROS-based robots

Shengmin Zhao, Seung-Hoon Hwang*

Division of Electronics and Electrical Engineering, Dongguk university, Seoul, Republic of Korea

Received 4 February 2023; received in revised form 4 June 2023; accepted 29 June 2023
Available online xxxx

Abstract

In this paper, a new Complete Coverage Path Planning (CCPP) scheme is proposed which combines path planning and dynamic tracking for robot operating system-based robots. For the path planning, firstly a sub-area division algorithm is considered to decompose the occupancy map according to the wall or obstacle position after simultaneous localization and mapping process. For each sub-area, an "S" shape path planning is employed, and then a Bidirectional A-star connects them. Additionally, the dynamic tracking ensures that the robot moves continuously. Simulation results show that the coverage ratio of the planning path is improved as 98% by the proposed scheme.

Keywords: Complete coverage path planning (CCPP); Robot operating system (ROS); Simultaneous localization and mapping (SLAM); Coverage path method; Dynamic tracking method

1. Introduction

With the development of intelligent control technology, autonomous navigation robots have been widely used for various applications such as industrial manufacturing, agricultural irrigation, and floor cleaning [1–4]. Path planning are used by mobile robots, unmanned aerial vehicles, and autonomous cars in order to identify safe, efficient, collision-free, and least-cost travel paths from an origin to a destination [5]. The path planning algorithms of robots can be categorized into two types: Point-To-Point (PTP) and Complete Coverage Path Planning (CCPP). The task of the PTP is to determine a collision-free path from a start point to a finish point. The CCPP problem is the task of determining a path that passes over all points of an area or volume of interest while avoiding obstacles [6,7]. Since the PTP is focusing on the start and finish points, A-star (A*) algorithm is often adopted for autonomous navigation robot [8].

Several methods have been proposed to implement the CCPP. Liu introduced a random walk method where the robot can complete coverage with random heading changes [9]. However, this method may not achieve high coverage ratio due to the random movement. Stachniss proposed the "S" shape method which was the fastest process to cover the entire area [10]. Sewan presented a spiral algorithm which allowed the robot to create an increased circle [11]. When using the above algorithm, the robot may move repeatedly in each room of the environment so that it cannot cover and traverse each space in turn, which increases time consumption. Some studies have proposed combining machine learning with CCPP. Chen suggested a clustering-based approach to generate optimal flight paths for Unmanned Aerial Vehicles (UAVs) [12]. An adaptive clustering-based algorithm is proposed to efficiently provide optimal flight paths for UAVs in a given planning space, while satisfying the achievability, safety and optimality constraints [13]. Additionally, an ant colony system-based algorithm was presented to obtain efficient paths for UAVs that cover all regions adequately [14]. The above algorithm is designed to generate coverage paths for UAVs, not for ground mobile robots.

In this paper, a new CCPP scheme is proposed for ROS-based autonomous navigation robot that efficiently determine a coverage path in real-time with low time consumption. The scheme includes a path planning method and a dynamic tracking method. The specific contributions are as follows: (1). we proposed the sub-area division algorithm to decompose the overall environment into sub-areas according to the wall position of the environment to reduce time consumption by

* Corresponding author.
E-mail addresses: shengmin.zhao@dgu.ac.kr (S. Zhao), shwang@dongguk.edu (S.H. Hwang).
Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2023.06.009

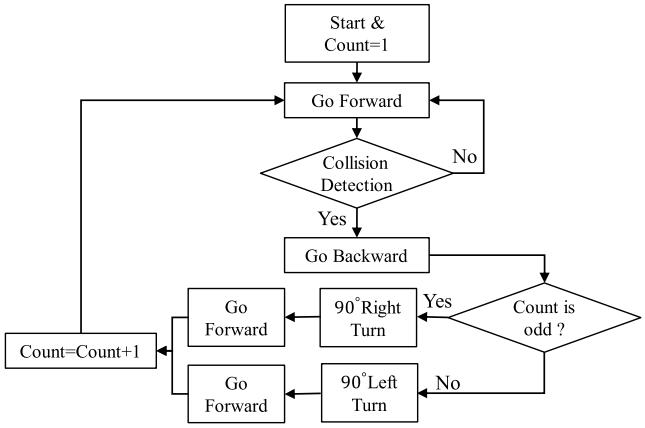

Fig. 1. Flow chart of the "S" shape algorithm.

repeated traversal. Within each sub-area, the density-adjustable 'S' method is designed to plan the path, which is then connected through the Bidirectional A* algorithm. (2). To further enhance movement efficiency in CCPP tasks, we propose a dynamic tracking method with the threshold for the ROS-based robot to avoid the time-consuming stop-and-go behavior commonly observed in navigation. It receives the path and odometer information by ROS topic and controls the robot moving effectively and uninterrupted.

2. Overview of technologies

2.1. "S" shape algorithm for CCPP

The algorithm follows a route map in the shape of the letter 'S' and is considered the fastest method for covering an entire room area [8]. For this algorithm, after every collision the robot has a sequence of movements direction of the robot continuously changes under this mode. The flowchart of the 'S' shape algorithm is given in Fig. 1. The robot moves forward until it encounters an obstacle, then rotates backward and chooses to move left or right based on whether the count is odd. These steps are repeated until the entire room is covered.

2.2. Bidirectional A* algorithm

The A* algorithm [8] is commonly used for path planning in Point-to-Point (PTP) systems, but it may not always provide the fastest planning time. The Bidirectional-A* (B-A*) algorithm offers a faster path planning speed, as demonstrated in Fig. 2. The robot plans the shortest path from the starting point X_1 to the target point X_2. First, the robot uses the A* algorithm to find two paths simultaneously. The first path starts from X_1 and ends at X_2. The other is the opposite. The algorithm runs until the explored node of one of the paths is in another set of explored node lists. Finally, according to the planning duration of the two paths, the fastest path is selected as the optimal path. In this example, the shortest path is represented by a solid line.

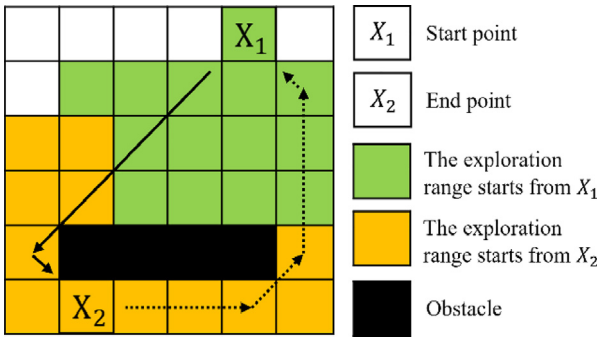

Fig. 2. Flow chart of the B-A* algorithm.

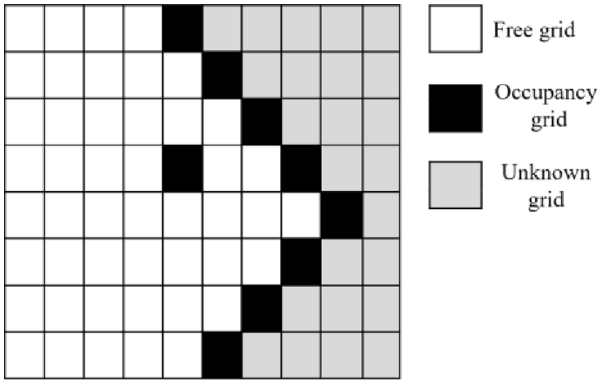

Fig. 3. Occupancy grid map and composition.

2.3. ROS and SLAM

One of the most popular applications of ROS is Simultaneous Localization and Mapping (SLAM). The objective of SLAM in mobile robotics is to construct and update a map of an unexplored environment using equipped sensors, such as 2D Lidar. The map constructed by the robot through 2D laser SLAM is a two-dimensional grid map represented by a grid composed of grids. The grid can store different values representing different meanings, as shown in Fig. 3. White represents a free, passable area, with a stored value of 0. Black represents an occupied, non-passable area, with a stored value of 100. Gray represents an unknown area, where it is not clear whether the grid is passable or not, with a stored value of -1. In ROS, the grid map is commonly referred to as an occupancy map as it depicts the occupied and unoccupied areas of a given environment.

3. Proposed CCPP scheme

A new CCPP scheme is proposed for ROS-based robots, which mainly includes two parts: path planning and dynamic tracking, as shown in Fig. 4. First, users need to initialize the coverage width w, which is the size of the grid the robot occupies. That is, the grid's length is the robot's diameter. The robot may be equipped with different equipment to adjust the parameters according to the actual situation. Then, set the following threshold μ, which is the distance between the actual position point and the target point. And then, the robot maps

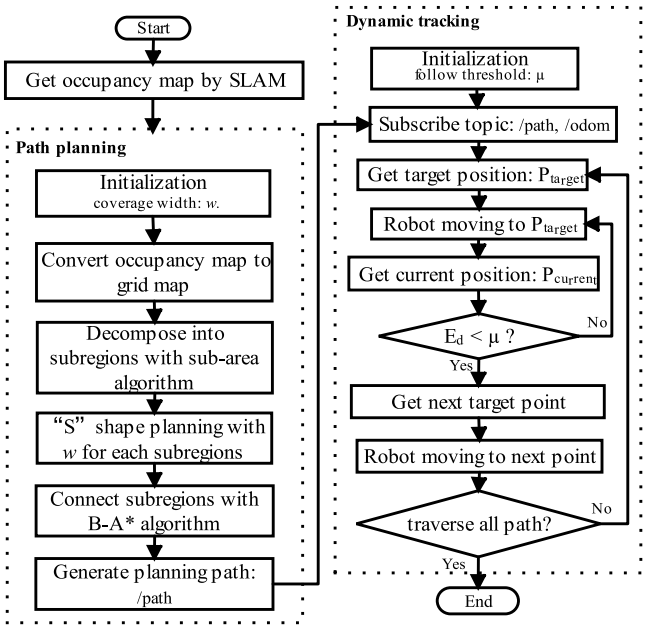

Fig. 4. The proposed CCPP scheme for ROS-based robot.

the environment through SLAM technology, which uses the explore_lite [15] function package.

3.1. Path planning method

The proposed CCPP scheme involves path planning method that generate coverage paths from an occupancy map of the environment. To control the density of planned paths, we added the coverage width w, which represents the size of the grid occupied by the robot in the grid map. For example, when the w value is 3, the "s" shape algorithm regards three parallel grids in the moving direction of the robot as a target point, and finally connects all points to generate a planned path. This value must be an odd number, because the planned path can be displayed exactly in the middle grid. This width can be adjusted based on the robot's equipment and the actual situation. When the environment has multiple complex rooms or obstacles are irregularly distributed, which can increase the robot's running time and difficulty. For example, with the "S" shape algorithm, the robot will turn left or right when encountering a wall. If there is no door between two opposing rooms, the robot can move back and forth between the two rooms and repeatedly pass through the connected area. It increases the time consumption, so doing the coverage traversal for each room one by one is more efficient. Therefore, we propose a sub-region decomposition algorithm for the path planning method before using the "S" shape planning algorithm, which decomposes the global path planning problem into multiple local path planning problems.

Algorithm 1: Proposed path planning method.

Input: Occupancy map: $\boldsymbol{M_{occp}}$.
Output: Planning path: /path (ROS topic).

w ← Coverage width
M_{grid} ← Grid map with only two values of 0 and 1, 0 means passable area, 1 means obstacle or wall
N_{row} ← The num of the rows of the grid map
N_{col} ← The num of the columns of the grid map
$C_{(x,y)}$ ← The coordinate value of the grid map
L_b ← List all boundary point
1. Initialize Coverage width: w;
2. Convert M_{occp} to M_{grid}
3. **for** m = 1 to N_{row} **do**
4. **for** n = 1 to N_{col} **do**
5. **while** $C_{(m,n)}$= 0 and $C_{(m,n+1)}$= 1 **do**
6. **for** j = m+1 to N_{row} **do**
7. **for** k = 1 to n+1 **do**
8. **while** $C_{(j,k)}$= 0 and $C_{(j,k+1)}$= 1 **do**
9. **if** k+1 < n+1 **do**
10. n ← k
11 repeat step 7
12. **else if** $C_{(j,k+1)}$=1 and $C_{(j+1,k+1)}$=0 or $C_{(j,k+1)}$=0 and $C_{(j+1,k+1)}$=1
13. **for** s = 1 to k **do**
14. **if** $C_{(j,s)}$! = 1 **do**
15. $C_{(j,s)}$←1
16. Store $C_{(j,s)}$ at L_b
17. **end**
18. **end**
19. **end**
20. **end**
21. **end**
22. **end**
23. **end**
24. **end**
25. **end**
26. Use "S" shape algorithm to planning for each subregion with w, get the partition path
27. Change all L_b from 1 to 0
28. Connect partition path by using the B-A* algorithm
29. Generate planning path and publish ROS topic:/path

The pseudocode of the path planning is shown in Algorithm 1, and its details are as follows:

- Step 1: Initialize the coverage width w. This is the grid size the robot can cover, which generally equal to the robot's diameter. The robot may be equipped with different equipment to adjust the parameters according to the actual situation. The size of M_{grid} is marked as N_{row} and N_{col}. The coordinate value of the M_{grid} is recorded as $C_{(x,y)}$.
- Step 2: Convert M_{occp} to M_{grid}. After the robot completes SLAM, it will get the occupancy map: M_{occp}, which contains accessible areas, obstacles, and unexplored areas. The unknown area is not considered; therefore, removing unknown area from M_{occp}. M_{grid} is a binary grid map, 0 indicates the accessible space, and 1 indicates an obstacle.
- Step 3: Sub-area division algorithm. Establishing x and y coordinate axes for M_{grid}; x represents the number of rows and y represents the number of columns. So, the upper left corner of the M_{grid} is the origin: $C_{(1,1)}$, and

the coordinates of the lower right corner are $C_{(N_{row},N_{col})}$. First, decompose the graph M_{grid}, that is, in the order from left to right and from top to bottom. Finding the minimum y coordinate values in each row when $C_{(m,n+1)}$ is 1. With $n+1$ as a fixed column, check the coordinate values of each row in turn from the range 0 to m. If there is a boundary point C(j,k+1), the values of the previous row and the next row in the same column are 0 or 1, respectively. Then, within the range of 1 to k columns, assign a value of 1 to all j-row boundary points. Save all boundary points in L_b.

- Step 4: "S" shape planning with w for each subregion, as shown in Fig. 1. During its process, we used the planning rules: Check whether the forward direction is passable. If there is an obstacle or if it has been visited, turn left or right . Then, get the partition path.
- Step 5: Change all L_b from 1 to 0 values.
- Step 6: Connect partition path by using the B-A* algorithm. After the sub-area traversal is completed, use the current position as the starting point and the no-traversal areas as the target point, and find the shortest path.
- Step 7: Repeat a Step 4, 5 6. If all areas are covered, then publish planning path by ROS topic:/path.

3.2. Dynamic tracking method

During a navigation task, a ROS-based robot typically has one target point and stops moving once it reaches it. However, when using the traditional navigation method for a CCPP task, the robot may experience stop-and-go behavior, resulting in significant time consumption. This is because the planned path contains multiple goal points, causing the robot to stop and restart multiple times. To address this inefficiency, a dynamic tracking algorithm has been proposed and implemented as ROS nodes for the robot's navigation, as depicted in Fig. 5. In this study, we propose the use of a fellow threshold μ to determine the distance between the actual robot position and the target point. When the robot is in motion and the distance from the current target point is less than μ, the system will immediately send the next target point command. This dynamic tracking approach ensures uninterrupted movement of the robot until it reaches the end. Our algorithm is designed to ensure smooth movement of the robot throughout the process, as described below:

- Step 1: Initialize the following threshold μ, which is the distance between the actual position point and the path planning point.
- Step 2: Subscribe the ROS topic:/path and/odom.
- Step 3: The robot sequentially receives the target points: P_{target} and current position: $P_{current}$. If the Euclidean distance between P_{target} and $P_{current}$ is less than the μ, the robot receives the next target value and moves towards it. If not, the robot continues to move towards the current target point.
- Step 4: Repeat step 3 until the robot reaches all target points in the/path.

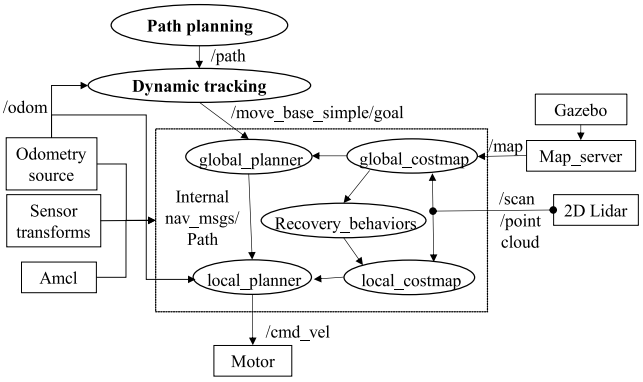

Fig. 5. Proposed ROS-based robot navigation framework for CCPP.

Algorithm 2.Proposed dynamic tracking method.

Input: Planning path: /path (ROS topic).

/odom ← ROS topic of robot odometer information
$P_{current}$← The current point in the /odom
P_{target} ← The target point in the /path
N_p ← The number of point in the /path
E_d ← Euclidean Distance between $P_{current}$ and P_{target}
1. Initialize fellow threshold: μ ;
2. Subscribe ROS topic: /path and /odom
3. **for** m = 1 to N_p **do**
4. Get P_{target}
5. Control robot moving towards P_{target}
6. Get $P_{current}$
7. **if** $E_d < \mu$ **do**
8. m ← m+1
9. **else**
10. repeat step 5
11. **end**
12.**end**

4. Simulation results

4.1. Simulation configuration

Work Environment: We set up three scenarios according to the parameters of the floor plan in Fig. 6(a). The scene size is 8 m×8 m, and there are four rooms and corridors. The wall size of each room near the hall is 3 m, and the width of the gallery is 2 m. At the same time, we arranged the robot in the lower left corner of room. The three scenes are all simulated environments produced by Gazebo software. Scenario 1 does not contain any obstacles. Scenario 2 and Scenario 3 contain a different number of obstacles in different positions.

Parameter setting: Faced with different tasks, the size of equipment carried by the robot is also different. We choose coverage widths w are 3 and 5 to simulate different tasks. The following thresholds μ are set to 0.1 m and 0.2 m. Therefore, there are four cases of parameter settings in three different

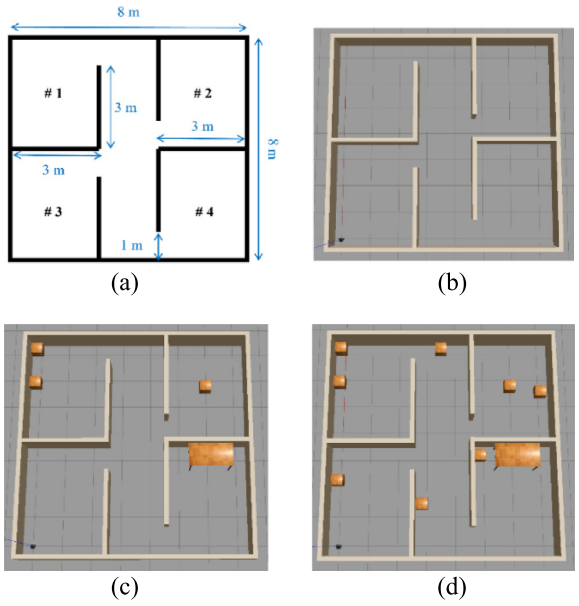

Fig. 6. Simulation environment and different scenarios. (a). Floor plan with size parameter, (b). Scenario 1, (c). Scenario 2, (d) Scenario 3.

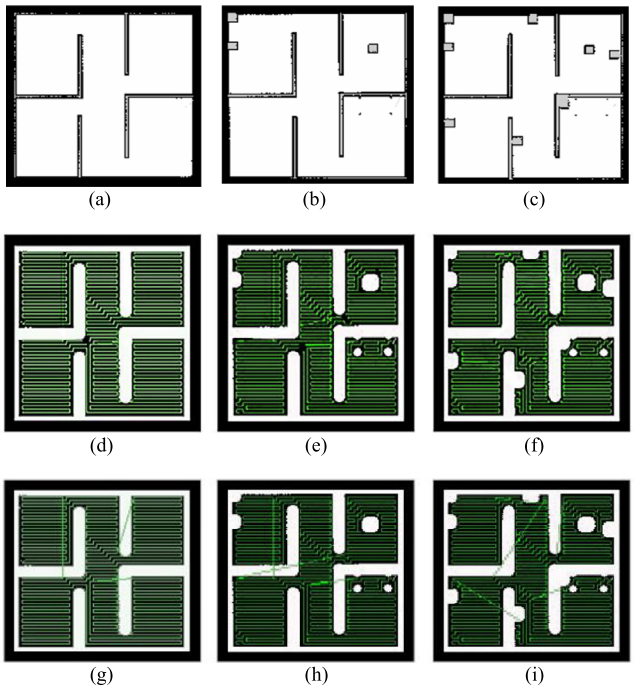

Fig. 7. SLAM, sub-area division, and path planning results in three different scenarios. (a) SLAM result for scenario 1, (b) SLAM result for scenario 2, (c) SLAM result for scenario 3, (d) Sub-area division result for scenario 1, (e) Sub-area division result for scenario 2, (f) Sub-area division result for scenario 3, (g) Planning path result of scenario 1, (h) Planning path result of scenario 2, (i) Planning path result of scenario 3.

scenarios, namely: Case1: ($w = 3$, $\mu = 0.1$); Case2: ($w = 3$, $\mu = 0.2$); Case3: ($w = 5$, $\mu = 0.1$); Case4: ($w = 5$, $\mu = 0.2$).

Evaluation index: According to the coverage path, the robot will travel all areas of the environment. In this process, we considered two indexes: Coverage ratio and Time consumption. In coverage ratio, using S_1 to represent the actual coverage area and S to represents the total area of the environment, the coverage ratio can be expressed as formula (1). Time consumption refers to the time the robot runs according to the planned path, but the coverage rate may not reach 100%.

$$C = \frac{S_1}{S} \times 100\% \tag{1}$$

4.2. SLAM and path planning result

Fig. 7 shows the SLAM and path planning result of ROS-based robot for various scenario. Figs. 7(a), 7(b) and 7(c) show the SLAM results after the robot automatically builds a map. The white part is the wall boundary, and the black area represents the robot's need for coverage. The sub-area division results before executing the bidirectional A* algorithm is shown in Figs. 7(d), 7(e), and 7(f), where all rooms are divided in each scene, and 'S' shaped paths are used for complete coverage planning. It's shown that there are multiple continuous paths that cover different rooms and are not connected to each other. Figs. 7(g), 6(h), and 6(i) show the planned route, which is indicated by the green line. In different scenarios, the planned path can cover the entire area. These paths consist of waypoints that cover the entire environment. The robot can complete the CCPP task by following these locations in order.

4.3. Evaluation result

After getting the planned path, the robot will move towards the target point to cover the environment. To better compare the proposed CCPP scheme, we track the actual movement trajectory of the robot. Figs. 8, 9, and 10 show the coverage trajectories and coverage areas in three different scenarios, respectively. Furthermore, we compare the coverage ratio and time consumption of our proposed CCPP scheme with the random walk method [9], the "S" shape method [10], and the spiral walk method [11]. It is important to note that the compared path planning algorithms only focus on generating complete coverage paths and do not involve ROS-based robot motion control. In contrast, our proposed CCPP scheme includes path planning method and dynamic tracking method specifically designed for control robots. So, we compare the different methods in Tables 1 and 2 by the following way: When the distance threshold is 0, it means that we do not use the dynamic tracking algorithm and use the traditional navigation method; that is, the robot moves to the target point to stop and then moves to the next target point. It is intermittent movement. When the distance threshold is other values, other algorithms also use the dynamic tracking method to drive the robot. The movement of the robot is continuous.

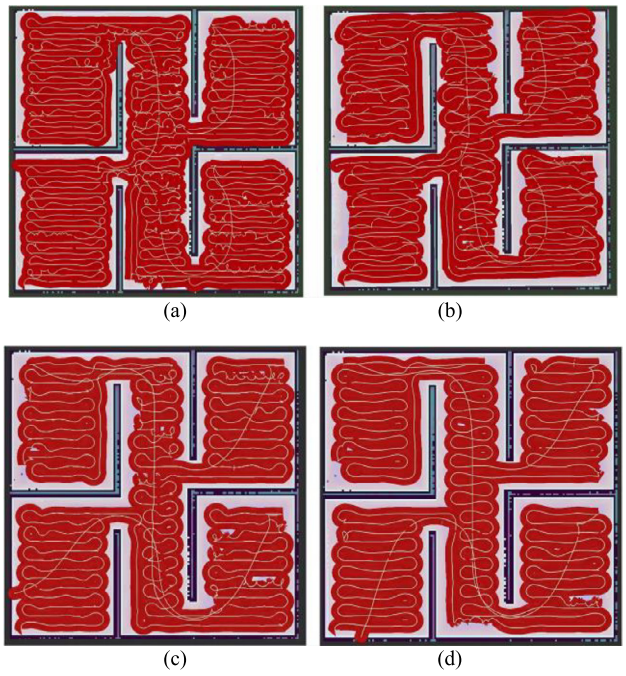

Fig. 8. Coverage results for scenario 1 (red area is the coverage area, and white line is the actual trajectory of the robot). (a) Case 1, (b) Case 2, (c) Case 3, (d) Case 4. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

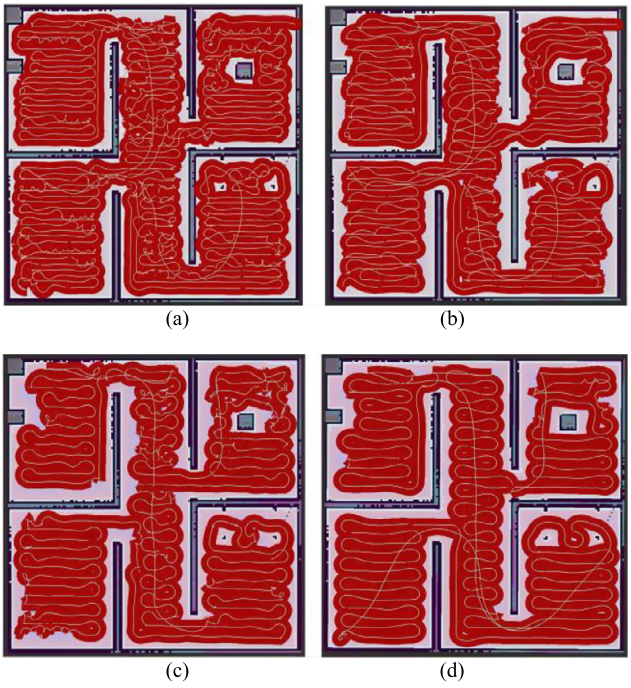

Fig. 9. Coverage results for scenario 2 (red area is the coverage area, and white line is the actual trajectory of the robot). (a) Case 1, (b) Case 2, (c) Case 3, (d) Case 4. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

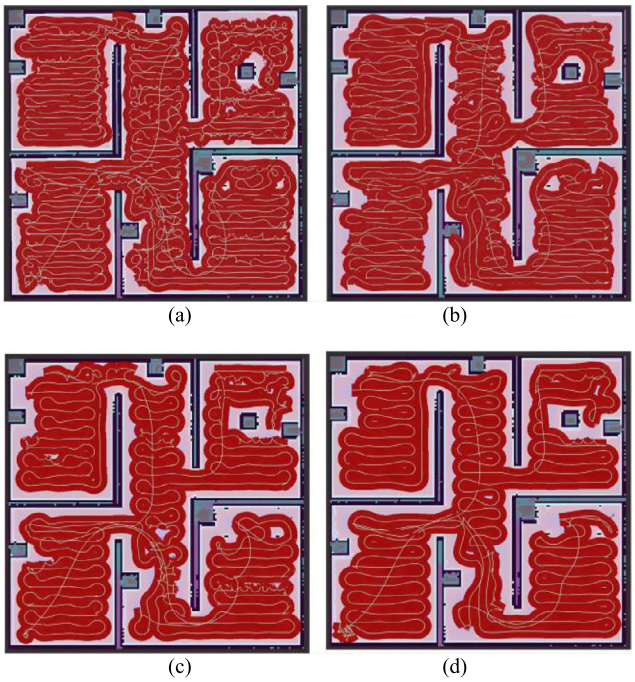

Fig. 10. Coverage results for scenario 3 (red area is the coverage area, and white line is the actual trajectory of the robot). (a) Case 1, (b) Case 2, (c) Case 3, (d) Case 4. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Comparison of the different methods in terms of coverage ratio in three scenarios.

Scenario	Cases		Coverage ratio (%)			
	w	μ(m)	Random walk method [9]	"S" shape method [10]	Spiral walk method [11]	Ours
1	3	0	74.30	96.74	97.18	97.1
	3	0.1	75.00	97.18	97.21	98.01
	3	0.2	70.16	88.23	89.70	91.37
	5	0	70.59	88.84	87.47	90.11
	5	0.1	71.28	89.06	89.18	91.28
	5	0.2	68.43	87.58	87.63	89.63
2	3	0	71.89	91.76	92.48	95.10
	3	0.1	72.73	92.49	93.51	95.51
	3	0.2	69.84	88.22	88.72	90.52
	5	0	65.78	83.96	85.09	85.48
	5	0.1	66.48	85.08	85.18	87.18
	5	0.2	65.14	86.55	86.52	88.59
3	3	0	65.14	93.36	93.01	95.78
	3	0.1	66.26	94.10	94.04	96.04
	3	0.2	64.63	87.28	87.18	89.73
	5	0	61.56	83.69	84.12	87.88
	5	0.1	62.35	84.59	84.53	88.86
	5	0.2	60.46	84.37	83.23	87.78

μ: following thresholds. w: coverage widths.

Table 2
Comparison of the different methods in terms of time consumption in three scenarios.

Scenario	Cases		Time consumption (min)			
	w	μ(m)	Random walk method [9]	"S" shape method [10]	Spiral walk method [11]	Ours
1	3	0	80.10	53.20	55.40	51.10
	3	0.1	77.70	48.80	49.90	48.30
	3	0.2	75.20	39.40	39.60	30.20
	5	0	68.30	38.90	50.40	30.10
	5	0.1	63.20	34.10	34.8	25.30
	5	0.2	57.10	27.10	28.0	17.40
2	3	0	93.70	60.10	59.90	57.40
	3	0.1	89.50	52.50	53.2	48.20
	3	0.2	57.50	35.90	37.9	28.50
	5	0	69.20	39.70	53.20	31.70
	5	0.1	55.10	32.10	33.00	23.50
	5	0.2	43.20	27.40	28.90	17.30
3	3	0	98.20	63.40	65.30	60.30
	3	0.1	85.40	59.80	58.20	49.20
	3	0.2	56.30	38.90	37.30	28.10
	5	0	99.80	42.10	56.80	33.50
	5	0.1	42.40	36.30	35.80	27.10
	5	0.2	40.20	30.40	29.30	18.20

μ: following thresholds. w: coverage widths.

Without dynamic tracking ($\mu = 0$), robots only using path planning and traditional navigational movement patterns will waste more time in the stop-and-start phase. The proposed algorithm achieves a robot running time of approximately 51 min when $w = 3$ and $\mu = 0$. However, when dynamic tracking is enabled ($\mu = 0.1$), the time consumption is reduced by 3 min, and the coverage rate is increased by 1%. These results confirm that our CCPP scheme can achieve higher coverage efficiency while guaranteeing less time consumption (see Tables 1 and 2).

5. Conclusion

In this paper, a new CCPP scheme is proposed, which is combined with the sub-area division algorithm and dynamic tracking for ROS-based robots. By dividing the complex environment into sub-areas using the sub-area division algorithm, the proposed scheme enables robots to plan paths for traversing all points in areas using the 'S' shape method and bidirectional A* algorithm. The dynamic tracking method ensures that the robot moves continuously without stopping and follows the planned route. Simulation results demonstrate the high coverage and effectiveness of the proposed scheme. Future research will explore a CCPP based on deep learning or reinforcement learning.

CRediT authorship contribution statement

Shengmin Zhao: Methodology, Software, Data curation, Visualization, Investigation, Writing – original draft, Writing – review & editing. **Seung-Hoon Hwang:** Conceptualization, Methodology, Investigation, Writing – review & editing, Supervision, Validation.

Declaration of competing interest

The authors declare that there is no conflict of interest in this paper.

Acknowledgments

The following are the results of a study on the "Leaders in Industry-university Cooperation3.0" Project supported by the Ministry of Education and National Research Foundation of Korea.

References

[1] M.A.V.J. Muthugala, M. Vega-Heredia, R.E. Mohan, et al., Design and control of a wall cleaning robot with adhesion-awareness, Symmetry 12 (1) (2020) 122.
[2] C. Hofner, G. Schmidt, Path planning and guidance techniques for an autonomous mobile cleaning robot, Robot. Auton. Syst. 14 (2–3) (1995) 199–212.
[3] Y. Irawan, M. Muhardi, R. Ordila, et al., Automatic floor cleaning robot using Arduino and ultrasonic sensor, J. Robot. Control (JRC) 2 (4) (2021) 240–243.
[4] M.H. Lee, J. Moon, Deep reinforcement learning-based model-free path planning and collision avoidance for UAVs: A soft actor–critic with hindsight experience replay approach, ICT Express (2022).
[5] K. Karur, N. Sharma, C. Dharmatti, et al., A survey of path planning algorithms for mobile robots, Vehicles 3 (3) (2021) 448–468.
[6] L. Gao, W. Lv, X. Yan, et al., Complete coverage path planning algorithm based on energy compensation and obstacle vectorization, Expert Syst. Appl. 203 (2022) 117495.
[7] E. Galceran, M. Carreras, A survey on coverage path planning for robotics, Robot. Auton. Syst. 61 (12) (2013) 1258–1276.
[8] B.K. Patle, A. Pandey, D.R.K. Parhi, et al., A review: On path planning strategies for navigation of mobile robot, Def. Technol. 15 (4) (2019) 582–606.
[9] F. Duchoň, A. Babinec, M. Kajan, et al., Path planning with modified a star algorithm for a mobile robot, Procedia Eng. 96 (2014) 59–69.
[10] Y. Liu, X. Lin, S. Zhu, Combined coverage path planning for autonomous cleaning robots in unstructured environments, in: 2008 7th World Congress on Intelligent Control and Automation, IEEE, 2008, pp. 8271–8276.
[11] C. Stachniss, Robotic Mapping and Exploration, Springer, 2009.
[12] J. Chen, C. Du, Y. Zhang, et al., A clustering-based coverage path planning method for autonomous heterogeneous UAVs, IEEE Trans. Intell. Transp. Syst. 23 (12) (2021) 25546–25556.
[13] J. Chen, Y. Zhang, L. Wu, et al., An adaptive clustering-based algorithm for automatic path planning of heterogeneous UAVs, IEEE Trans. Intell. Transp. Syst. 23 (9) (2021) 16842–16853.
[14] J. Chen, F. Ling, Y. Zhang, et al., Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system, Swarm Evol. Comput. 69 (2022) 101005.
[15] Explore_lite - ROS wiki, 2023, Available online: http://wiki.ros.org/explore_lite/. (Accessed 28 January 2023).