Proceedings of the 39th Chinese Control Conference
July 27-29, 2020, Shenyang, China

A Frontier-Based Coverage Path Planning Algorithm for Robot Exploration in Unknown Environment

Songqun Gao[1], Yulong Ding[2], Ben M. Chen[3]
1. University of Science and Technology of China, Anhui 230000, P. R. China
E-mail: wilson@mail.ustc.edu.cn
2. Beijing Institute of Technology, Beijing 100081, P. R. China
E-mail: dingyvlong@gmail.com
3. The Chinese University of Hong Kong, Hong Kong SAR 999077, P. R. China
E-mail: bmchen@mae.cuhk.edu.hk

Abstract: This paper presents a path planning algorithm for robot exploration in unknown large-scale environments. The proposed algorithm employs a finite state machine to iteratively derive the action that the robot takes to explore its environment when no prior map is available. The finite state machine contains two states, that is zigzag state and travel state. In the zigzag state, the robot explores the current cell using a zigzag pattern and adds frontier nodes. In the travel state, the robot moves to another cell by going to the nearest frontier node. Compared with previous studies, the proposed algorithm can realize complete exploration while maintaining efficiency. Simulation results demonstrate that the efficiency and robustness of the proposed algorithm for exploring the unknown large-scale environments as compared with two other state-of-art algorithms.

Key Words: Robotics, motion planning, exploration, coverage path planning

1 Introduction

Robot plays a more important role in helping humans to accomplish a variety of tasks with its autonomy increasing. In practical applications like mapping abandoned mines [1], search and rescue [2], lunar exploration [3], and underwater exploration [4], robots are required to explore the given large-scale unknown environment and gain as much information as possible autonomously.

In the problem of exploring an unknown environment with a robot, the robot starts with an empty map and seeks to find a path to explore and map the full environment with its sensors. There are serveral methods to solve the exploration problem, like graph partitioning [5, 6], multi-robot exploration [7, 8], frontier-based methods [9, 10] and stochastic methods [11–13] and so on. the concept of the frontier was introduced in [9, 10], which was defined as the region between explored free space and unknown space in the local map. Besides, next best views (NBV) [14] and rapidly-exploring random tree (RRT) [15] is adopted to find an effective exploration path in [11–13]. They used the RRT to pick up a target having the most unknown information. However, current studies could not guarantee the completeness of the exploration, or could not ensure efficiency while achieving the complete exploration.

The finite state machine (FSM) was also introduced to resolve the robot exploration problem [16, 17]. Especially, the exploration coverage path planning (ECPP) algorithm [17] used an FSM with two states: normal state and travel state. The main idea of this algorithm is to decompose the area into individual cells based on landmarks and use zigzag pattern paths to ensure the complete coverage of every cell. In the normal state, morse decomposition introduced by [18] was used to generate a zigzag pattern path in the cell. While in the travel state, the robot moves between cells. The ECPP algorithm could assure the efficient exploration path in every cell, however, it cannot simultaneously ensure the short distance between cells.

In this paper, we propose a novel frontier-based coverage path planning (FBCPP) algorithm for robot exploration. In the FBCPP algorithm, the search area is decomposed into cells by detecting frontier nodes and each cell is explored using a zigzag pattern. The proposed algorithm employs a finite state machine to iteratively derive the action that the robot takes to explore its environment. The finite state machine contains two states, that is zigzag state and travel state, respectively. In the zigzag state, the robot explores the current cell with the zigzag pattern and when the robot finishes exploring the current cell, it turns to the travel state. In the travel state, the robot navigates to the nearest frontier node where the robot can gain new information and enters the new cell, then the robot returns to the zigzag state. The whole process is repeated until all the information is acquired.

The main contributions of the paper are the following:

- A new finite state machine with two states is proposed to iteratively derive the robot to accomplish the complete exploration. The zigzag pattern path in the zigzag state can assure efficient exploration in every cell.
- Frontier node is introduced to increase the efficiency of the path generated. In the travel state, the robot moves to the nearest frontier node, which can shorten the path length generated in the travel state.
- Simulations demonstrate that our algorithm is efficient and robust and outperforms the RH-NBV and ECPP algorithms in aspects of path cost of complete exploration.

The remainder of the paper is organized as follows. Section 2 discusses the preliminaries of the problem and gives out problem formulation. Section 3 presents the frontier-based coverage path planning algorithm, and Section 4 validates our algorithm. Finally, conclusions are presented in Section 5.

Corresponding author: Yulong Ding

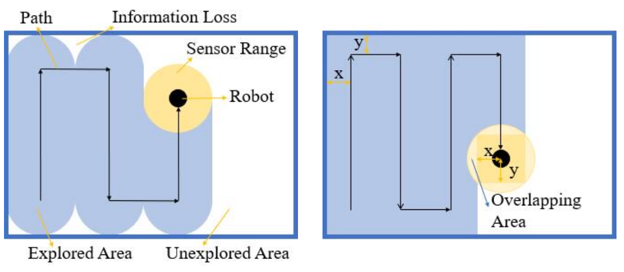


Fig. 1: Morse decomposition (a) The information loss by using the sensory circle. (b) No information loss by only using the map information in the inscribed rectangle of the sensory circle. The *x* indicates how far the robot should keep from the explored path or the obstacle alongside when going straight, and *y* indicates that when the robot comes to the front of the obstacles, how far from the wall when it should turn into another direction.

2 Problem Description

2.1 Morse Decomposition

Morse decomposition algorithm, proposed by [18], is an algorithm that decomposes a space into many non-overlapping regions, denoted by cells. In this algorithm, morse decomposition is adopted to generate the zigzag pattern path in every cell, which ensures the coverage of those cells, just like the "mowing the lawn" pattern as shown in Fig. 1. However, according to [19], in morse decomposition, a zigzag path fails to cover the edge of the cell, which will cause the loss of map information, as shown in Fig. 1(a), the white area at the boundary has not been explored. To avoid it, previous methods used repeated rectangular motion cycles, but it will greatly expand the path cost.

To ensure the complete exploration of the cell, in our method, the robot only uses the map information in the inscribed rectangle of the sensory circle for path planning, as shown in Fig. 1(b). There is no information loss in the area. The *x* indicates how far the robot should keep from the explored path or the obstacle alongside when going straight, and *y* indicates that when the robot comes to the front of the obstacles, how far from the wall when it should turn into another direction. We adopt *x* equals *y* because while exploring a new cell, the robot doesn't know the exact size of the cell. Therefore, to make sure the worst case cannot occur, we maintain the symmetry between x and y.

2.2 Problem Formulation

2.2.1 Environmental Modeling

The following environmental model is adopted considered in this paper.

Workspace(WS): The workspace of the robot is built as a Cartesian coordinate system in a two-dimensional plane. In the article a number of obstacles $(Obs_j, j = 1,2,3,\dots)$ are located in the WS. The robot is treated as a particle in the two-dimensional plane WS, and the position of the robot at time t is defined as $P(t) = (P_x(t), P_y(t))$. In the WS, the node can be expressed as $N = (N_x, N_y)$, and each node have the value either empty or occupied, that is, $WS(N) \in \{empty, occupied\}$. The distance between any two points in the Cartesian coordinate system is determined by their Euclidean distance:

$$d(N_1,N_2) = \sqrt{(N_{1x}-N_{1x})^2 + (N_{1y}-N_{2y})^2}. \quad (1)$$

The robot can only detect the information in a circle domain with the radius of R, then the sensor range can be defined by (2):

$$SR(P,t) = \{N | d(N,P(t)) \leq R\}. \quad (2)$$

It is assumed that the map information is only locally known, not globally known, which means the information can only be gained by the onboard sensor. To start up, t_0 is set to be the initial time $t_0 = 0$, and local map M is initialized as an empty set, i.e. $M(t_0) = \emptyset$, which means all the node in the WS is unknown at first. Then robot gathers the map information gradually and adds empty or occupied nodes obtained by the sensor into M:

$$\begin{aligned} M(t_i) = M(t_{i-1}) \cup \{(N, state) | N \in SR(P,t_i), \\ state \in \{empty, occupied, unknown\}\}. \end{aligned} \quad (3)$$

2.2.2 The Model of Robots

The model of the robot in this article is based on the following premises and assumptions.

- The robots is initialized as $P(t_0) = (P_{x0}, P_{y0})$.
- The robots do not have prior knowledge of the environment globally and they obtain environmental information locally by their sensor.
- Each robot can detect the environmental information in a circle domain with the position of the robot as the center and R as the radius, and they can thus map out its path in this local environment. We call this circular area as the current rolling window of the robots.
- The onboard sensor is modeled as a small disc with a radius of 0.5 m.

2.2.3 Exploration Problem

The problem considered in this work is to explore the whole given area WS and gather complete map information in the WS and add it into M, meanwhile minimizing the path length. According to (1)-(3), we can express the whole optimization model as following:

$$\begin{aligned} & S = \arg\min J = \sum_{i=1}^{N} D(P(t_i), P(t_{i-1}), M, t_i) \\ s.t. \quad & P(t_i) = (P_{xi}, P_{yi}), i = 0,1,\dots,N \\ & M(t_i) = M(t_{i-1}) \cup \{(N, state) | N \in SR(P,t_i), \\ & state \in \{empty, occupied, unknown\}\}, i = 1,2,\dots,N \end{aligned}$$

where N is the number of time series. $P(t_i)$ and t_i are the visit position and the visit time for the robot. $S = [P(t_i), P(t_2), \dots P(t_N)]$ is the sequence of the robot's position. $D(P(t_i), P(t_{i-1}), M, t_i)$ is the path length from $P(t_{i-1})$ to $P(t_i)$ generated by A* algorithm at local Map M. J then represents the total path length.

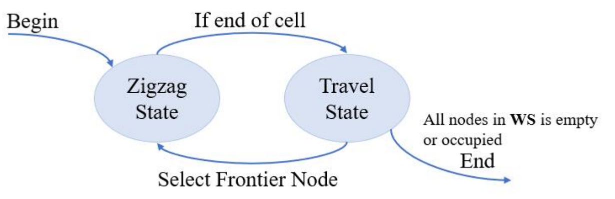


Fig. 2: The FBCPP algorithm is realized as a finite state machine with two states.

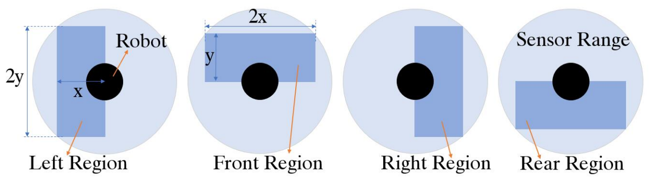


Fig. 3: Adjacent regions in four directions. The black circle is the robot, the sky-blue circle is the sensor range, and the dark blue rectangle denotes adjacent region. Adjacent region is used to determine if the robot needs to explore in the certain direction.

3 Frontier-Based Coverage Path Planning Algorithm

This section will introduce the FBCPP algorithm. The algorithm decomposes the area into cells by detecting frontier nodes and each cell is explored using a zigzag pattern. It is realized as a finite state machine having two states: zigzag state and travel state shown in Fig. 2. In the zigzag state, the robot explores the current cell with the zigzag pattern and add up the nodes in the node assembly S. When the robot has done exploring the current cell, it turns to the travel state. While in the travel state, the robot first gets rid of all the visited nodes in the node assembly S, then it goes to the nearest frontier node selected from the S and enters the new cell. Finally, the robot returns to the zigzag state. The algorithm ends when the states of all nodes in WS are empty or occupied.

3.1 Zigzag state

To represent four directions, four Adjacent Regions are selected (left region, front region, right region and rear region), as shown in Fig. 3. Four regions are one-to-one corresponding to four directions (left, front, right, rear). The zigzag pattern is realized by moving in these four directions. Searching in these four regions has low computing power to ensure real-time quality.

The method to determine if the robot needs to explore in a certain direction is: Using the A* algorithm ([20]) to decide if the robot can reach the farthest node of the corresponding Adjacent Area. If the robot can reach it, which means that there is unexplored information remained in that direction, the farthest node will be added in the node assembly S and visited later. Fig. 4(a) shows the robot can reach the farthest node in the left region, so, the robot adds the node into the node assembly S. But if the robot cannot reach it, it means the robot has arrived at the edge of the cell, and there is no need to add nodes to the node assembly S in this direction,

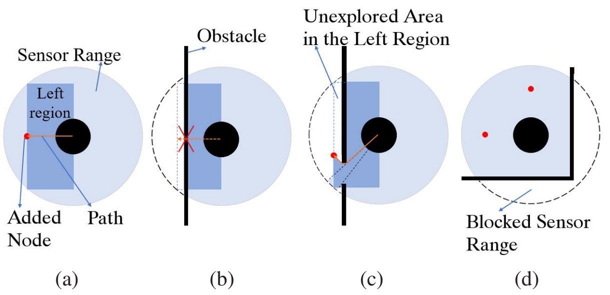


Fig. 4: Several situations of determining if the certain direction needs to be explored by looking into the corresponding adjacent region. The red dot is the node that added in the node assembly S, the yellow array is the generated path

as shown in Fig. 4(b). Fig. 4(c) indicates that in the process of A* path searching, the unexplored area is not considered. And Fig. 4(d) shows the diagram of adding nodes, paths can be made in both front region and left region, while can neither be made in the rear region nor the right region, so the robot only adds nodes in the front region and left region.

The overall algorithm is shown in **Algorithm 1**. In the i^{th} iteration, in the zigzag state, the robot first determines if every direction needs to continue exploring and extracts fewer than four nodes in these directions. According to the zigzag pattern, the robot picks one node as the target. The robot then adds all the nodes except one that been set as the target into the node assembly S. Finally, the robot starts from the $P(t_i)$ position, navigate to the target $(P(t_{i+1}))$ generated before. The robot then begins the $(i+1)^{th}$ iteration and switches to travel state until the cell reaches the end.

Algorithm 1 Zigzag state algorithm

```
1:  Initialize: randomly pick an accessible direction.
2:  while Not End of the Cell do
3:      for $m = 1 : 4$ do
4:          Check in four directions (left, front, right, rear) of the
            robot if the robot needs to explore the certain direction.
5:          if the direction needs to be explored then
6:              Add the farrest point into the node assembly $S$.
7:          end if
8:      end for
9:      if the robot's current direction is accessable then
10:         Pick the corresponding point as the target of next move.
11:     else
12:         Pick another point as the target among other available
            points according to the zigzag pattern.
13:     end if
14:     Delete the selected node from $S$.
15:     Move to the target.
16: end while
17: Change to the travel state.
```

3.2 Travel state

In the FBCPP algorithm, we introduce the concept of the frontier node. When the robot first enters the travel state, the robot makes judgments on all the nodes in the node assembly S, If the robot goes to the node and can gain new map information, then this node is a frontier node, otherwise, it is not. Then the robot deletes this node from the node assembly S. Intuitively, the frontier node is the node that contains un-

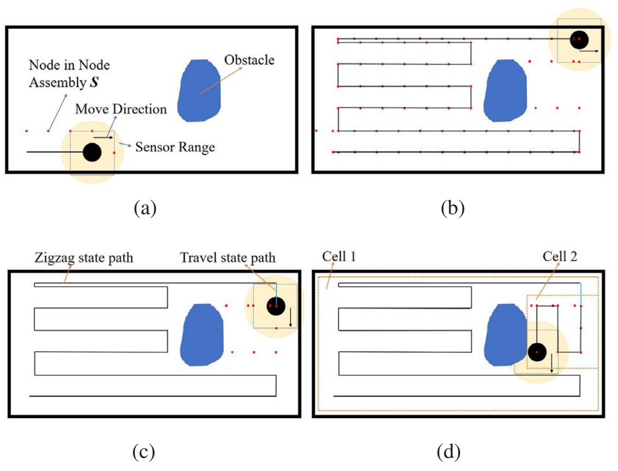


Fig. 5: Diagram of Frontier-Based Coverage Path Planning Algorithm.

known map information, and the robot is supposed to move to there in the travel state so that the robot can gain new information.

The general method in the travel state is shown in **Algorithm 2**. In the i^{th} iteration, when the robot first enters the travel state, it first cleans the node assembly S and leaves only frontier nodes, then it picks the nearest frontier node in the updated node assembly S as the target and moves to it. Because the previous cell has been completely explored, where no new map information will be acquired. However, there is new information at the frontier node, which means the frontier node is in a new cell. So, going to the frontier node represents entering the new cell. Then, the robot returns to the zigzag state and begins the $(i+1)^{th}$ iteration. The robot will explore a new cell then. It is worthwhile mentioning that as a result of choosing the nearest node, it ensures the low-cost path in the travel state.

The whole process is shown in Fig. 5, the robot starts in the zigzag state and explores cell 1. When it arrives at the end of cell 1, it cleans the node assembly S, picks the nearest frontier node and moves to there. Finally, the robot explores cell 2 and the whole exploration of the area is finished.

Algorithm 2 Travel state algorithm

```
1: for All Nodes in the Node Assemble $S$ do
2:     if the node has been visited then
3:         Delete the Node.
4:     end if
5: end for
6: Select the nearest node as the target.
7: Move to the target.
8: Change to the zigzag state.
```

4 Experiment

To evaluate the performance of the proposed FBCPP algorithm, two different simulation studies have been conducted using MATLAB in three scenarios of different complexity, as shown in Fig. 6. The ECPP algorithm [17] and RH-NBV algorithm [11] are adopted for comparison. ECPP algorithm decomposes the area into cells based on landmark detection and using a zigzag pattern to explore each cell. RH-NBV algorithm uses RRT to pick up a target having the most unknown information. The first experiment is to compare the FBCPP algorithm to the ECPP algorithm and the RH-NBV algorithm in the aspect of runtime and cost of the generated path. The second simulation is to verify the robustness of the FBCPP algorithm by randomly choosing the start point. In the simulation, the robot was set to explore three 7.5m$\times$4.5m scenarios and the robot can detect the map information in a circle domain with a radius of 0.5m. All the tests were performed under the parameters shown in Table 1.

Table 1: Parameters

Parameters	Value
Map size	7.5 m$\times$4.5 m
Map resolution	0.1 m
Radius of robot sensor	0.5 m
Degress Coefficient	0.5
RRT max edge length	0.5 m
RRT max node number	30
x	0.3 m
y	0.3 m

4.1 Path Cost and Runtime of Different Algorithms

In the first simulation, the runtime of generating the target and the path costs of three different algorithms in different scenarios are tested, as listed in Table 2.

The results show that the FBCPP algorithm is much more efficient than other algorithms compared. Besides, The FBCPP and ECPP algorithm can realize the complete exploration of the given area in limited time while RH-NBV takes a long time to cover. Fig. 7 also shows the exploration progress using different algorithms in different scenarios. Notably, the line parallel to the X-axis is the path with no map information gains. In the FBCPP and the ECPP algorithm, it denotes the path generated by travel state. As can be seen from Fig. 7, the FBCPP algorithm has fewer lines parallel to the X-axis than the ECPP algorithm. It means that by reducing the cost of the travel state path, the FBCPP takes a fewer path to acquire the complete map information than the ECPP algorithm. The generated paths of different algorithms in scenario (a) are shown in Fig. 8.

As shown in Table 2, the FBCPP algorithm outperforms the ECPP algorithm and RH-NBV algorithm in terms of path cost and it is time efficient. The reason why the FBCPP algorithm is better than the other two algorithms compared is that the RH-NBV algorithm adopts RRT to generate many nodes and pick up the node having the best gain, which is time-consuming. It will also cause the problem of failing to explore the area with a little map information because it is prone to explore the place having the most map information. As a result, the RH-NBV algorithm can obtain map information faster than the other two algorithms. As shown in Fig. 7, the growth rate of map information in RH-NBV algorithm is larger than other algorithm at the beginning. However, it takes a long time to obtain a near coverage exploration (95%). This is because the robot will randomly wander around when the surrounding is completely explored, which is easily stuck in this nearby environment.

As for the ECPP algorithm, it needs to detect landmarks to

Table 2: Comparision between different algorithms

Algorithm	Scenario(a) (m)	runtime in (a) (ms)	Scenario(b) (m)	runtime in (b) (ms)	Scenario(c) (m)	runtime in (c) (ms)
FBCPP	**46.0**	**2.2**	**62.9**	6.0	**82.8**	10.6
ECPP	57.7	3.2	82.3	**5.6**	101.0	**7.3**
RH-NBV (95% coverage)	290.2	23.6	169.8	25.2	752.1	28.6

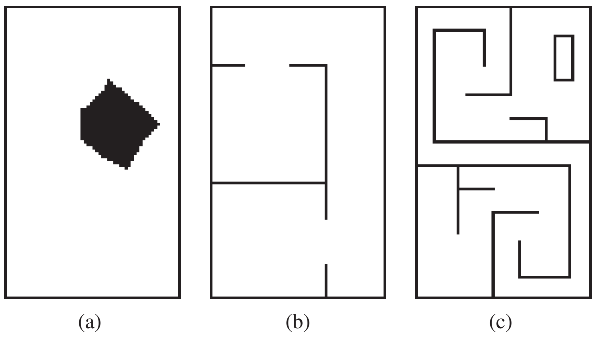


Fig. 6: Three experimental scenarios

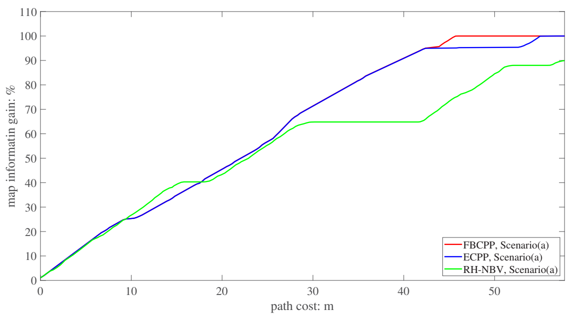
add nodes, which is easily influenced by the scenario, while the FBCPP algorithm does not depend on landmarks to add nodes, and it can add nodes when generating the next move. Moreover, the target generated by the ECPP algorithm is not always the nearest node where the robot can gain new map information.

4.2 Different Path Cost Under Random Starting Point

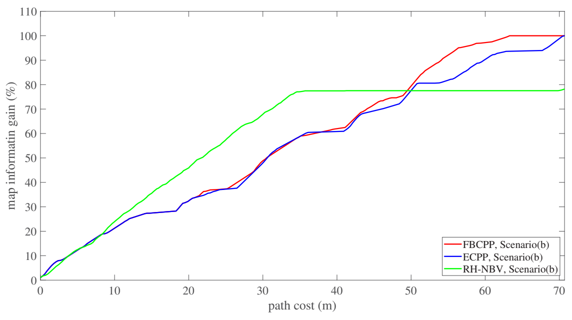
In Scenario (b), random starting points are generated to test the robustness of the proposed FBCPP algorithm. Table 3 shows the path length generated by FBCPP, ECPP, and RH-NBV with seven start points. As shown in Table 3, the average path length of FBCPP is 64.6m and its variance is 1.088, while the ECPP and RH-NBV's variance is much larger. Therefore, our proposed FBCPP algorithm is more robust than other two algorithms to withstand uncertainties. Since the ECPP algorithm adds nodes by obstacle detection, which is easily influenced by the scenario, shown in Fig. 8(b). However, FBCPP relies on the frontier nodes that are generated every single move in the adjacent regions. It is more robust against complex scenarios, shown in Fig. 8(a). Besides, the RH-NBV algorithm picks nodes of the best gain in the generated RRT and the method cannot guarantee robustness.

Table 3: Path cost of different algorithms with random start points in Scenario (b)

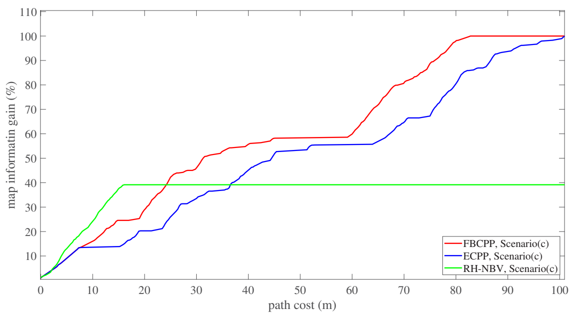
Start Point (m)	FBCPP (m)	ECPP (m)	RH-NBV (m) (95% coverage)
(2.8, 0.2)	**64.4**	66.3	174.0
(3.6, 3.3)	**63.8**	68.4	228.6
(4.2, 4.5)	**64.9**	68.7	278.2
(0.6, 6.9)	**62.9**	82.3	157.6
(2.1, 3.4)	**64.8**	64.8	400.3
(4.2, 6.4)	**66.0**	74.9	448.6
(1.3, 4.1)	**63.4**	63.9	269.4
average	**64.3**	699.0	2795.3
variance	**1.088**	42.903	11978.640



(a) Results in Scenario(a)



(b) Results in Scenario(b)



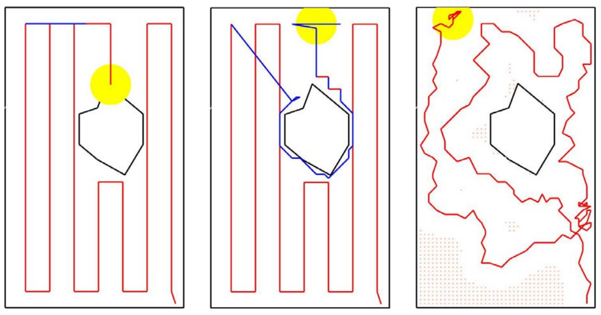
(c) Results in Scenario(c)

Fig. 7: Exploration progress using different algorithms in different scenarios.

5 Conclusion

In this work, the Frontier-Based Coverage Path Planning algorithm was proposed to guide the robot to explore unknown areas. The FBCPP algorithm is realized by a finite state machine with two states: zigzag state and travel state. In the zigzag state, the robot explores the current cell in a zigzag pattern and add nodes to the node assembly *S*. When the cell reaches the end, the robot turns to the travel state. In the travel state, the robot first cleans all the visited nodes to the node assembly *S*, then goes to the nearest frontier node and enters the new cell, Finally, the robot returns to the zigzag state. The whole process is repeated until all the information is acquired.

Our algorithm improves the existing exploration algo-



(a) FBCPP (b) ECPP (c) RH-NBV

Fig. 8: The generated paths for different algorithms in scenario (a). The yellow circle denotes robot's sensor range and orange dots means unexplored area (whose states are unknown). In (a) and (b), The red and blue lines represent the paths generated in the zigzag state and the travel state, respectively.

rithms in several aspects. First, the zigzag pattern exploration in the cell is efficient and complete. Second, by choosing the nearest frontier node in the travel state, the robot goes to the nearest place that can gain new map information and reduce the path cost. Two simulations are operated to validate our FBCPP algorithm. The results demonstrate the efficiency and robustness of the algorithm. A future direction of this work is to generalize our algorithm to take the multi-robot exploration into consideration.

References

[1] S. Thrun, S. Thayer, W. Whittaker, C. Baker, W. Burgard, D. Ferguson, D. Hahnel, M. Montemerlo, A. Morris, Z. Omohundro, C. Reverte, and W. Whittaker, Autonomous exploration and mapping of abandoned mines: Software architecture of an autonomous robotic system, *IEEE Robotics and Automation Magazine*, 11(4): 79–91, 2004.

[2] T. Petiek, V. alansky, K. Zimmermann, and T. Svoboda, Simultaneous exploration and segmentation for search and rescue, *Journal of Field Robotics*, 36(4): 696–709, 2019.

[3] G. Hegde, C. J. Robinson, C. Ye, A. Stroupe, and E. Tunstel, Computer vision based wheel sinkage detection for robotic lunar exploration tasks, in *Proceedings of 2010 IEEE International Conference on Mechatronics and Automation* , 2010: 1777–1782.

[4] S. Lai, K. Wang, K. Li, and B. M. Chen, Path planning of rotorcrafts in unknown environment, in *Proceedings of 2016 35th Chinese Control Conference*, 2016: 10900–10905.

[5] G. Lozenguez, L. Adouane, A. Beynier, A.-I. Mouaddib, and P. Martinet, Map partitioning to approximate an exploration strategy in mobile robotics, *Multiagent and Grid Systems*, 8(3): 275–288, 2012.

[6] A. Gautam, V. S. Shekhawat, and S. Mohan, A graph partitioning approach for fast exploration with multi-robot coordination, in *Proceedings of 2019 IEEE International Conference on Systems, Man and Cybernetics*, 2019: 459–465.

[7] M. Corah and N. Michael, Distributed matroid-constrained submodular maximization for multi-robot exploration: theory and practice, *Autonomous Robots*, 43(2), 485–501, 2019.

[8] B. Xin, G.-Q. Gao, Y.-L. Ding, Y.-G. Zhu, and H. Fang, Distributed multi-robot motion planning for cooperative multi-area coverage, in *Proceedings of 2017 13th IEEE International Conference on Control & Automation*, 2017: 361–366.

[9] B. Yamauchi, Frontier-based approach for autonomous exploration, in *Proceedings of 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997: 146–151.

[10] T. Cieslewski, E. Kaufmann, and D. Scaramuzza, Rapid exploration with multi-rotors: A frontier selection method for high speed flight, in *Proceedings of 2017 IEEE International Conference on Intelligent Robots and Systems*, 2017: 2135–2142.

[11] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, Receding horizon next-best-view planner for 3d exploration, in *Proceedings of 2016 IEEE International Conference on Robotics and Automation*, 2016: 1462–1468.

[12] C. Papachristos, S. Khattak, and K. Alexis, Uncertainty-aware receding horizon exploration and mapping using aerial robots, in *Proceedings of 2017 IEEE International Conference on Robotics and Automation*, 2017: 4568–4575.

[13] M. Selin, M. Tiger, D. Duberg, F. Heintz, and P. Jensfelt, Efficient autonomous exploration planning of large-scale 3-d environments, *IEEE Robotics and Automation Letters*, 4(2): 1699–1706, 2019.

[14] C. Connolly, The determination of next best views, in *Proceedings of 1985 IEEE International Conference on Robotics and Automation*, 1985: 432–435.

[15] S. M. LaValle, Rapidly-exploring random trees : a new tool for path planning, tech. rep., 1998.

[16] S. C. Wong and B. A. MacDonald, A topological coverage algorithm for mobile robots, in *Proceedings of 2003 IEEE International Conference on Intelligent Robots and Systems*, 2003: 1685–1690.

[17] D. Jia, M. Wermelinger, R. Diethelm, P. Krusi, and M. Hutter, Coverage path planning for legged robots in unknown environments, in *Proceedings of 2016 International Symposium on Safety, Security and Rescue Robotics*, 2016: 68–73.

[18] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, Morse decompositions for coverage tasks, *International Journal of Robotics Research*, 21(4): 331–344, 2002.

[19] E. Galceran and M. Carreras, A survey on coverage path planning for robotics, *Robotics and Autonomous Systems*, 61(12): 1258–1276, 2013.

[20] P. E. Hart, N. J. Nilsson, and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107, 1968.