International Core Journal of Engineering Volume 10 Issue 4, 2024
ISSN: 2414-1895 DOI: 10.6919/ICJE.202404_10(4).0007

Research on Path Planning Algorithm for Water Surface Garbage Cleaning Robots

Wencheng Wang[1,2], Quandi Wu[3], Guoqing Cai[1,2], Ruilin Hao[1,2]

[1] Department of Mechanical Engineering, Hebei University of Water Resources and Electric Engineering, Cangzhou, Hebei 061000, China

[2] Hebei Industrial Manipulator Control and Reliability Technology Innovation Center, Cangzhou, Hebei 061000, China

[3] College of Mechanical and Electrical Engineering, Hebei Normal University of Science & Technology, Qinhuangdao, Hebei 066000, China

Abstract

To achieve full coverage path planning for lake surfaces and reduce the repetition rate of travel paths for water surface garbage cleaning robots, a full coverage path planning algorithm combined with the A* algorithm has been designed, integrating the Zigzag_path function with the A* algorithm, and optimizations were made targeting the issue of path repetition rates. And the the results of the program before and after optimization were compared. The results of the program indicate that the optimized program is capable of fulfilling the requirements of full coverage for water surface garbage cleaning robots and has significantly reduced the repetition rate of paths.

Keywords

Garbage Cleaning Robot; Path Planning; A* Algorithm; Zigzag_Path Function; Path Repetition Rate.

1. Introduction

In the process of globalization, water pollution is continuously aggravating. The design of robots as an effective tool to solve this problem makes their path planning particularly important. Path planning concerns the efficiency and cost of the cleaning process, directly affecting the effectiveness of the cleanup operation.

Existing full-coverage path planning algorithms mainly include the random method, maximum area first algorithm, ISC (Inner Spiral Cover algorithm), template-based complete coverage method, Boustrophedon reciprocating method, etc[1-5]. The random method allows the robot to randomly choose a direction to move forward, turning around and continuing forward upon encountering an obstacle[6]. This method is simple to implement, but cannot guarantee complete coverage. The maximum area first algorithm divides the area to be covered into many grids. As the robot moves, it constantly calculates the number of uncovered grids in the front, left, and right directions from its current position. If there are uncovered grids ahead, the robot continues forward; otherwise, it turns towards the direction with more uncovered grids. The template-based complete coverage method uses a known environmental map and a series of templates to plan out coverage paths, applying different templates to different terrains. This method can achieve complete coverage but requires prior knowledge of the environmental map. The Boustrophedon reciprocating method consists of a simple back-and-forth strategy and an improved reciprocating strategy. The simple strategy leaves uncovered areas caused by obstacles, and the more obstacles, the more uncovered areas. The improved strategy performs a second coverage perpendicular to the first direction of travel, thus improving the coverage rate. Therefore, the above methods all have room for improvement in coverage efficiency.

2. Problem Description

When the designed surface-floating garbage robot autonomously travels, it is required to sweep over the entire lake surface. Upon encountering obstacles, it must circumnavigate and return to the original path to continue traveling. When encountering the lake boundary, it should turn around and continue scanning the next path, cycling back and forth covering the entire lake surface area except for the obstacles, as shown in Figure 1.

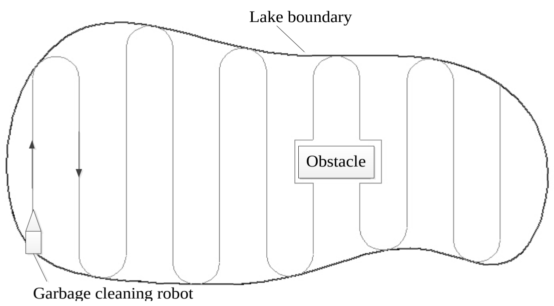

Figure 1. Robot travel trajectory

3. Map Construction

To achieve the above requirements, it is first necessary to construct the lake environment, that is, the passable area and obstacles on the water surface. A grid map is easy to calculate for complete coverage. Thus, this part uses a grid to build a lake map, where white represents a passable area, and black indicates obstacles, as shown in Figure 2.

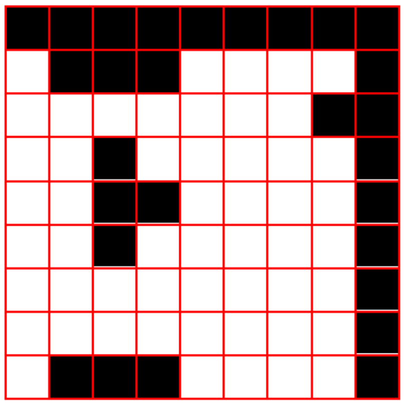

Figure 2. Grid map

For ease of program calculation, the environmental map can be converted into a matrix form, where 0 represents passable, and 1 represents impassable.

4. Algorithm Description

Designed a Zigzag_path method based on the full coverage path planning algorithm for full coverage. When encountering an obstacle, the A* algorithm plans the shortest path from the current point to an uncovered point behind the obstacle in the direction of the current path, then begins covering the new area until the entire coverage is complete.

4.1 Zigzag_path Method Principle

The principle of the Zigzag_path method is a path planning algorithm designed to achieve full coverage in a given area efficiently and systematically. The basic approach is:

(a) Define the work area: First, you need to define the area that should be completely covered by the robot or device, which can be any shape. For simplicity in algorithm implementation, rectangular areas are usually used to reduce complexity.

(b) Set the starting point: Choose a point as the starting point, often one of the corners of the area.

(c) Choose a direction: Determine the initial direction of movement for the robot or device. The Zigzag_path algorithm needs to move back and forth in a straight line to cover the entire area, so decide whether to move back and forth vertically or horizontally.

(d) Move along a straight line: The robot starts from the initial point and moves across to the other side in the set direction.

(e) Turn and move parallel: Once the boundary is reached, the robot needs to turn and move a certain width, usually determined by the working width of the robot. Then it will move in the opposite direction parallelly.

(f) Back-and-forth coverage: The robot repeats the forward movement and turns until it returns to the starting point or covers the entire area.

4.2 A* Algorithm Obstacle Avoidance Principle

The A* algorithm is a classic graph traversal and path search algorithm, combining the features of best-first search and Dijkstra's algorithm. When used to search for a path from the starting point to the endpoint on a grid map, the A* algorithm will consider avoiding obstacles and finding an effective path. Below are the basic principles for obstacle avoidance using the A* algorithm:

(a) Heuristic Function: The A* algorithm requires an estimation function to evaluate the expected cost from the current node to the target node. This estimation needs to be consistently less or equal to the actual minimum cost from the current node to the target node. In an obstacle-avoidance grid, the commonly used heuristic function is the Manhattan distance, which is the number of grid squares from the current point to the target point allowing only horizontal and vertical movements.

(b) Total Cost: To choose a search path, the A* algorithm calculates the total cost f(n) for each node, which is the sum of the actual cost g(n) and the heuristic estimate cost h(n).

(c) Priority Queue: The A* algorithm uses a priority queue to store all nodes awaiting search and sorts the queue by the nodes' total cost f(n). It always first checks the node with the smallest total cost, ensuring that the search direction is towards the target with the smallest costs (path length and obstacle avoidance).

(d) Node Expansion: When a node is taken out of the priority queue, all its feasible neighbor nodes (those not occupied by obstacles and not yet visited) are evaluated. For each neighbor node, the algorithm updates its g(n) and f(n) values. If a better path is found (i.e., lower f(n)), the parent node is updated to the current node.

(e) Obstacle Avoidance: When expanding nodes, the algorithm considers the presence of obstacles. Whenever a node is considered as part of a path, the algorithm checks if it is an obstacle. If so, that node is excluded and not added to the priority queue.

(f) Path Reconstruction: Once the target node is processed, meaning a path has been found, the algorithm reconstructs the entire path by tracing back the parent nodes from the target node to the starting point.

Through these steps, the A* algorithm can find the optimal path from the starting point to the endpoint in an environment with obstacles, while avoiding them.

5. Description of the Designed Full Coverage Algorithm

By combining the Zigzag_path algorithm and the A* algorithm, a full coverage path planning algorithm suitable for the robot's travel routes was proposed. The specific steps of the algorithm are as follows:

STEP1: Execute full coverage according to the Zigzag_path algorithm.

STEP2: If the grid ahead is a boundary, rotate 180° to the right and reverse to travel along a straight line.

STEP3: Turn to STEP4 if the robot encounters an obstacle; otherwise, return to STEP1.

STEP4: Use the A* algorithm to plan the shortest path from the current robot position in the direction of the path to the uncovered point found behind the obstacle, then the robot follows the path to reach the uncovered grid. There will be a certain amount of repeat coverage, but since the A* algorithm provides the shortest path, it ensures a lower repetition rate as much as possible.

STEP5: The robot continues the coverage task from a new starting point, back to STEP1.

STEP6: The algorithm concludes.

Python was used to write the code for the proposed algorithm.

6. The Running Results of The Program

The proposed algorithm was programmed using Python, and a map as shown in Figure 3 was created. The robot's movement coordinates sequence and travel route map were outputted during operation, as shown in Figure 4. Figure 4 indicates that the algorithm has achieved complete coverage of the lake surface and obstacle avoidance for the robot.

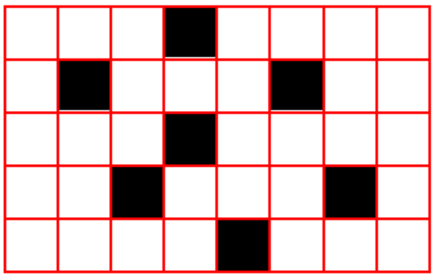
Figure 3. Lake map

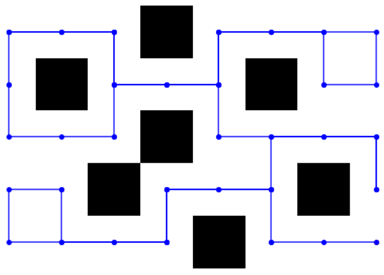
Figure 4. Robot route map

The returned sequence of coordinate points the robot passes through is as follows:

[(0, 0), (0, 1), (0, 2), (0, 2), (1, 2), (1, 3), (1, 4), (0, 4), (0, 4), (0, 5), (0, 6), (0, 7), (1, 7), (1, 6), (1, 6), (0, 6), (0, 5), (0, 4), (1, 4), (1, 4), (1, 3), (1, 2), (1, 2), (0, 2), (0, 1), (0, 0), (1, 0), (1, 0), (2, 0), (2, 1), (2, 2), (2, 2), (1, 2), (1, 3), (1, 4), (2, 4), (2, 4), (2, 5), (2, 6), (2, 7), (3, 7), (3, 7), (2, 7), (2, 6), (2, 5), (3, 5), (3, 5), (3, 4), (3, 3), (3, 3), (4, 3), (4, 2), (4, 1), (3, 1), (3, 1), (3, 0), (4, 0), (4, 1), (4, 2), (4, 3), (4, 3), (3, 3), (3, 4), (3, 5), (4, 5), (4, 5), (4, 6), (4, 7)]

Combining the running coordinates sequence of the robot with the route map, we can obtain the number of times the robot passes through each grid point, as shown in Figure 5. Obstacles are represented by black; passing through once is grey; twice is blue; thrice is red.

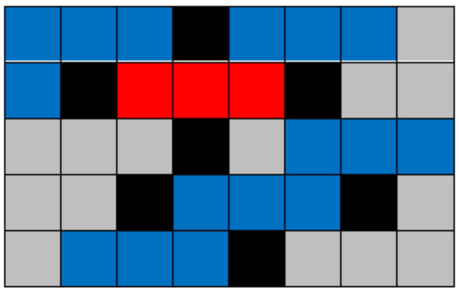

Figure 5. Robot route map before optimization

Through calculation, it can be seen that when the robot is controlled by this program, the repeat rate of path is as high as 66%. In order to achieve full coverage, the high path repeat rate increases the energy waste of the robot, so this program needs to be further improved.

7. Improvement of Algorithm

To reduce the repetition rate of points passed in the travel path, the Zigzag_path function has been optimized. The method involves considering a turn strategy in the current forward direction when encountering an obstacle. This allows for a change of direction along the original path where possible, rather than simply circumventing the obstacle.

The Make_turn function has been added to the existing Zigzag_path function, making the optimized approach attempt to turn before the obstacle first, and when there is no alternative route, the A* algorithm is initiated to find a path.

(a) Before and after each encounter with an obstacle, check if it’s possible to move forward.

(b) If direct forward movement is not possible, attempt to change direction (turn) and choose an available direction.

(c) If turning still doesn’t allow for continuation, use the A* algorithm to find a route around the obstacle.

Specifically, when the robot encounter an obstacle, it will first check if there is space available upwards or downwards and move there, resorting to navigating around the obstacle only when hitting a dead end is inevitable.

Running the optimized path planning program yields the route map for the robot as shown in Figure 6. The sequence of coordinates the robot passes through, as returned by the optimized program, are as follows:

[(0, 0), (0, 1), (0, 2), (1, 3), (0, 4), (0, 5), (0, 6), (0, 7), (1, 7), (1, 6), (2, 5), (1, 4), (1, 3), (1, 2), (2, 1), (1, 0), (2, 0), (2, 1), (2, 2), (3, 3), (2, 4), (2, 5), (2, 6), (2, 7), (3, 7), (4, 6), (3, 5), (3, 4), (3, 3), (4, 2), (3, 1), (3, 0), (4, 0), (4, 1), (4, 2), (4, 3), (3, 4), (4, 5), (4, 6), (4, 7)]

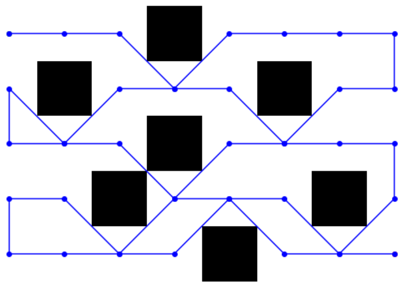

Figure 6. Robot route after optimization

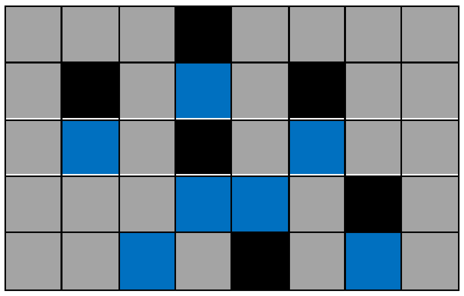

Figure 7. Robot route map after optimization

Considering the sequence of coordinates the robot passes through, as returned by the optimized program, it can be determined the number of times the robot passes through each grid point, as shown in Figure 7. After calculation, it was found that the optimized program reduces the travel route repetition rate by 45%, down to just 21%. This significantly decreases energy waste during operation and also improves the efficiency in cleaning up floating garbage on the water surface.

8. Conclusion

This article combines the Zigzag_path algorithm with the A* algorithm to propose a full-coverage path planning algorithm suitable for the designed robot's travel route. In response to the issue of high path overlap rate in the algorithm, a make_turn function is proposed to optimize the algorithm. Program code was written, and the results of the program before and after optimization were compared. It was concluded that the optimized path planning program was able to achieve full coverage of the lake surface environment, and the path overlap rate was significantly reduced.

Acknowledgments

Funded by Science Research Project of Hebei Education Department(ZD2021329).

References

[1] CARVALHO R N, VIDAL H, HIEIRA P. Complete coverage path planning and guidance for cleaning robots[C]//Proceedings of the IEEE International Symposium on Industrial Electronics, Guimaraes, Portugal: Institute of Electrical and Electronics Enginee, 1997 :677-682.

[2] ITAIA,PAP ADIMITRIO C, SZW ARCFITER L. Hamilton paths in grid graphs[J]. S IAM Journal on Computing,1982,11(4):676-686.

[3] GAGE D. Randomized search strategies with imperfect sensors[C]//Proc of SPIE Mobile Robots VI. Boston, 1993:270-279.

[4] BALCH T.The case for randomized search[C]//Proc of IEEE Inter-national Conference on Robotics and Automation. San F rancisco,CA, 2000:264-275.

[5] CHOSETH, PIGNON P.Coverage of known spaces:the Boustrophedon cellular decomposition[J]. Autonomous Robotics, 2000,9(3) :247-253.

[6] LATOME J-C, BARRAQUAND J.Robot motion planning:a distributed presentation approach[J]. International Journal of Robotics Research, 1991,10:628-649.