Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

A New Intelligent Trajectory Planning Algorithm Based on Bug2 Algorithm:Bridge Algorithm*

Shuai Yang Xinqing Wang[1] Yan Wen Jirong Wang Shiqing Li

***Abstract*— A new algorithm is developed for unmanned omni-directional robot to establish a feasible path that avoids obstacles and does not have randomness in a given environment : Bridge Algorithm. The implementation of the Bridge Algorithm is similar to the process of finding the shortest feasible path for human beings, which determines the straight line where the starting point and the end point are located, then determines the obstacles intersecting with the straight line, selects the short edge of the obstacle contour as the obstacle avoidance path, and repeats the process until the end point. The implementation ensures the consistent of the path generated by the algorithm, i.e., for the same starting point, end point and environment, the path generated each time is consistent. This consistency is crucial for application occasions that emphasize safety and stability. According to the experimental results of the algorithm in the real unmanned omni-directional wheel robot, Bridge Algorithm has good real-time performance, and the actual walking trajectory of the robot is basically the same as the planned trajectory, which proves that the algorithm has good performance.**

I. INTRODUCTION

The methods of trajectory planning for intelligent vehicles include Rapidly-exploring Random Trees (RRT) algorithm [1], ant colony algorithm [2], artificial potential field method [3], Bug algorithm, Bug2 algorithm and so on.

For people, knowing their location or starting point, can quickly plan an almost optimal path after seeing the end point and obstacles, without having to explore in space. This is because when people plan the path, they use the information of the starting point, the end point and the obstacles in turn– from the starting point to the end point, then confirm the obstacles between the starting point and the end point, and select the feasible path. The Bridge Algorithm learns the planning strategies of humans, uses the starting point, the end point and the obstacles information in the trajectory planning to perform directional optimization. In this case, only the obstacles between the starting point and the end point need to be considered, which is referred to herein as effective obstacles.

*This work is partially supported by Shandong major science and technology innovation project under the project number 2017CXGC0902, Qingdao applied basic research project (youth special project) under the project number 18-2-2-13-jch and the Fundamental Research Funds for the Central Universities under the project number 18CX02088A.

[1]Xinqing Wang is with the Automobile Manufacturing Laboratory, China University of Petroleum(East China), Qingdao, China, xqwang@upc.edu.cn

In order to realize the above information utilization strategy, we first look at this problem from a higher dimension, so that the intelligent vehicle can travel from the starting point to the end point in the two-dimensional map. When encountering effective obstacles in the process, the intelligent vehicle should be endowed with the ability to move in the three-dimensional space and directly overcome the effective obstacles. In order to enable the intelligent vehicle to have three-dimensional motion ability, it is necessary to build a "bridge" between the two points (A and B in Fig.1.), so that the intelligent vehicle can cross the effective obstacle on the "bridge". The height of the "bridge" is dertermined by the height of the effective obstacle, but the obstacle in the two-demensional map has no height. Therefore, in order to determine the height of the bridge, it is necessary to extend the two-dimensional effective obstacle to three-dimensional. This specific method takes the straight line connecting point A and point B (as shown in Fig.1) as the rotation axis, which divides the effective obstacle into two parts, then select the smaller height part of the effective obstacle to rotate 90° along the rotation axis. The outline of the part is the desired "bridge" outline, as shown in Fig.1.

After the outline of the "bridge" is obtained, it is projected onto a two-dimensional map. Flip the outline onto a two-dimensional map on the axis of the line connecting the starting and end points. In this way, a path can be built between A and B to bypass effective obstacles, and the above algorithm is called Bridge Algorithm.

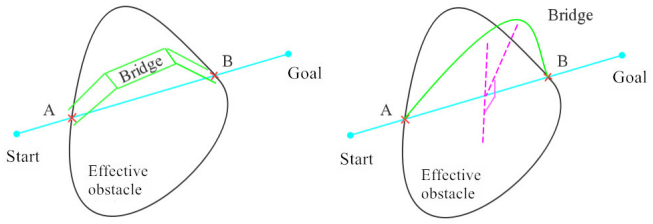


Fig. 1. Bridge Algorithm

II. RELATED WORK

A. VGRAPH Algorithm

Wesley and Lozano et al. first proposed the VGRAPH algorithm for trajectory planning [4], whose basic strategy is to transform the search problem of path into the screening

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

problem of a series of line segments. The VGRAPH algorithm is convergent and optimal in length, but it is easy to cause collision between intelligent vehicles and obstacles. Meanwhile, the VGRAPH algorithm is applicable to polygonal obstacles, but fails for circular obstacles.

In order to solve this problem, our proposed algorithm has two states: moving along a straight line toward the target and bypassing along the obstacle boundary. When encountering an obstacle of any shape, the Bridge Algorithm walks around the boundary of the obstacle.

B. Rapidly-exploring Random Trees Algorithm

Rapidly-exploring Random Trees (RRT) algorithm is an incremental sampling search method, which does not need any parameter setting in the application and has good performance and it is very suitable for solving the high-dimensional space problems. However, the shortcomings of RRT algorithm are also obvious: the computational efficiency of the algorithm is not high, and the random tree search has no purpose. In order to solve the problem of low efficiency of the RRT algorithm, many improved algorithms of the RRT algorithm emerged. For example, the RRT-connect algorithm [5] generates two trees from the initial point and the target point until the two trees are connected together and the algorithm converges. The RRT* algorithm [6] introduces the search for the neighbor nodes of the newly generated nodes, which is a breakthrough approach to solve the high-dimensional optimal path planning problem.

For higher efficiency, the Bridge Algorithm takes the straight line connecting the starting point and the end point as the direction of the robot's forward behavior, avoiding the problem of aimless search and low efficiency.

C. Bug2 Algorithm Improvements

At present, many improved algorithms of Bug2 algorithm have emerged. In [7], the researchers combined the Dubins path with a Bug2 algorithm to make the path smoother for mobile robots with non-holonomic constraints. In [8], a hybrid algorithm for the autonomous navigation of a Unmanned Ground Vehicle(UGV) based on the Bug2 algorithm and the use of the entropy of digital images as method of search has been presented. The method uses a visual topological map to autonomously navigate in the environment, which does not need know the coordinates in the enviroment unlike the classic Bug2 algorithm. In the method, the robot uses the landmark detection for know its position. A visual Bug2 algorithm is used by an Unmanned Aerial Vehicle (UAV) in typical indoor navigation tasks in [9].

III. IMPLEMENTATION OF BRIDGE ALGORITHM

Considering that the Bridge Algorithm needs to extract the contour information of obstacles, a more efficient image processing algorithm is adopted, and at the same time, the raster map expressed by matrix is transformed into the image raster map expressed by pixel.

1) the algorithm needs to input the location of global Map data Map, starting point $Start$ and end point $Goal$.

2) Connect $Start$ and $Goal$ to generate line segment L as the global path when the intelligent vehicle walks along the straight line.

3) In the direction of $Start$ to $Goal$, write the intersection of L and each obstacle contour in the Map into the cell $Messenger$.

4) Determine the intersection of L and the obstacle: $i = 1$, if $i <= 0.5 * Length(Messenger)$, it means that L passes at least one obstacle, records the intersection of L and the obstacle, and H_s near $Start$, close to $Goal$ is H_g, then calculate the distance from H_s clockwise and counterclockwise around the obstacle boundary to H_g, denoted as Dcw and $Dccw$, if $Dcw <= Dccw$, select clockwise for the detour direction, otherwise choose counterclockwise for the direction of the detour. then $i++$, the above calculation process is repeated. If $i > 0.5 * Length(Messenger)$, it means that L has already crossed the boundary of the last obstacle, and $Goal$ is added to the path.

Algorithm 1 Bridge Algorithm

```
P.init(Map, Start, Goal)
L.generate(Start, Goal)
while True do
    From Start,move toward Goal along L
    if an obstacle is encountered at hit point H_i then
        Messenger ← H_i
    end if
    if Goal is reached then
        break the loop
    end if
end while
for k = 1, i = 1, k <= 0.5 * Length(Messenger), k++, i++ do
    H_s ← Messenger(i)
    H_g ← Messenger(i++)
    path_c ← go clockwise from H_s to H_g
    path_cc ← go counterclockwise from H_s to H_g
    PATH ← min(path_c, path_cc)
end for
return PATH
```

In order to clearly see the excellent performance of the Bridge Algorithm,we firstly take a look at the implementation of Bug2 algorithm, then we will show the difference between the two algorithms in performance with the actual results.

From the implementation process of the two algorithms, when the Bug2 algorithm encounters obstacles, it is only a mechanical direct left turn or right turn instruction. The choice of left turn or right turn is specified in the program.

Algorithm 2 Bug2 Algorithm

Require: The starting point and the end point
Ensure: A path to $Goal$ or a conclusion no such path exists

```
while True do
    From Start move toward Goal along L
    if an obstacle is encountered at hit point H_i then
        turn left (or right)
    end if
    if H_i is re-encountered then
        failure
    end if
    if Goal is reached then
        break the loop
    end if
end while
return PATH
```

However, the boundary of obstacles is irregular in most cases, so the mechanism of Bug2 algorithm will lead to the lengthy path.

From the perspective of the implementation mechanism of the Bridge Algorithm, when encountering obstacles at hit point H_i, the Bridge Algorithm will find the next point H_{i+1} where intersects the obstacles along the direction of L, and calculate the shortest distance from H_i to H_{i+1}, taking the shortest path as the final path.

Fig.2 is the performance result of Bug2 algorithm in a specific environment. It can be seen from the figure that no matter how to choose the starting point, Bug2 algorithm always rotates in the counterclockwise direction, even if the path is very lengthy.

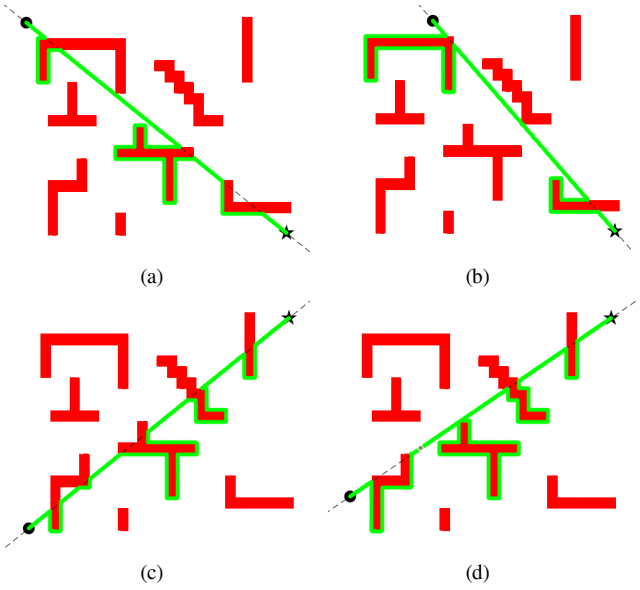


Fig. 2. The result of Bug2 algorithm

Fig.3 shows the results of the Bridge Algorithm in the same environment. It can be seen that when changing different starting positions, the Bridge Algorithm will adjust the direction of the detour to select the shortest path.

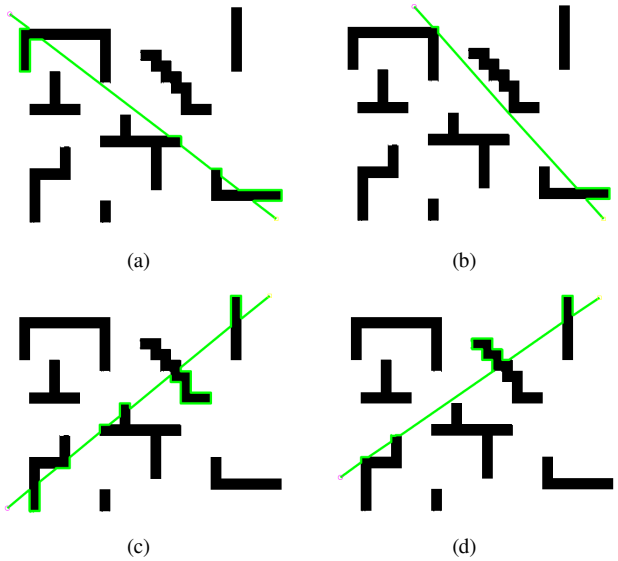


Fig. 3. The result of Bridge Algorithm

Tab.I shows the analysis of the execution results of the above two algorithms, including path length, average execution time and operating system platform. From the data in the table, it is clear that the Bridge Algorithm is better than the Bug2 algorithm in terms of path length and calculation time.

TABLE I
ALGORITHMS EXECUTION COMPARISON

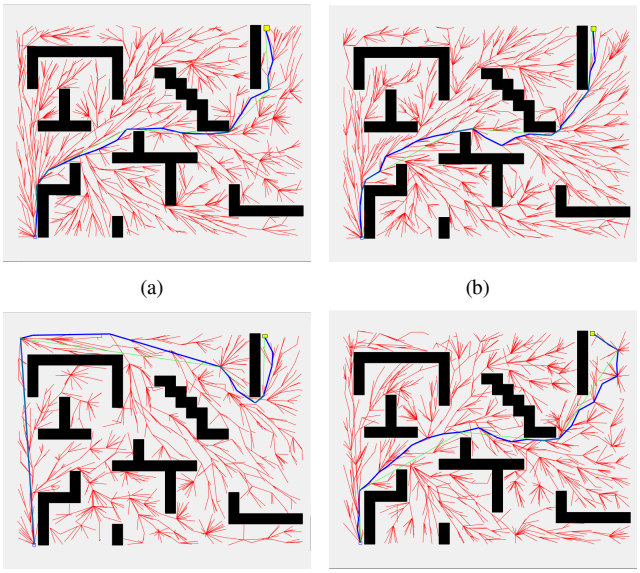
Algorithm	Length of Path	Execution Time (ms)	Test Environment
Bridge Algorithm	(a) 828.4 (b) 612.1 (c) 826.7 (d) 720.2 Ave: 746.85	(a) 26.36 (b) 20.47 (c) 28.91 (d) 28.06 Ave: 25.95	Intel i5-5200U
Bug2 Algorithm	(a) 966.4 (b) 875.2 (c)1045.3 (d)1306.7 Ave: 1048.4	(a) 79.81 (b) 70.62 (c) 98.51 (d) 80.92 Ave: 82.47	Intel i5-5200U

In the ”Length of Path” group (d) data, the path length generated by Bridge Algorithm even is only about half of that of Bug2 algorithm. From the average length of path data, we can see that the length of path generated by Bridge Algorithm is far shorter than that of Bug2 algorithm. From the ”Length of Path” group data, in these four experiments, we can find that the longest path length generated by the

Bridge Algorithm is 828.4, the shortest is 612.1, and the maximum change range is 212.3. While the longest path length generated by Bug2 algorithm is 1306.7, the shortest is 875.2, and the maximum change range is 431.5. Therefore, compared with Bug2 algorithm, the path length generated by the Bridge Algorithm is more stable and the change range is smaller.

From the "Execution Time" group, we can see the more obvious difference. The average execution time of Bug2 algorithm is more than three times that of Bridge Algorithm. The Bridge Algorithm has a complete advantage in real-time.

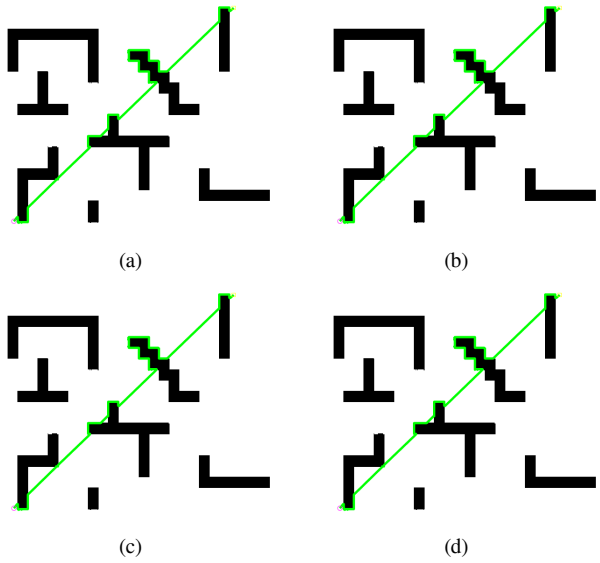
At the same time, we compare the Bridge Algorithm with RRT star algorithm which also has good performance. Fig.4 is the experimental results of four times RRT star algorithm under the same starting point, end point and experimental conditions. We can find that RRT star algorithm has excellent performance. The paths it generates are not only relatively smooth, but also rarely redundant. But RRT star algorithm also has a big disadvantage. From Fig.4, we can see that the path generated by RRT star algorithm is random. Under the same starting point, end point and experimental conditions, each experimental path is different. This is due to the principle of RRT star algorithm.



(a) (b) (c) (d)

Fig. 4. The result of RRT star algorithm

Fig.5 shows four experiments of the Bridge Algorithm under the same starting point, end point and experimental conditions. We can see that the paths generated by four experiments of the Bridge Algorithm are consistent. There is no randomness in the implementation mechanism of the Bridge Algorithm. This kind of consistency and stability is very important for the application of many industrial products.



(a) (b) (c) (d)

Fig. 5. The result of Bridge Algorithm

IV. DESIGN PATHER APPLICATION FOR TRAJECTORY PLANNING

In order to make the use of Bridge Algorithm and other trajectory planning algorithms more convenient, we have designed the trajectory planning application software Pather. Pather is designed to incorporate the improved RRT algorithms and Bridge Algorithm into an application that can be installed and run independently on a computer.

Pather's menu design is shown in Fig.6. Users can call different trajectory planning algorithms through various commands in the software menu, and arbitrarily set different grid maps, starting points and end points (domains) for trajectory planning, and conveniently save the current path. Pather software provides IRRT algorithm, GIRRT algorithm(they are the improved RRT algorithms.) and Bridge Algorithm to choose from. In addition, starting point and end point (domain) can be selected on the map through the cursor.

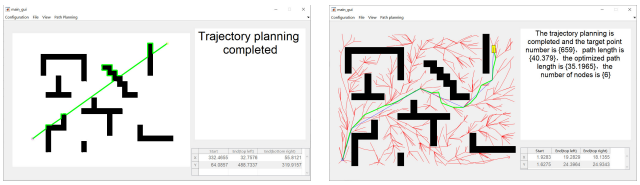
To facilitate user analysis of the results of trajectory planning, Pather provides a concise Graphical User Interface (GUI). In the GUI, the optimal path, obstacles, starting point and end point (domain) can be displayed in the graphics window in whole or in part. In addition, Pather will be able to store the best planned approach for practical application or follow-up research.

Fig.7 are the GUI of Pather software using Bridge Algorithm (a) and IRRT algorithm (b) for trajectory planning respectively. In addition to menu items, graphical window and information table show the starting and end (domain) location and an operation prompt window. When using IRRT algorithm and GIRRT algorithm for trajectory planning, the operation prompt window will display the length of path, number of nodes and other information after the completion

- Configuration
  - IRRT algorithm
  - GIRRT algorithm
  - Bridge Algorithm
- File
  - Import the map
  - Save the trajectory
  - Quit
- View
  - RRT view
    - Obstacles
    - Start and end fields
    - Search tree
    - Optimal path
    - Optimized optimal path
    - Plan again on the current map
      - Keep the current start and end fields
      - Keep only the current map
  - Bridge Algorithm view
    - Path
    - Start and end fields
- Path planning
  - Enter the starting and ending fields
  - Start the calculation
  - Clear current content

Fig. 6. Pather software menu structure

of trajectory planning.



(a) (b)

Fig. 7. GUI of Pather software

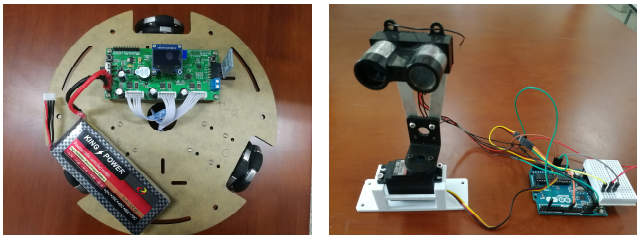
V. EXPERIMENT

In order to verify the performance of the Bridge Algorithm applied to the actual vehicle, we built a test platform of omni-directional wheel unmanned vehicle.

The experimental platform is standard omni-directional wheel car as shown in Fig.8. The car adopts STM32F103C8T6 microcontroller, and three DC motors drive omni-directional wheel through a reducer with a transmission ratio of 30:1. Each motor is equipped with 390 AB phase incremental hall encoder. The car communicates with the computer via zt-040 bluetooth.

In order to ensure the motion accuracy, PID control [10] is adopted, and each motor is controlled by two PID loops, in which the outer loop is the position loop, the inner loop is the speed loop.

Lidar-Lite V1 single-line laser radar, hitec-hs422 steering gear, Arduino UNO development board and other equipment are used to build the laser radar scanning platform for the experimental platform. The lidar is installed on the output shaft of the steering engine, which can rotate reciprocally within a range of 0-180°. The lidar platform is shown in Fig.8(b). The radar rotates $0.5°$ and samples once every 40ms, and the sampled data and angular position are returned through the computer serial port in real time.



(a) Omni-wheel car (b) Lidar platform

Fig. 8. Experimental platform

We set up the experimental environment as shown in Fig.9.

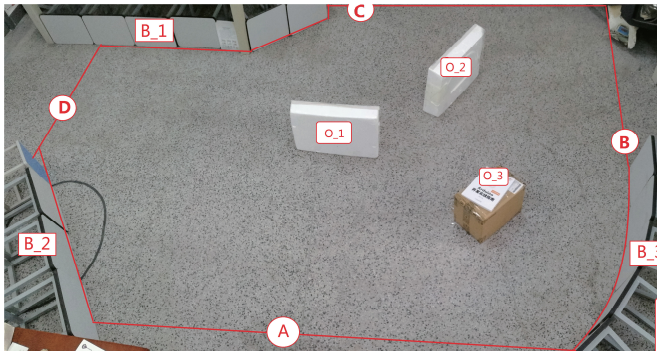


Fig. 9. Experimental environment

We get the feature map of the experimental environment through the experimental platform, because the feature map of the discrete contour is unsatisfactory for the expression of obstacles and boundaries, the feature map of the discrete contour is converted into a grid map according to the occupation of the grid map theory [11]. At the same time, in order to avoid the collision of the unmanned vehicle due to its actual size and obstacles, it is necessary to expand the obstacle in the map outward by half of the maximum size of the intelligent vehicle. After the above processing, the obtained grid map is as shown in Fig.10.

The experimental results are shown in Fig.11, where the solid green line is the planned path and the dotted red line is the actual trajectory of the unmanned vehicle. It can be seen

Fig. 10. The grid map of experimental environment

that in addition to some deviation at the corner and some deviation about 1cm near the end point, the unmanned vehicle basically track the route accurately. In fact, the deviation about 1cm of near the end point is caused by the problems of lidar accuracy and odometer sensor accuracy. For any lidar or odometer sensors, there will always be errors, especially when turning, and in the process of data calculation there will also be the loss of data accuracy. After comprehensive consideration, the deviation is within the acceptable and normal range. In a word, the experimental results can prove that the Bridge Algorithm can be easily applied to the actual unmanned vehicle, and has high accuracy and real-time performance, even in the low performance processor.

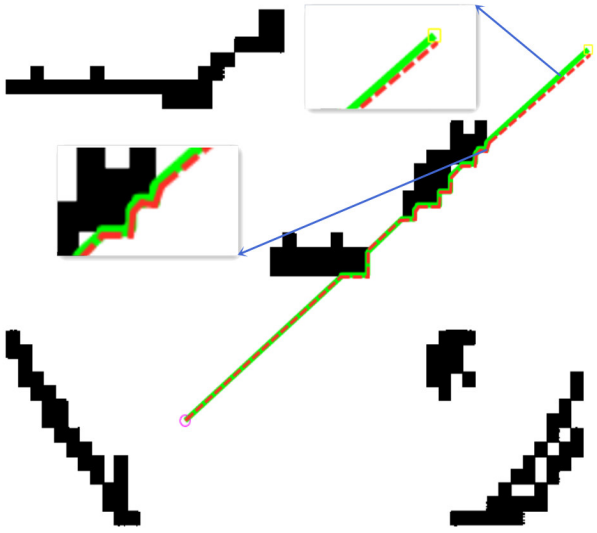


Fig. 11. Experimental results

VI. CONCLUSIONS

An excellent trajectory planning algorithm is crucial to the safe and efficient driving of unmanned vehicles. In this paper, we proposed the Bridge Algorithm based on Bug2 algorithm. The path generated by Bridge Algorithm is shorter and better than the path generated by Bug2 algorithm under the same conditions with less computing time, compared with RRT star algorithm, the generated path is more stable and efficient. The experimental results verify the possibility and feasibility of the Bridge Algorithm in the unmanned vehicle. Under the condition of general lidar and odometer sensor configuration, it can still maintain relatively high trajectory accuracy, with better robustness and stability. But at the same time, the Bridge Algorithm also has some shortcomings to be improved, such as it can not be applied to the locomotive with nonholonomic constraints and the generated path is not smooth enough.

REFERENCES

[1] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," *Technical Report. Computer Science Department*, Iowa State University, 1998.

[2] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29-41, Feb. 1996.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, pp. 500-505, 1985.

[4] L. P. Tomás and M. A. Wesley. "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles," *Commun. ACM*, vol. 22, pp.560-570, 1979.

[5] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 995-1001, 2000.

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846-894, 2011.

[7] Q. Xu, T. Yu and J. Bai, "The mobile robot path planning with motion constraints based on Bug algorithm," *2017 Chinese Automation Congress (CAC)*, Jinan, pp. 2348-2352, 2017.

[8] D. Maravall, J. D. Lope, and J. P. Fuentes." Visual Bug Algorithm for Simultaneous Robot Homing and Obstacle Avoidance Using Visual Topological Maps in an Unmanned Ground Vehicle," *Bioinspired Computation in Artificial Systems*, 2015.

[9] D. Maravall, J. de Lope, and J. P. Fuentes, "Navigation and Self-Semantic Location of Drones in Indoor Environments by Combining the Visual Bug Algorithm and Entropy-Based Vision." *Frontiers in neurorobotics*, vol. 11, 2017.

[10] K. J. Astrom and T. Hagglund, "PID Controllers, Theory, Design and Tuning," 2nd Edition, *Instrument Society of America*, 1995.

[11] M. E. Bouzouraa and U. Hofmann, "Fusion of occupancy grid mapping and model based object tracking for driver assistance systems using laser and radar sensors," *IEEE Intelligent Vehicles Symposium*, pp. 294-300, 2010.