Proceedings of 2021 IEEE
International Conference on Mechatronics and Automation
August 8 - 11, Takamatsu, Japan

D* Lite-Based Navigation Algorithm for Multiple Spherical Underwater Robots Collaboration

Awa Tendeng[1], Shuxiang Guo[2,3]
[1]Graduate School of Engineering
[2]Department of Intelligent Mechanical Systems Engineering
Faculty of Engineering, Kagawa University
2217-20, Hayashi-cho, Takamatsu, Kagawa, Japan
s20g517@stu.kagawa-u.ac.jp

Ruochen An[1], and Chunying Li[1]
[3]Key Laboratory of Convergence Medical Engineering
System and Healthcare Technology, The Ministry of Industry
and Information Technology, School of Life Science
Beijing Institute of Technology, Beijing 100081, China
guo@eng.kagawa-u.ac.jp

***Abstract* – This research investigates the multiple robotic systems navigation in an unknown environment problem. As an attempt to solve this challenge, a navigation strategy based on D* Lite search algorithm is proposed. To ensure an optimal path to the goal, we implemented the D* Lite algorithm and improved the safety of the algorithm by defining the non-reachable cells accordingly to the size and position of the obstacles. The collaborative navigation is ensured by a fault tolerance strategy based on a low computation cost fault detection and leader switching algorithm. The performed experiments in a water tank and the simulation demonstrated that the search algorithm can safely find the shortest path to the goal. The robustness of the path planning algorithm is demonstrated further with relatively few collisions occurrence. Also, the simulation shows that the robots can safely adapt their formation from V-shape to line shape when obstacles are present in the environment. However, the formation navigation presents some inefficiencies in terms of accuracy. The developed strategy will be extended in future works with more experiments.**

Index Terms* – *D* Lite, spherical underwater robot, collaborative navigation, fault tolerance, gazebo.

I. INTRODUCTION

Current developments in the field of robotics have led to a growing interest in the application of robots in underwater operations. The intervention of a single AUV in marine environment requires the equipment of that robot with several sensors and other onboard electronics, making the overall system very heavy. In addition, the robot has to stay compact in an attempt to achieve greater flexibility and maneuverability. These technical constraints suggest the use of a team of robots in underwater missions.

A crucial issue to consider when deploying a fleet of robots is the safety and reliability of navigation, especially in a challenging environment. The marine environment presents a particular complexity highlighted by the dynamics of the water and the presence of a priori unknown different types of obstacles. Several studies are conducted on how to move a robot from a starting point to the target while considering these constraints. The problem of optimal path finding in a dynamic unknown or partially known environment is widely explored in the literature [1]-[3]. With a large variety of solutions proposed so far [4]-[9]. [1] proposes a classification of those methods in two categories: Geometric Model Search Methods and Bionics search methods. Geometric methods are widely used in path planning problems and show strong performances and less computation [2]. Among those methods, those based on heuristic search like Djikstra, A*, D*(Dynamic A*) and D* Lite have gained wide spread application in autonomous vehicle and attract the interest of the research community [4][5]. D* Lite algorithm is based on Djikstra but is more efficient for dynamic search. Its suitability for underwater dynamic path search has been demonstrated in [5] and [6]. [7] and [8] have proven the efficiency of D* Lite for an AUV through 2D and 3D simulations.

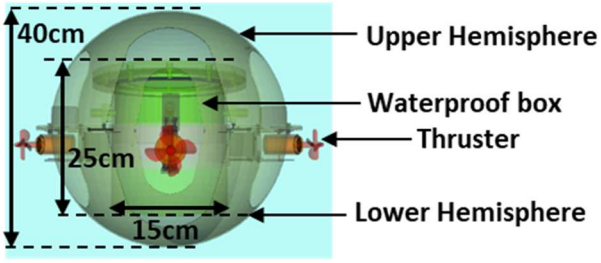
In collaborative navigation, robots must be able to move while being aware of the position of other robots and maintaining a coordinated formation. Different coordinated navigation approaches have been explored with great attention [10]-[14]. [12] proposed a blockchain-based navigation coordination algorithm for autonomous robots. This algorithm has been implemented on the amphibious spherical underwater robots (SUR) in terrestrial environment with satisfactory performance and has solved the problem of mission tracking when the group leader is out of service. The main drawback of this algorithm is the computation cost and the need for reliable communication. Indeed, the fault tolerance relies on the possibility of electing a new leader by the non-faulty robots.

Considering the constraints related to communication in underwater environment, the development of a navigation strategy requiring a minimum of communication is necessary. The effectiveness of the navigation depends also on the use of an efficient algorithm to find the shortest path between the starting point and the goal as well as to update the path as the navigation progresses according to the presence of obstacles. In this work, we implemented the D* Lite algorithm with consideration of the position and size of the obstacles as well as the preservation of a relative safety distance between the obstacles and the robots. The overall effect is to adjust the formation's shape of the robots according to the presence of obstacles.

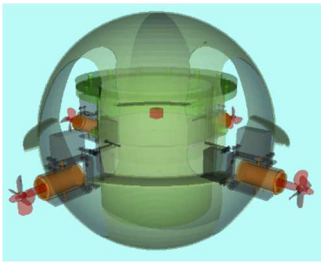
This work is organized as follow: Section II gives the details of the spherical underwater robot's structural design, section III explains the navigation approach, section IV presents the strategy to control the robots' formation and tolerance to failure, section V is an exposition of the results obtained by carrying out an experiment of the D* Lite algorithm and simulating the navigation of a set of robots in Gazebo, and section VI finally gives the conclusion of this work and some future perspectives.

II. Structural Design Of The SUR

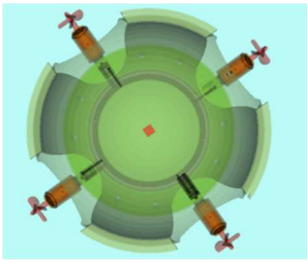
Several versions of the spherical underwater have been designed in the past. In its current version, the main components of the spherical robot are the hull composed of two hemispheres, a cylindrical waterproof box, the servomotors and the thrusters. Fig.1 shows the structural design of the robot. The two hemispheres, made with acrylic material, are designed to bear the pressure at a water depth of at least 8 m. Each of them has a diameter of 40cm and 3mm thickness. Since the hemispheres are not waterproof, the robot is equipped with a waterproof box that contains electronic components such are the different sensors, batteries, and onboard controllers. The version used in this research is composed of 8 servomotors and 4 propellers to steer the robot in a 3D motion.



(a)Front view



(b) Side view



(c) Top view

Fig. 1 Structure of the spherical underwater.

III. Unknown Environment Navigation

A. Goal-directed navigation approach

Planning a trajectory is a key task for an underwater vehicle's navigation. When the optimal trajectory is planned in a non-deterministic domain, only local navigation can be considered. A perception of the environment is provided by the robot sensors (camera, acoustic beam, etc.). This information is processed to detect the presence of eventual obstacles and to adjust the robot behavior accordingly. The selection of an optimal and dynamic path planning technique thus becomes the most important step in the navigation task. In the goal-directed navigation, the position of the robot is first initialized on the map and the goal settled. The robot computes the shortest path to the goal from the starting points by considering the a priori known obstacles as non-reachable cells, then follows that path. During the navigation to the goal, the agent receives pose estimation and environment information from its sensors, then updates the map vertex by assigning a reachable or unreachable label to cells and, and finally computes the shortest path from its current position to the fixed goal. The shortest path depends on the location of traversable cells as well as the cost of moving from the robot's position to an adjacent cell. From any cell belonging to the computed path, its successor is, therefore, the most optimal with minimum cost. Among the various algorithms presented in the literature, the D* Lite [4][5] offers many advantages in terms of finding the optimal solution and minimizing computation cost, with optimal path finding when implemented in underwater as well as terrestrial environment.
The D*Lite is an optimization of the D*(Dynamic A*) which is based on LPA*(Longlife Planning A*), an incremental version of A*. LPA* is very useful in graph search where vertices are added or deleted over time

D* efficiently recalculate the shortest path from the robot's current position by only computing the goal path that has been modified. Which means that even if the environment change, the algorithm will not recompute the path unless a change occurs on the previously computed path.
In Fig.2 we have an example of D* Lite search. The initial path, in blue, is recalculated when some obstacles are detected on the path. The obtain path, in green isn't recompute when the changes that occur in the environment don't concern the path.

B. D* Lite Implementation

D* Lite repeatedly determines the shortest path from the robot's current position, s_{start} vertex to the goal vertex s_{goal}. The environment is modeled as an eight fully-connected graph.
D* is based on incremental search. The expansion direction of the search is given by the *rhs(s)* value (1). The *rhs(s)* is a heuristic that corrects the cost value in case of local inconsistency: when the cost of moving to edge s is different from the path length to *s*. From the robot's current position, the algorithm calculates the cost of the 8 surrounding vertices. The lowest cost will give the next point of the path.
To estimate the value of the grid's edges, the algorithm uses priority queue arrangement parameters. The priority vertex to expend in the priority queue is determined by the key $k(s)=[k_1(s); k_2(s)]$.

The key k is a value for comparison. The formula to calculate it is given in (2) and (3).

$$rhs(s) = \begin{cases} 0 \; if \; s = start \\ min_{s' \in pred(s)}(g(s') + c(s,s')) \end{cases} \quad (1)$$

Where, S is the finite set of vertices of the graph, $Succ(s) S \subseteq S$ is the set of successors of the vertex $s \in S$, $c(s,s')$ is the cost of moving from edge s to edge s', $g(s)$ is the total length from start to edge *s*.

$$k_1(s) = min(g(s), rhs(s)) + h(s)) \quad (2)$$

$$k_2(s) = min(g(s), rhs(s)) \quad (3)$$

Where $h(s,goal)$ is the goal distance from the vertex *s*.

The steps of the algorithm are summarized given in table 1. To increase the safety of the path, we added to the algorithm a vertex update program that provides the required change to the vertex when an obstacle is detected. Fig.1 summarizes the working principle of the navigation strategy: the algorithm computes the shortest path from the robot's current position to the goal. If some obstacles are detected, then the edges of the graph are updated according to the size of obstacles and the shortest path recomputes again.

TABLE I

D* Lite algorithm

Procedure CalculateKeys(s)
{01} Return $[min(g(s), rhs(s)) + h(S_{start}, s) + k_m; min(g(s), rhs(s))]$
Procedure Initialize()
{02} $U = \emptyset$;
{03} $k_m = 0$
{04} For all $s \epsilon S\ rhs(s) = g(s) = \infty$;
{05} $rhs(S_{goal}) = 0$;
{06} $U.Insert(S_{goal}, [h(s_{start}, k_{new}); 0])$;

Procedure UpdateVertex(u)
{07} $if\ g((u) \neq rhs(u)\ \ AND\ u \in U)\ U.Update(u, calculateKey(u))$;
{08} $else\ if\ (g(u) \neq rhs(u)\ AND\ u \notin U)\ U.Update(u, calculateKey(u))$;
{09} $else\ if\ (g(u) = rhs(u)\ AND\ u \in U)\ U.Remove(u)$;

Procedure UpdateEdgesCost(S)
{10} $If\ obstacle == True$;
{11}$Compute\ non\ passable\ cells(a, size\_a)$;
{12}$Update\ edges\ cost$;

Procedure ComputeShortestPath()
{13} $While\ (U.YopKey(\ \ ) < CalculateKey(s_{start}) or\ rhs(s_{start}) > g(s_{start}))$
{14} $u = U.Top()$;
{15} $k_{old} = U.TopKey()$;
{16} $k_{new} = CalculateKey(u)$;
{17} $if\ (k_{old} < k_{new})$
{18} $U.Update(u, k_{new})$;
{19} $else\ if\ (g(u) > rhs(u))$
{20} $g(u) = rhs(u)$;
{21} $U.Remove(u)$;
{22} $for\ all\ s\ Pred(u)$
{23} $if\ (s \neq s_{goal})rhs(s) = min(rhs(s), c(s, u) + g(u))$;
{24} $UpdateVertex(s)$;
{25} $Else$
{26} $g_{old} = g(u)$;
{27} $g(u) = \infty$;
{28} $for\ all\ s\ Pred(u) \cup \{u\}$
{29} $if(rhs(s) = c(s, u) + g_{old})$
{30} $if\ (s \neq s_{goal})rhs(s) = min_{s' \epsilon Succ(s)}(c(s, s') + g(s'))$;
{31} $UpdateVertex(s)$;

Procedure Main ()
{32} $s_{last} = s_{start}$;
{33} $Initialize()$;
{34} $ComputeShortestPath()$;
{35} $While(s_{start} \neq s_{goal})$
{36} $/*\ if\ rhs(s_{start} = \infty)\ then\ there\ is\ no\ known\ path\ */$
{37} $s_{start} = \arg\min_{s' \epsilon Succ(s_{start})}(c(s_{start}, s') + g(s'))$;
{38} $Move\ to\ s_{start}$;
{39} $Scan\ graph\ for\ changed\ edge\ cost$;
{40} $If\ any\ edge\ cost\ changed$
{41} $k_m = k_m + h(s_{last}, s_{start})$;
{42} $s_{last} = s_{start}$;
{43} $For\ all\ directed\ edges(u, v) with\ changed\ edge\ costs$
{44} $c_{old} = c(u, v)$;
{45} $Update\ the\ edge\ cost\ c(u, v)$;
{46} $if\ c_{old} > c(u, v$
{47} $if\ (u \neq s_{goal})\ rhs(u) = min(rhs(u), c(u, v) + g(v))$;
{48} $else\ if\ rhs(u) = c_{old} + g(v))$
{49} $if\ (u \neq s_{goal})\ rhs(u) = min_{s' \epsilon Succ(u)}(c(u, s') + g(s'))$
{50} $UpdateVertex(u)$;
{51} $ComputeShortestPath()$;

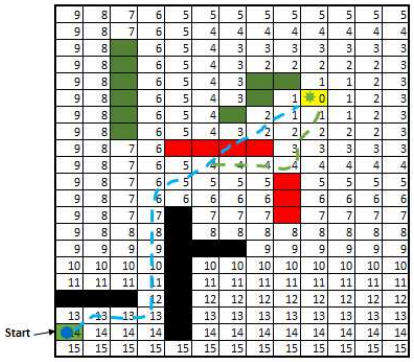


Fig. 2 Example of shortest path computation.

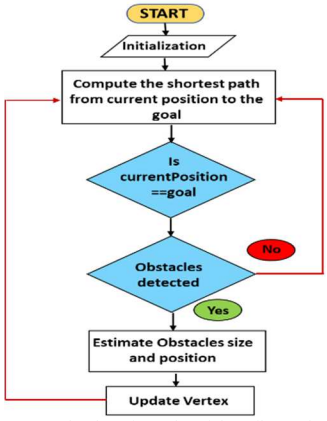


Fig. 3 Path planning algorithm flow-chart.

IV. Formation Control And Fault-Tolerance Strategy

A. *Multi-robots Formation control method*

This section describes the details of the formation strategy. Two distinct types of formation are considered. When the robots are in an obstacle-free domain, they implement a V-shape formation and when obstacles are encountered, they favor a line formation to diminish the computation cost by allowing only the leader to computes the optimal path and also to increase the safety of the overall system.

The model presented here assumes that each robot is aware of its global position and is able to estimate the position of the surrounding robots using sensors.

The robot's formation is based on a trigonometric formulation. Let's consider a virtual leader placed on the gravity center of the formation figure. The movement of the leader is a simple translation on the horizontal plane. In order to keep the initial formation, the followers need to calculate their positions according to the virtual leader. In a way such the initial formation is conserved.

The initial position of the robots determines the formation to keep during the navigation. Fig.4 presents the initial positions of three robots in the case of a triangle formation.

The Euclidian coordinates of the virtual leader and the followers are respectively $L = [x_{VL}(0), y_{VL}(0), z_{VL}(0)]^T$ and $F_i = [x_{Fi}(0), y_{Fi}(0), z_{Fi}(0)]^T$.

During the navigation, the robots should keep their initial formation. We assume that the robot's positions along the

vertical axis don't change much, $z = z_i$. Therefore, we can ignore the variations along the vertical axis and only consider the horizontal plane. The formation parameters (4) are the Euclidian distance between each follower and the virtual leader d_{Fi} and the angle between them θ_{Fi}.

$$\begin{cases} d_{Fi} = \sqrt{\left(x_{VL}(0) - x_{Fi}(0)\right)^2 + \left(y_{VL}(0) - y_{Fi}(0)\right)^2} \\ \theta_{Fi} = tan^{-1}\left(\dfrac{y_{VL}(0) - y_{Fi}(0)}{x_{VL}(0) - x_{Fi}(0)}\right) \end{cases} \quad (4)$$

By neglecting the error induced by water dynamics, the position of each follower is deducted from the virtual leader position (Fig.5). The coordinates of the follower *Fi* are given in (5), where $\alpha(t-1)$ is the motion direction of the virtual leader between positions *k-1* and *k.*

Depending on the values of θ_{Fi} and d_{Fi}, the formation shape can be a triangle as well as a circle, or any other geometric shape.

$$\begin{cases} x_{Fi}(k) = x_{VL}(k) + d_{Fi}cos(\theta_{Fi} + \alpha(t-1)) \\ y_{Fi}(k) = y_{VL}(k) + d_{Fi}sin(\theta_{Fi} + \alpha(t-1)) \\ z_{VL}(k) = z_{Fi}(k) \\ \alpha(t-1) = tan^{-1}\left(\dfrac{y_{VL}(k) - y_{VL}(k-1)}{x_{VL}(k) - x_{VL}(k-1)}\right) \end{cases} \quad (5)$$

However, in the case where the robots have to switch to a line formation, a leader is necessary in order to compute the path planner algorithm. Therefore, the follower 1 becomes leader and each robot adjusts its rank($F^{i+1} \leftarrow F^i$) in a way such the distance between two consecutive robots is equal to the distance a distance *d* to define.

Let's consider two consecutive positions of a successor robot *k-1* and *k*. The knowledge of two successive positions of the robots enables to determine the direction of its motion. Thus, each following robot will carry out its motion by only considering its predecessor's direction of movement and the distance to be maintained.

B. Fault adaptability strategy

When the adopted formation is not a line, the system is naturally fault tolerant because the failure of a multi-robot's system is generally caused by the fault occurring on the physical leader. In the case of the line formation, we considered the case where a leader robot has encountered a fault and is not able to move anymore to the goal, without necessarily being aware of its failure or being able to notify the followers.

At the beginning, the robot checks if its status is *leader.* if it is the case, it moves to the goal by computing the shortest path. A *follower* robot ascertains repeatedly the position of the leader and updates its own position accordingly. When the leader has not yet reached the goal and remains motionless for a period of time *T*, it is considered as encountering a fault. Therefore, *follower1* becomes the leader and the other followers update their ranks as follows: $F^{i+1} \leftarrow F^i$. Fig.6 gives the steps of the fault-detection and adaptation strategy.

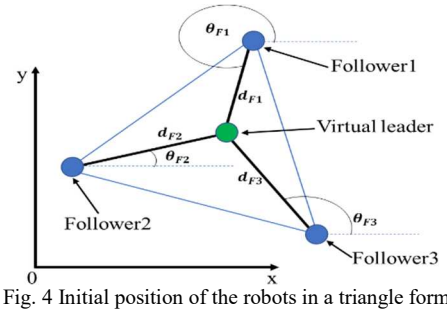


Fig. 4 Initial position of the robots in a triangle formation.

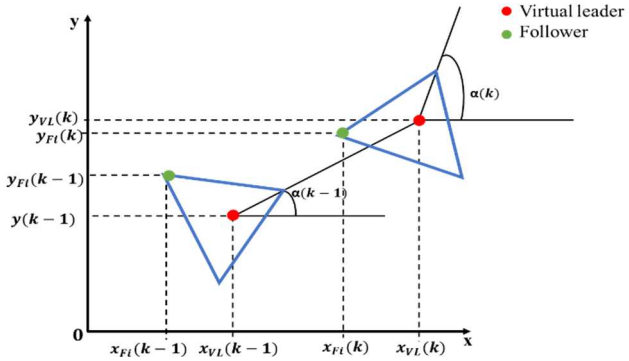


Fig. 5 Virtual Leader and followers' positions during navigation in triangle formation.

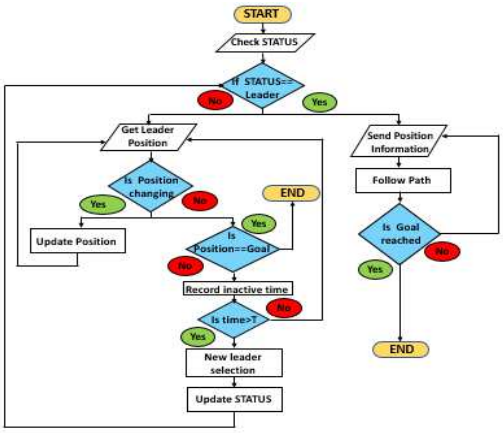


Fig. 6 Fault-tolerance algorithm flow-chart.

V. Simulation and Experimental Results

A. Simulation setup

To Evaluate the effectiveness of the D* Lite search algorithm and the formation control, we conducted a simulation on Gazebo platform. gazebo9 release and Ros Melodic were installed on an Ubuntu 18.04 running computer. The characteristics of the computer are 3Ghz processor and 8GB RAM. The simulation is performed in 2 steps. First, a single robot is operated to test the robustness of the search algorithm and then a team of 3 robots is deployed to verify the feasibility of our navigation strategy.

B. Evaluation of the path planning algorithm

One robot is launched on the simulation and some obstacles generated (Fig.7) during the navigation. Each time an obstacle is generated, the robot receives an estimate of the position and size of the obstacle. Fig.7b and Fig.7c show the robot's navigation.

The robot is able to go through the obstacle zone and arrive safely to the goal as shown in Fig.7d, Fig.7e and Fig.7f. The results of the simulation demonstrate the optimality of the path generated by the algorithm. Also, by comparing the path followed by the robot and the one generated by the algorithm (Fig. 8), we can notice that the robot follows the generated path with enough smoothness. The offset between the two is slightly significant in some areas. However, it can be reduced with an improvement of the control algorithm in a way to reduce the influence of the water dynamics on the robot's motion.

D. Environment adaptability strategy analysis

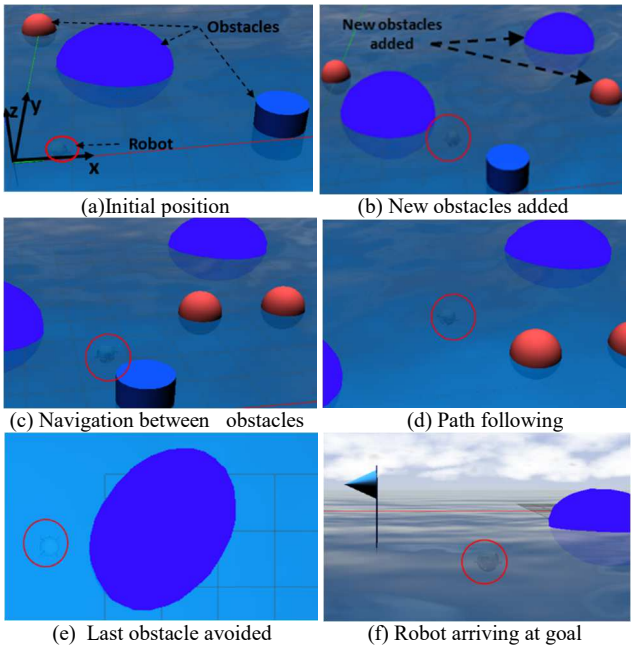
In this part of the work, a team of 3 robots (Fig.9) is launched to evaluate the robustness of the strategy. We first assigned the status *leader*, *follower1* and *follower2* to the robots. At the beginning, the robots maintained a triangle formation of parameters $d_{F1} = 0.5$, $d_{F1} = d_{F2} = \sqrt{2}$, and $\theta_{F1} = 0, \theta_{F2} = 3\pi/4, \theta_{F3} = 5\pi/4$.

When the first obstacle is detected, the SURs switch to line formation, so the safety of the navigation is preserved. Only the leader computes the shortest path to the goal. The other robots adjust their positions according to the leader's position while maintaining a distance of 1 meter at least.

Fig.11.b and Fig11.c show the robots switching from triangle to line formation and continuing the motion with the new formation. However, some problems are noticed in maintaining the formation. The first factor that explains this issue is the fact that each follower only considers the position of the robot in front of it and the second is the effect of the water dynamics on the motion of the robots.

The fault-tolerance is evaluated by inducing a fault on the leader. The other robots are able to continue the navigation to the goal as shown in Fig.9.b and 10.

Thus, the feasibility and the robustness and of the algorithm are demonstrated.



(a)Initial position (b) New obstacles added

(c) Navigation between obstacles (d) Path following

(e) Last obstacle avoided (f) Robot arriving at goal

Fig. 7 Single robot navigating in complex environment.

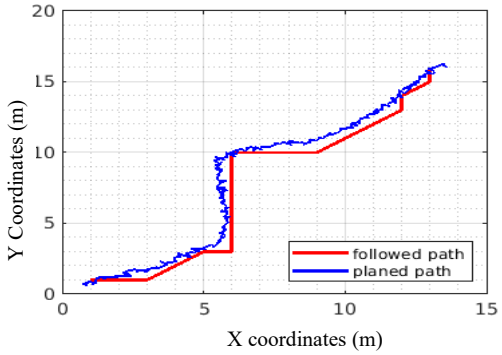
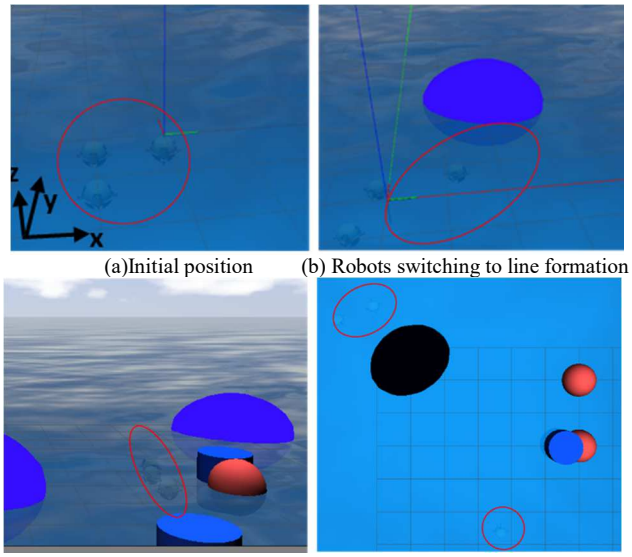


Fig. 8 Comparison of planned path and robot's path on the horizontal plane.



(a)Initial position (b) Robots switching to line formation

(c) Path following (d) Faulty robot stopped

Fig. 9 Multiple robot's navigation in formation

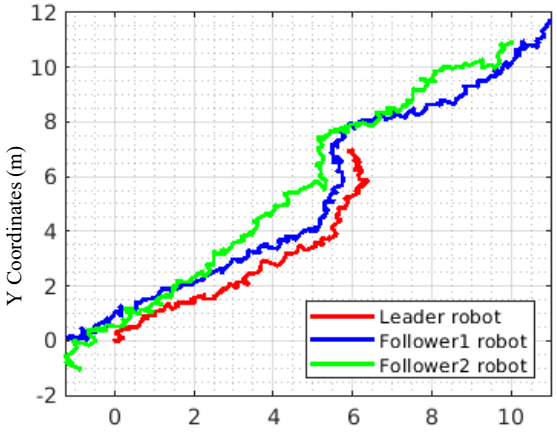


Fig. 10 Fault-tolerance navigation.

E. Experimental results of the D Lite algorithm*

The experiment is carried out in a water tank of 1.5m large, 2.5m long, and 0.5m depth. The robot is equipped with an IMU sensor that provides an estimate of its location and a Bluetooth communication is used to indicate to the robot the position of the obstacles. Considering the limitation of Bluetooth communication in an underwater environment, the robot

navigation is constrained to the surface of the water to keep the communication liaison. Initially, the robot (in the black triangle) was placed at x=6cm and y=2cm, and the goal (black circle) coordinates settled at x=2cm, and y= 23cm as shown in Fig.11. The tank is assimilated to an 8 connected-lattices of 50cm distance from one edge to the nearest. Some obstacles (red circles) were added just at the beginning of the navigation and theirs positions and estimated sizes sent to the robot. The robot computes the shortest path then starts navigating to the goal (Fig.12.a). The obtained path allows the robot to avoid all obstacles (Fig.12.a and Fig.12.b) and arrive safely to the goal (Fig.12.c). The experiment proved the robustness of D* Lite algorithm in finding the shortest path in a non-deterministic underwater environment.

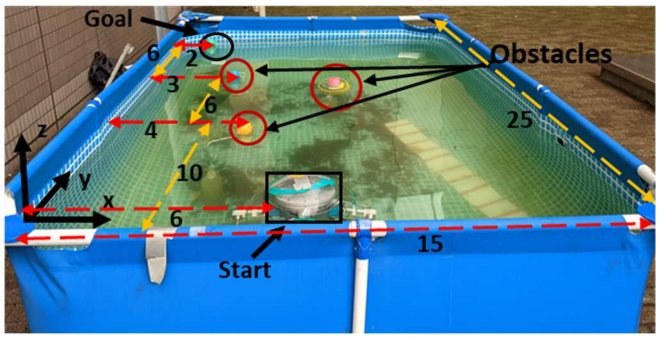
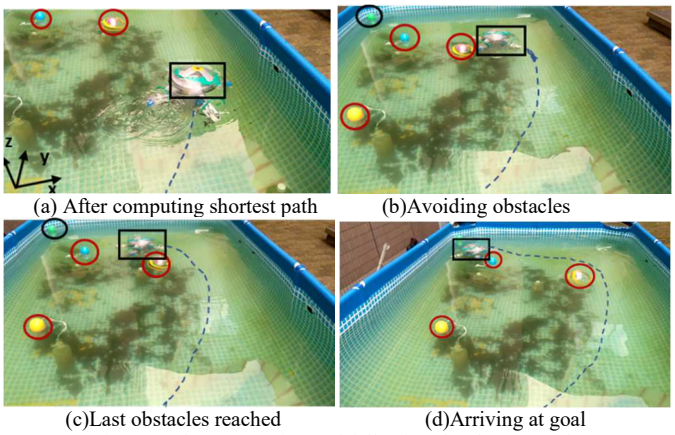


Fig. 11 Experimental setup.


(a) After computing shortest path (b)Avoiding obstacles

(c)Last obstacles reached (d)Arriving at goal

Fig. 12 Robot computing and following the shortest path.

VI. Conclusion And Future Work

This paper explores the problem of safe navigation in non-deterministic underwater environment for multiple robot's collaboration. A path planning strategy based on D* Lite search algorithm is examined to provide the optimal path to the goal. The safety of the algorithm is enhanced by considering the size of encountered obstacles as well as by maintaining a certain guard interval between the robot and the obstacles to avoid any collision. The collaborative navigation is achieved by designing an adaptative multi-robot's formation shape. The robots adjust the formation between a V-shape and line shape accordingly to the environment.

The collaborative navigation is realized with suitable robustness in the simulation. The robots can smoothly change their formation to increase the safety of the system. However, the formation control strategy presents some shortages in terms of keeping the appropriate distance and angle between the robots.

The experiment conducted in the water tank validates the use of D* Lite algorithm in finding the optimal path when obstacles are present in the environment. Also, the algorithm presents promising performances regardless of the size of obstacles. The simulation's results furthermore attest to the good accuracy of the planned path.

This work requires to be extended with further experiments to demonstrate the reliability of the formation control and the feasibility of the collaborative navigation. Also, a camera sensor should be considered for detecting the obstacles in order to solve the constraints induced by the Bluetooth communication. In the future, this research will be improved with real-word conditions of experiment to prove the effectiveness of the aforementioned strategies.

References

[1] Y. Guo, H. Liu, X. Fan, and W. Lyu, "Research Progress of Path Planning Methods for Autonomous Underwater Vehicle", Mathematical Problems in Engineering, vol. 2021, Article ID 8847863, 2021.

[2] V. Yordanova and B. Gips, "Coverage Path Planning with Track Spacing Adaptation for Autonomous Underwater Vehicles", in IEEE Robotics and Automation Letters, vol. 5, no. 3, pp. 4774-4780, July 2020.

[3] B.K. Patle, B. L Ganesh, A. Pandey, D.R.K. Parhi, and A. Jagadeesh, "A review: On path planning strategies for navigation of mobile robot", in Defence Technology, vol. 15, issue 4, pp 582-606, 2019.

[4] S. Koenig and M. Likhachev, ''Fast replanning for navigation in unknown terrain,'' IEEE Trans. Robot., vol. 21, no. 3, pp. 354–363, 2005.

[5] S. Koenig, C. Tovey, and Y. Smirnov, "Performance bounds for planning in unknown terrain", Artificial Intelligence, vol. 147, Issue 1–2, pp 253-279, 2003.

[6] A. T. Le, M. Q. Bui, T. D. Le, and N. Peter, "D* lite with reset: Improved version of D* lite for complex environment", in Proceeding of the 2017 IEEE International Conference on Robotic Computation (IRC), pp. 160–163, 2017.

[7] K. Xie, J. Qiang and H. Yang, "Research and Optimization of D-Start Lite Algorithm in Track Planning", IEEE Access, vol. 8, pp. 161920-161928, 2020.

[8] B. Sun and D. Zhu, "Three-dimensional D* lite path planning for autonomous underwater vehicle under partly unknown environment", in Proceeding of the 2016 IEEE world congress on Intelligent Control and Automation., pp. 3248–3252.

[9] J. Guo, C. Li, and S. Guo, "Path Optimization Method for the Spherical Underwater Robot in Unknown Environment", J Bionic Eng. 17, pp 944–958, 2020.

[10] L. Shi, K. Tang, S. Guo, X. Bao, S. Pan and P. Guo, "Leader-follower cooperative movement method for multiple amphibious spherical robots", in proceeding of 2016 IEEE International Conference on Mechatronics and Automation, pp. 593-598, 2016.

[11] L. Zheng, S. Guo, Y. Piao, S. Gu, and R. An, "Collaboration and Task Planning of Turtle-Inspired Multiple Amphibious Spherical Robots", Micromachines, vol. 11, no. 71, 2020.

[12] S. Guo, S. Cao and J. Guo, "Study on Collaborative Algorithm for a Spherical Multi-robot System based on Micro-blockchain", in Proceeding of 2019 IEEE International Conference on Mechatronics and Automation, pp. 1465-1470, 2019.

[13] X. Hou, S. Guo, L. Shi, H. Xing, et al, "Improved Model Predictive-Based Underwater Trajectory Tracking Control for the Biomimetic Spherical Robot under Constraints", Applied Science, vol.10, no.22,2020.

[14] J. Guo, C. Li and S. Guo, "Study on the Autonomous Multirobot Collaborative Control System Based on Spherical Amphibious Robots", IEEE Systems Journal, 2020.