IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 May 2023, accepted 21 June 2023, date of publication 26 June 2023, date of current version 30 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3289719

RESEARCH ARTICLE

Adapted D* Lite to Improve Guidance, Navigation and Control of a Tail-Actuated Underwater Vehicle in Unknown Environments

JUAN A. ALGARÍN-PINTO[1], LUIS E. GARZA-CASTAÑÓN[1], (Member, IEEE), ADRIANA VARGAS-MARTÍNEZ[1], AND LUIS I. MINCHALA-ÁVILA[2], (Senior Member, IEEE)

[1]School of Engineering and Sciences, Tecnologico de Monterrey, Monterrey, Nuevo León 64849, Mexico
[2]Department of Electrical Engineering, Electronics and Telecommunications, Universidad de Cuenca, Cuenca, Azuay 010101, Ecuador

Corresponding author: Luis E. Garza-Castañón (legarza@tec.mx)

This work was supported in part by Consejo Nacional de Humanidades Ciencia y Tecnología (CONAHCYT), Mexico; and in part by Tecnologico de Monterrey.

ABSTRACT Biomimetic Autonomous Underwater Vehicles (BAUVs) navigate aquatic environments by mimicking natural propellants from fish species. These vehicles move part(s) of their bodies using various mechanisms to propel and swim forward or laterally. Their main goal is to follow and adjust defined paths to reach a target autonomously. Local path planning is of paramount importance during navigation tasks due to unexpected obstacles. Moreover, path planning strategies should consider the environment's information obtained by the vehicle during its mission, as well as its dynamics and mechanical limitations, to define new routes properly. This article presents the development of a waypoint generator based on the D$*$ Lite algorithm. The proposed planner considers a frontal-short-sighted and tail-actuated BAUV with motion constraints to adjust the vehicle's path towards a target coordinate. By identifying obstacles, the planner adjusts and defines inner waypoints inside the vehicle's vision range by considering closeness to obstacles found and BAUV's current position. The developed strategy reduces collision risks due to the discrimination of nodes near obstacles, prioritizing broad hallways and safer swimming distances between the vehicle's current position and inner waypoints. The effectiveness of the proposed algorithm is simulated using the BAUV's hydrodynamics model and by adding a waypoint tracking controller to correct the vehicle's swimming performance inside three scenarios. The vehicle can reach the goal by properly defining inner waypoints while safely avoiding collisions, narrow hallways, and sharp turns.

INDEX TERMS Biomimetic autonomous underwater vehicle (BAUV), D$*$ Lite algorithm, path planning, path tracking, waypoint guidance systems.

I. INTRODUCTION

Biomimetic autonomous underwater vehicles are a novel alternative for developing missions inside aquatic environments. These vehicles are characterized for imitating fish's swimming locomotion; that is, by moving part(s) of the vehicle's structure, a BAUV can thrust itself as well as correct its position and orientation [1], [2], [3]. Furthermore, by mimicking natural systems, BAUVs may outperform classic underwater vehicles with turbine-based propellers. This is possible due to biomimetic shapes that allow higher levels of maneuverability and efficiency compared to the energy-expensive turbines employed in classic systems [4].

BAUVs present different shapes according to different swimming styles [5]. One of the most effective ways of swimming is by moving only the body's last section and a caudal fin (BCF locomotion) [6]. By moving their bodies and tails to generate a pressure difference in water, BCF swimmers propel themselves and achieve momentum to spin. BAUVs that follow BCF-thunniform locomotion only employ their

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Huei Cheng.

flapping tail to thrust and spin, representing low energy-cost alternatives for long navigation tasks.

A biomimetic vehicle should regulate its inner motion to attain an adequate swimming speed and turning moment to follow defined paths properly. Further, the swimming performance of a BAUV will mainly depend on its biomimetic features and hydrodynamics. Hence, turning moment and cruising speeds will be limited to the control applied to the vehicle's propulsion system and its mechanical design.

When path planning is applied, a collision-free path between an initial position and a destination is generated based on specific constraints and control conditions [7]. One way to start planning optimal paths is by building grid models. The navigation space is divided into cells (grids) or nodes with two possible states: free to visit or an obstacle. If the whole environment is known and static, a global path planner will find the optimal (shortest, safest, or low-cost) node tree between starting and final coordinates. Inside these scenarios, the A* algorithm is the most common methodology implemented due to its high efficiency in finding the shortest path inside the grid [8]. When navigating inside partially or fully unknown environments, local path planning techniques should be employed to adjust the vehicle's path due to unexpected obstacles [9]. For these cases, the D* (dynamic A*) search is commonly implemented [10]. This algorithm combines the original planning information with the online search to adjust paths in real-time once an obstacle has been detected or has moved. While A* tends to be computationally expensive with larger high-resolution grids [11], [12], D* tends to be memory expensive [8].

The D* Lite algorithm is an excellent alternative to the problems found in the algorithms mentioned above [13]. It presents a new replanning strategy between the goal and the vehicle's current position, changing the search direction compared to A* and D*. By fixing the target coordinate and by only considering a changing starting point through time, D* Lite increases the search efficiency attaining a fast-replanning task.

Local planners try to find the shortest and optimal path inside unknown scenarios; however, such paths are not necessarily the safest. Sometimes, following suggested paths might not be achievable due to the vehicle's non-holonomic constraints, or it might be too risky due to their proximity to obstacles. Furthermore, these algorithms consider that the robot knows the complete information about its surroundings and can reach either of its neighboring nodes. Inside aquatic environments, local planners should consider the vehicle's features, such as a short-sighted vision range, hydrodynamics, localization, orientation, closeness to obstacles, and other circumstances to generate the best (or adjust) reachable paths towards the goal [12], [14].

This article builds a waypoint path planner based on the principles employed in the D* Lite algorithm. The planner is responsible for selecting new nodes inside a grid based on the information obtained from a short-sighted BAUV with BCF-thunniform locomotion. By considering motion constraints, the planner introduces waypoints as inner targets for the vehicle to reach to achieve the final goal. For such purposes, the algorithm defines costs to discriminate free nodes based on their Euclidean distance to the vehicle, obstacles, and current seen space. By locally planning inner targets, the vehicle is driven using a waypoint path tracking controller to regulate its motion and swimming performance.

The D* Lite replanning strategy is compared to the proposed local path planner. Three simulated environments are presented to test the effectiveness and feasibility of the proposed algorithm. Such simulated experiments have proven optimization over navigation time, traveled distance, and expanded nodes and the generation of safer, collision-free navigation paths.

This article is divided as follows: Section II details the literature review for optimizing classic planners to attain safer paths. Section III presents the mechanical design, kinematics, and hydrodynamics of the tail-actuated BAUV. Section IV details the path-tracking control strategy employed to regulate the swimming performance of the vehicle. Section V develops the modified waypoint planner and details its main differences from D* Lite. Section VI details all attained results. Finally, all concluding remarks are stated in Section VII.

II. RELATED WORK

Finding optimal paths does not guarantee collision-free missions, especially when vehicles present unstable transitional phases between states. Hence, enhancing path-planning algorithms has become an important field of research [15]. Some studies have focused on smoothening the resulting path suggested by global planners. In [16], Imran et al. incorporated a probabilistic road map to iteratively produce achievable paths for underwater vehicles after implementing an A* algorithm. On [17], Ataei et al. employed a multi-objective GA to find paths that satisfied four criteria: traveled distance, a margin of safety, smoothness of the planar motion, and gradient of diving. Likewise, Aghababa [18] developed a global path planner that considered the underwater vehicle's dynamics and a nonlinear optimal control problem with time constraints to plan achievable paths. While optimizing classic planners such as A* or novel alternatives such as genetic and evolutionary algorithms helps build safer paths, their computational cost enlarges when employed for local planning inside unknown environments.

Hybrid A* planners consider the vehicle's kinematics and non-holonomic constraints for dynamically planning continuous motion inside grids [19]. By defining the current position and orientation, new paths are defined based on curves toward a new node. In [20], Van Dang et al. optimized a Hybrid A* local planner by implementing cost functions to new routes' curvatures to reduce collision risks and overcome poor path tracking control problems. On [21], Wang et al. developed an artificially guided Hybrid A* (AGHA*) to define kinematically-feasible paths for unmanned surface vehicles inside complex harbor environments. The algorithm combines human experience and the vehicle's full motion constraints to improve searching efficiency to generate a docking trajectory along with control variables for the task. Further, Hybrid A* may be employed to generate coarse paths by adapting conditions in the searching process. In [22], a search resampling optimization framework was developed to produce a series of safer corridors along coarse paths produced from a Hybrid A* planner.

Intelligent algorithms may also be employed to search for safer and achievable paths. In [23], authors developed an improved artificial potential field planner capable of working simultaneously with a model predictive controller at each decision step. By considering the vessel's motion constraints, such as angular velocity and short-sight perception, fast planning and path tracking was achieved for detouring obstacles inside unknown environments. Praczyk [24] developed a neural network that performs a dynamic selection of nodes based on object detection, current mapping, and waypoint-based path planning. Although optimal paths are not found, good approximations are attained to avoid collisions. Deep learning is applied for vision-based obstacle detection and avoidance in [25], where the safest spot inside the vehicle's local range is selected as the following node. NN are robust to highly nonlinear or poorly modeled systems; however, they tend to require large samples and training [26]. Even when these algorithms present good robustness when finding the safest path, they present a poor generalization performance and may present slow processing speed.

Finally, D* Lite algorithms have also been optimized for replanning inside complex environments. Le et al. [27] proposed the D* Lite with reset, where the algorithm discards old data and starts a new search when specific criteria are met, such as the ratio of traversed length and high complexity of the remaining path. In [28], Peng et al. improved D* Lite for replanning multi-robot paths. By setting safe distances between robots, the movement costs of the grids around each robot are computed to locally replan collision-free paths between vehicles. Xie et al. [29] optimized local planning by selecting priority levels of child nodes. This way, safer paths were obtained by discriminating nodes near obstacles.

Based on current literature, this article's main contribution describes the development of an enhanced guidance, navigation, and path control framework for a tail-actuated underwater vehicle. This is done by improving how the BAUV defines routes inside unknown environments when obstacles have been detected. To regulate its swimming performance, a waypoint guidance system and path tracking controllers are developed. Furthermore, the highly efficient D* Lite algorithm is adapted for fast and safe replanning. Then, by adapting the algorithm to the frontal short-sight feature of the vehicle and by implementing strategies such as discriminating nodes and planning longer inner paths, the vehicle is capable of reaching target coordinates while following collision-free, safer, and attainable paths.

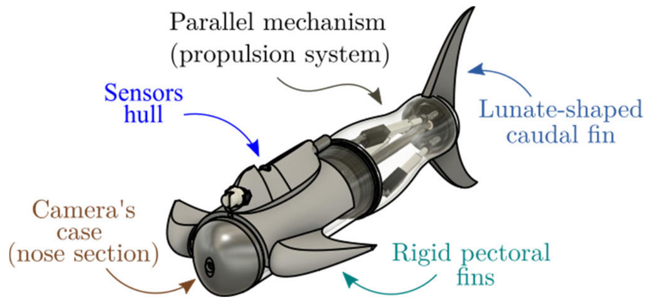

FIGURE 1. **BAUV designed for navigation tasks. Nose section houses a camera for obstacle detecting purposes. At the midbody section, two rigid pectoral fins are employed for stabilization. The main propeller is a lunate-shaped caudal fin that is driven by a parallel mechanism.**

III. BAUV MODELS AND PROPULSION SYSTEM DESIGN

Figure 1 shows the designed BAUV modeled for this study. The vehicle thrusts itself by moving a lunate-shaped caudal fin attached to its aft. Its hull presents three main sections. At the bow (nose section), a Myring shape was designed to reduce dragging [30]. The nose section houses a camera, the only vision system inside the vehicle for obstacle identification. A sensor case with rigid pectoral fins is implemented at the midbody section. Pectoral fins are only employed for stabilization purposes while swimming. The hull houses a parallel robotic mechanism in its last section, forming the BAUV's primary propulsion system. The vehicle presents sideways flapping, where the caudal fin is driven due to the oscillatory effect produced by the parallel mechanism.

A. PROPELLER'S KINEMATICS

The parallel mechanism is three degrees of freedom with three universal-cylindrical-universal and one spherical joint configuration linking two platforms (3-DOF 3-UCU-1S) [31]. The upper moving platform oscillates by regulating the displacement of three linear actuators (limbs), causing the flapping effect in BAUV's caudal fin. Due to the structural stiffness, stability, and significant level of support that the designed parallel system brings, the vehicle's tail may reach great position accuracy at high-frequency flapping. Furthermore, the propulsion system can produce a vectored thrust by biasing the upper platform's oscillations. Hence, by regulating the flapping performance of the vehicle's caudal fin, the BAUV can swim forward and turn towards specific goals.

Figure 2a shows the design of the 3 UCU-1S parallel robot incorporated inside the propulsion system. Three limbs are responsible for moving linearly, causing the moving platform to change its orientation. The motion from each limb is described by linear displacement d_i for each of the *ith* actuators. The two platforms are linked by their centers using a fourth restrictive limb to avoid translational motion. Then, the upper platform can only change its orientation once limbs

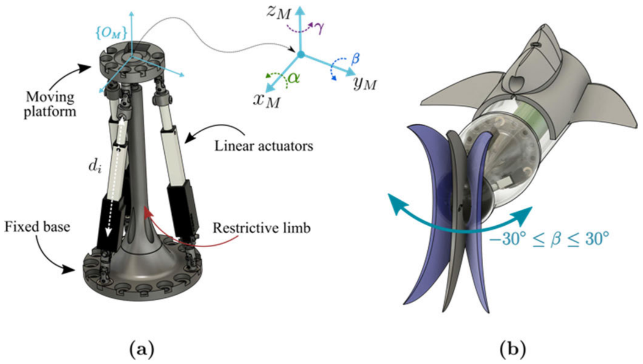

FIGURE 2. On (a), the 3 UCU-1S parallel mechanism employed inside the propulsion system is shown. By moving linear actuators, the upper platform oscillates, producing a flapping angle β. Motion attained from the lunate-shaped caudal fin due to motion of the moving platform is shown on (b).

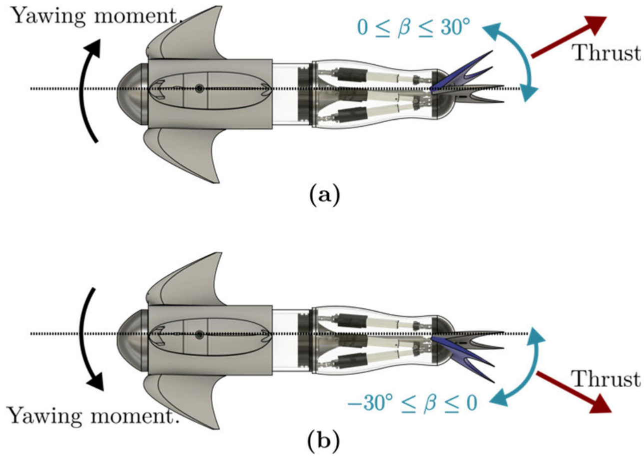

FIGURE 3. By biasing vehicle's caudal fin oscillatory motion, the BAUV starts turning. Expected flapping range β, yawing moment and thrust during full biased oscillations to (a) starboard and (b) port are shown.

start moving. Transferred motion to the caudal fin and its oscillatory effect is defined by β, representing the moving platform's final orientation.

The flapping effect on the caudal fin is induced by setting a function to the pitch angle β. This is done by approximating oscillations to a sine function. Hence, the caudal fin starts flapping by oscillating the upper platform over its y_M axis. The fin's oscillations will range by mechanical design to a maximum/minimum point of $\beta = \pm 30°$, as shown in Figure 2b. Based on its mechanical constraint, the caudal fin sideways motion presents a complete workspace of 60°. Then, the vehicle's tail presents an oscillatory flapping motion approximated by:

$$\beta\left(t\right) = Asin\left(2\pi ft + \rho\right) + b. \tag{1}$$

Thrust and moment are induced to BAUV according to how fast and how biased is the flapping performance of the caudal fin. The vehicle's tail motion is regulated by adjusting flapping parameters on (1). Furthermore, the rest of the BAUV's swimming motion is regulated by regulating the fin's motion.

Four parameters should be controlled during navigation tasks to correct the vehicle's speed and turning moment. On (1), A defines the flapping amplitude from the caudal fin, and its value depends on whether biased flapping is desired. Parameter b determines the level of bias induced to flapping. When b is a nonzero value, the vehicle's tail starts biasing to either port or starboard, producing a turning moment to the BAUV's centroid. When b is regulated, a vectored thrust is produced. Further, when combined adequately with parameter A, the platform is set to reach its mechanical maximum or minimum of $\pm 30°$. After each control period, the ρ shifting parameter is iteratively computed to set the platform to always start at zero position ($\beta = 0°$). Finally, the flapping frequency parameter f determines how fast the propeller oscillates. Thrust is expected to increase when the flapping frequency f starts increasing.

Figure 3 illustrates how vectored thrust is attained when bias is induced. The turning moment is expected to bias to either starboard or port once the parameter b increases or decreases (respectively). Since swimming performance becomes unstable with fast cruising speeds, a proper parameter-tuning regulator should be employed. By controlling the propeller's kinematics, an adequate navigation performance is attained. Parameters (b, f) are regulated by the designed path tracking controller, and their values range from $(-15, 15)°$ and $(3, 5)$ Hz, respectively. The illustrated scenario shows the expected flapping performance when b is set to 15° and $-15°$ (Fig. 3a, 3b, respectively).

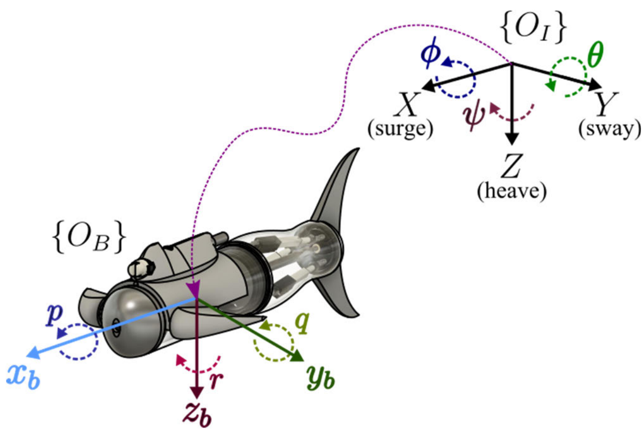

FIGURE 4. Earth-fixed $\{O_I\}$ and body-fixed $\{0_B\}$ coordinate systems employed to define vehicle's motion in space.

B. BAUV'S KINEMATIC AND DYNAMIC MODELS

Two reference frames are employed to track the vehicle's position and orientation in space: an inertial Earth-fixed frame (x, y, z) and a body-fixed frame (x_b, y_b, z_b). Figure 4 displays the frames employed for the kinematics analysis. The SNAME nomenclature employed for underwater motion is also shown [32].

Body-fixed frame's origin $\{O_B\}$ is fixed on BAUV's center of buoyancy. BAUV's position (x, y, z) and orientation

(ϕ, θ, ψ) are referenced to Earth-fixed frame $\{O_I\}$ by position vector $\boldsymbol{\eta}$. For BAUV's six-degrees-of-freedom, (u, v, w) represent translational velocities, and (p, q, r), represent angular velocities measured along (x_b, y_b, z_b). Forces and moments exerted on the vehicle are defined by $\boldsymbol{\tau}$. Velocities $\boldsymbol{v}$ and forces $\boldsymbol{\tau}$ are all referenced to $\{O_B\}$. BAUV's spatial position, velocities, forces, and moments exerted by the propeller are then defined by:

$$\boldsymbol{\eta} = (x, y, z, \phi, \theta, \psi)^T \tag{2}$$
$$\boldsymbol{v} = (u, v, w, p, q, r)^T \tag{3}$$
$$\boldsymbol{\tau} = (X, Y, Z, K, M, N)^T . \tag{4}$$

Linear and angular velocities are measured and referenced to the BAUV's own body, and a change of coordinates should be computed. Transformation matrices are used to convert velocities $\boldsymbol{v}$ to velocities $\dot{\eta}$ respect to the Earth-fixed frame. By incorporating transformation matrices $\mathbf{J_1}\left(\boldsymbol{\eta_1}\right)$ and $\mathbf{J_2}\left(\boldsymbol{\eta_2}\right)$, change of coordinates is obtained:

$$\dot{\boldsymbol{\eta}}_{\mathbf{1}} = (\dot{x}, \dot{y}, \dot{z})^T = \mathbf{J_1}\left(\boldsymbol{\eta_1}\right)[u, v, w]^T \tag{5}$$
$$\dot{\boldsymbol{\eta}}_{\mathbf{2}} = \left(\dot{\phi}, \dot{\theta}, \dot{\psi}\right)^T = \mathbf{J_2}\left(\boldsymbol{\eta_2}\right)[p, q, r]^T \tag{6}$$

where $\dot{\boldsymbol{\eta}} = \left[\dot{\boldsymbol{n}}_{\mathbf{1}}, \dot{\boldsymbol{\eta}}_{\mathbf{2}}\right]$. Transformation matrices are obtained by implementing the Euler angles convention for roll, pitch, and yaw and are defined as:

$$\mathbf{J_1}\left(\boldsymbol{\eta_1}\right) = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{7}$$

$$\mathbf{J_2}\left(\boldsymbol{\eta_2}\right) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix}. \tag{8}$$

Short notations $c\blacksquare$, $s\blacksquare$, and $t\blacksquare$ inside transformation matrices (7) and (8) stand for cosine, sine, and tangent of the respective rotation angle.

The dynamic model of the vehicle was derived on [33], where hydrodynamic coefficients such as added mass and dragging terms were duly computed. To such analysis, the vehicle was assumed to be a rigid body of constant mass and buoyancy through all missions. Also, symmetry over (x, y) and (x, z) planes were assumed for the study. Then, the BAUV's dynamic model is described by the Newton-Euler equations of motion:

$$\boldsymbol{M}\dot{v} + \mathbf{C}\left(\boldsymbol{v}\right)\boldsymbol{v} + \mathbf{D}\left(\boldsymbol{v}\right)\boldsymbol{v} + \boldsymbol{g}\left(\boldsymbol{\eta}\right) = \boldsymbol{\tau}. \tag{9}$$

From the left-hand side of (9), $\boldsymbol{M} \in \mathbb{R}^{6\times6}$ is the added mass matrix, $\mathbf{C}\left(\boldsymbol{v}\right) \in \mathbb{R}^{6\times6}$ is the added mass centripetal forces and Coriolis matrix, $\mathbf{D}\left(\boldsymbol{v}\right) \in \mathbb{R}^{6\times6}$ is the hydrodynamic damping matrix, and $\boldsymbol{g}\left(\boldsymbol{\eta}\right) \in \mathbb{R}^{6\times1}$ is the restoring vector for BAUV's hydrostatic forces. From the right-hand side of (9), $\boldsymbol{\tau} \in \mathbb{R}^{6\times1}$ is the input vector of forces and moments produced by the motion of the vehicle's caudal fin and its orientation. For brevity purposes, the full form of matrices can be found on Appendix A. Table 1 shows physical and hydrodynamic parameters used for swimming simulation of the designed BAUV during navigation tasks.

TABLE 1. BAUV physical and hydrodynamic parameters.

Parameter	Symbol	Value	Units
Axial length	l	0.64	m
Fin to fin width	pw	0.34	m
Mass	m	4.00	kg
Weight	W	39.24	N
Buoyancy	B	44.54	N
Center of gravity	(x_g, y_g, z_g)	(0,0,0.05)	m
Center of buoyancy	(x_b, y_b, z_b)	(0,0,0)	m
Inertia moments	(I_{xx}, I_{yy}, I_{zz})	(0.005,0.085,0.085)	$kg \cdot m^2$
Added mass terms	$X_{\dot{u}}$	−0.2239	kg
	$Y_{\dot{v}}$	−8.0675	kg
	$Y_{\dot{r}}$	1.1598	kg · m/rad
	$Z_{\dot{w}}$	−8.0625	kg
	$Z_{\dot{q}}$	−1.1598	kg · m/rad
	$K_{\dot{p}}$	−0.0096	$kg \cdot m^2/rad$
	$M_{\dot{w}}$	−1.1598	kg · m
	$M_{\dot{q}}$	−0.4971	$kg \cdot m^2/rad$
	$N_{\dot{v}}$	1.1598	kg · m
	$N_{\dot{r}}$	−0.4971	$kg \cdot m^2/rad$
Damping terms	$X_{u\lvert u\rvert}$	−0.5055	kg /m
	$Y_{v\lvert v\rvert}$	−4.4950	kg /m
	$Z_{w\lvert w\rvert}$	−4.4950	kg /m
	$K_{p\lvert p\rvert}$	−0.1300	$kg \cdot m^2/rad^2$
	$M_{q\lvert q\rvert}$	−0.3920	$kg \cdot m^2/rad^2$
	$N_{r\lvert r\rvert}$	−0.3920	$kg \cdot m^2/rad^2$

IV. PATH TRACKING CONTROL STRATEGY

A. LINE OF SIGHT GUIDANCE SYSTEM

On our designed vehicle, thrust and moment are generated by water displacement resulting from the motion of the vehicle's platform and caudal fin. The propulsion system applies forces through each linking limb to make the propeller flap at a certain amplitude, bias, and frequency [31]. This means that the three linear actuators serve as a transmission system that exerts forces through flapping, pushing the vehicle forward and laterally [33].

BAUVs mainly depend on their propellers' biomimetic features to regulate swimming. Hence, these systems should be smart enough to understand their mechanical limitations to excel in their performance. By providing these vehicles with guidance systems, it is possible to track information from the cruising execution of the robotic fish.

Guidance systems are used to give the smart vehicle information regarding its position and attitude in space respecting to a final goal. A guidance system may compute a path-tracking error by tracking an arriving coordinate and the vehicle's forward velocity and orientation. Based on the

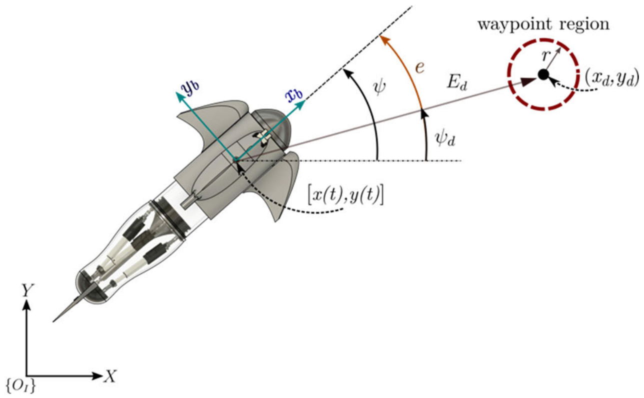

FIGURE 5. Line of sight strategy implemented for the BAUV's guidance system. Vehicle's heading angle deviation (e) and Euclidean distance to waypoint (E_d) are iteratively computed to track navigation performance.

error's behavior, corrections could be induced in the swimming performance of the vehicle in order to reduce it.

In the waypoint tracking strategy, the vehicle should reach a preestablished set of coordinates in space (waypoints). The vehicle should swim toward a waypoint by keeping a nonzero forward velocity and regulate its heading direction by correcting the turning moment on its body. Once the vehicle has reached the preestablished coordinate, some others may be introduced to form a desired path.

A Line of Sight (LOS) strategy is employed to measure the path tracking error (vehicle's heading deviation towards the goal). In the LOS algorithm, coordinate transformations are used where the desired and actual positions of the vehicle are iteratively compared to reduce their distance and to compute a desired orientation.

Figure 5 depicts the behavior of the implemented LOS system. In this strategy, a waypoint is set as the primary goal. The guidance system implements a LOS for the vehicle and computes how deviated (e) it is from an ideal path and how far the vehicle is from the goal (E_d). Each goal coordinate (x_d, y_d) is commonly surrounded by a waypoint region. Once the vehicle has reached such region ($E_d \leq r$), it may try to reach the following coordinate or finish the mission.

The main objective is always to keep the vehicle's LOS aligned while approaching the goal (by reducing both e and E_d near zero). Euclidean distance (E_d) from the vehicle's position to the goal and the heading angle's deviation (e) are defined as:

$$E_d = \sqrt{(x_d - x)^2 + (y_d - y)^2}\ (\text{m}) \tag{10}$$

and,

$$e = \begin{cases} \psi_d - \psi, & -\pi < \psi_d - \psi < \pi \\ \psi_d - \psi - 2\pi, & \psi_d - \psi \geq \pi \\ \psi_d - \psi + 2\pi, & \psi_d - \psi \leq -\pi \end{cases} \quad (\text{rad}) \tag{11}$$

where:

$$\psi_d = atan2\left((y_d - y), (x_d - x)\right)(\text{rad}). \tag{12}$$

From (12), $atan2(\blacksquare Y, \blacksquare X)$ is the two-argument arctangent and 2π variant of $\tan^{-1}(\blacksquare X/\blacksquare Y)$. BAUV's desired heading direction then ranges between $-\pi \leq \psi_d \leq \pi$. According to desired and actual orientation, bias flapping should be induced for turning moment generation.

B. WAYPOINT TRACKING CONTROLLER

Two parameters are controlled throughout the navigation task: fin's flapping frequency f and bias b. Such parameters are regulated according to information brought by the guidance system. A path-tracking controller is implemented to regulate the flapping parameters of BAUV's caudal fin. Then, two control phases are implemented, a speed regulator and an attitude controller.

Figure 6 shows the closed-loop path tracking control diagram employed inside simulations. A path planner is responsible for generating waypoints in space. The waypoint guidance system will compute error parameters while the vehicle is swimming. Based on the Euclidean distance to an inner waypoint E_d, a desired forward speed $\dot{x}_d$ is defined for the BAUV. Furthermore, based on the vehicle's deviation error (e) towards the inner waypoint, a bias is induced in the fin's flapping.

Figure 7a shows the P-PD controller implemented for BAUV's speed regulation. Each control step considers the inner waypoint, not the vehicle's final goal. In its first stage, the desired forward velocity $\dot{x}_d$ is defined proportionally to the vehicle's distance to the generated waypoint. This is done strategically since the vehicle should reduce its velocity to attain a more stable swimming performance while approaching the waypoint. This helps to avoid missing the inner goal due to high cruising speed. It also allows the vehicle to swim faster once a new waypoint is induced to reduce navigation time. In its second stage, $\dot{x}_d$ is compared to the vehicle's actual speed $\dot{x}$, and speed error $e_{\dot{x}}$ is computed. The PD is responsible for computing an increment of frequency Δf that is then added to the fin's actual flapping frequency f. Increments of frequency are saturated and may only range between $-2 \leq \Delta f \leq 2$ Hz. Then, the speed control law to define the tail's flapping frequency is determined as follows:

$$f = f + K_{P,j} \cdot e_{\dot{x}} + K_{V,j} \cdot \dot{e}_{\dot{x}} \tag{13}$$

where:

$$e_{\dot{x}} = \dot{x}_d - \dot{x} \tag{14}$$

$$\dot{x}_d = K_{P,i} \cdot E_d. \tag{15}$$

Figure 7b shows the incorporation of the PD controller for bias definition. For the attitude control stage, heading deviation error e is tracked to compute b. The bias parameter is then defined as:

$$b = K_{P,k} \cdot e + K_{V,k} \cdot \dot{e}. \tag{16}$$

Flapping and bias regulation are introduced to the fin position control stage after each control period of $1/f$ seconds. Table 2 shows the proportional and derivative gains used for

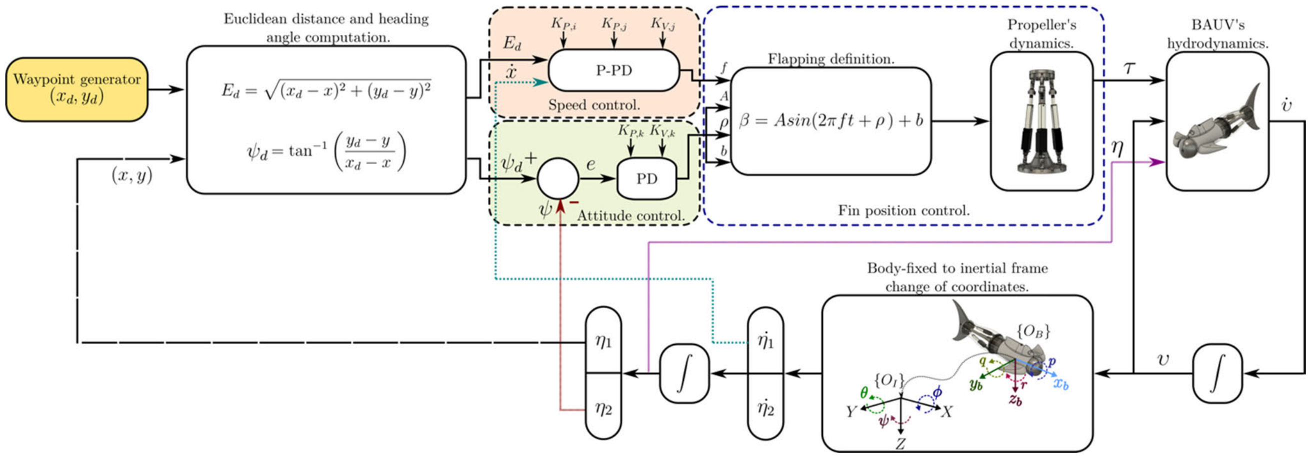

FIGURE 6. Waypoint tracking controller designed for the development of the parameter-tuning task. The controller defines BAUV's fin flapping frequency, bias, amplitude, and phase to correct its course towards inner waypoints. A P-PD speed and a PD attitude controllers are responsible for the tuning task.

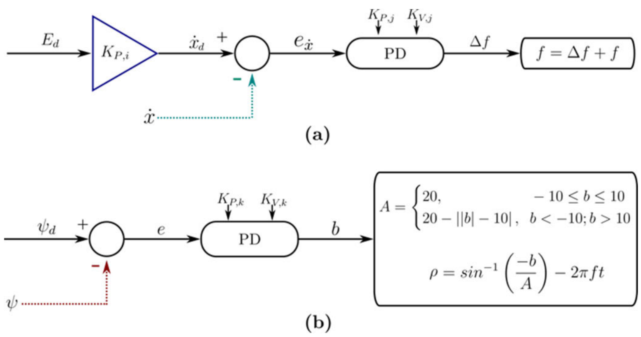

FIGURE 7. On (a), the P-PD control block diagram to define desired forward speed and regulate flapping frequency is shown. On (b), the PD control stage for attitude regulation based on BAUV's heading error is detailed.

TABLE 2. Control gains for speed and attitude regulation.

Stage	Gain	Value
Speed P-PD Controller	$K_{P,i}$	0.15
	$K_{P,j}$	2
	$K_{V,j}$	0.10
Attitude PD Controller	$K_{P,k}$	5
	$K_{V,k}$	$2\sqrt{5}$

the speed and attitude control stages. Once the new values for frequency and bias are defined, flapping amplitude and phase-shifting parameters should also be adequately defined. When the induced flapping bias is slight ($-10^\circ \le b \le 10^\circ$), the propeller is set to flap at a full range of 40°. When the bias is significant, the mechanical constraints will reduce such workspace until the fin only flaps at a complete range of 30° (when $b = -15^\circ$ or $b = 15^\circ$). A defines the flapping workspace. Finally, the phase-shifting parameter ρ is strategically computed after each control period to start flapping at zero position with $\beta = 0$. Then, amplitude and shifting parameters (A, ρ) are defined as follows:

$$A = \begin{cases} 20^\circ, & -10^\circ \le b \le 10^\circ \\ 20^\circ - \left||b| - 10^\circ\right|, & b > 10^\circ; b < -10^\circ \end{cases} \tag{17}$$

$$\rho = \sin^{-1}\left(\frac{-b}{A}\right) - 2\pi ft. \tag{18}$$

The vehicle is then set to reach inner waypoints toward a final goal. To achieve them, the BAUV should swim using its propulsion system. The waypoint tracking controller's primary goal is to combine flapping frequency and bias on the caudal fin appropriately. By tracking information from the proposed guidance system and applying regulation stages for the flapping tail, the vehicle produces a vectored thrust and turning moment to regulate its course. The correct development of the parameter-tuning task then enhances the BAUV's swimming efficiency and overall performance while navigating.

V. LOCAL PATH PLANNING DEVELOPMENT

To visualize the feasibility of both a D* Lite local planner and the proposed modified version, three different environments shown in Figure 8 are tested. Simulations are driven over a rectangular swimming space of 12 by 19 meters, ranging from $(-6, 6)$ width over the x-axis and $(-1, 18)$ height over the y-axis, respectively. Furthermore, the field is mapped into a grid of nodes over the (x, y) plane with a resolution of 0.5 m. The inertial Earth-fixed frame origin $\{O_I\}$ is positioned at $(0, 0, 0)$ coordinate. For all scenarios, the BAUV's starting position $\boldsymbol{\eta}$ was arbitrarily set to $(1, 0, 0, 0, 0, 90^\circ)$, and the final goal was set to $(1, 15.5, 0)$ measured with respect to $\{O_I\}$. Hence, the vehicle is faced toward the goal at the beginning of each simulation. The vehicle should swim inside the simulated free space towards the goal and correct its course based on the presence of obstacles.

From Figure 8, all nodes inside red areas are considered obstacle-forbidden nodes that the BAUV should detect to cor-

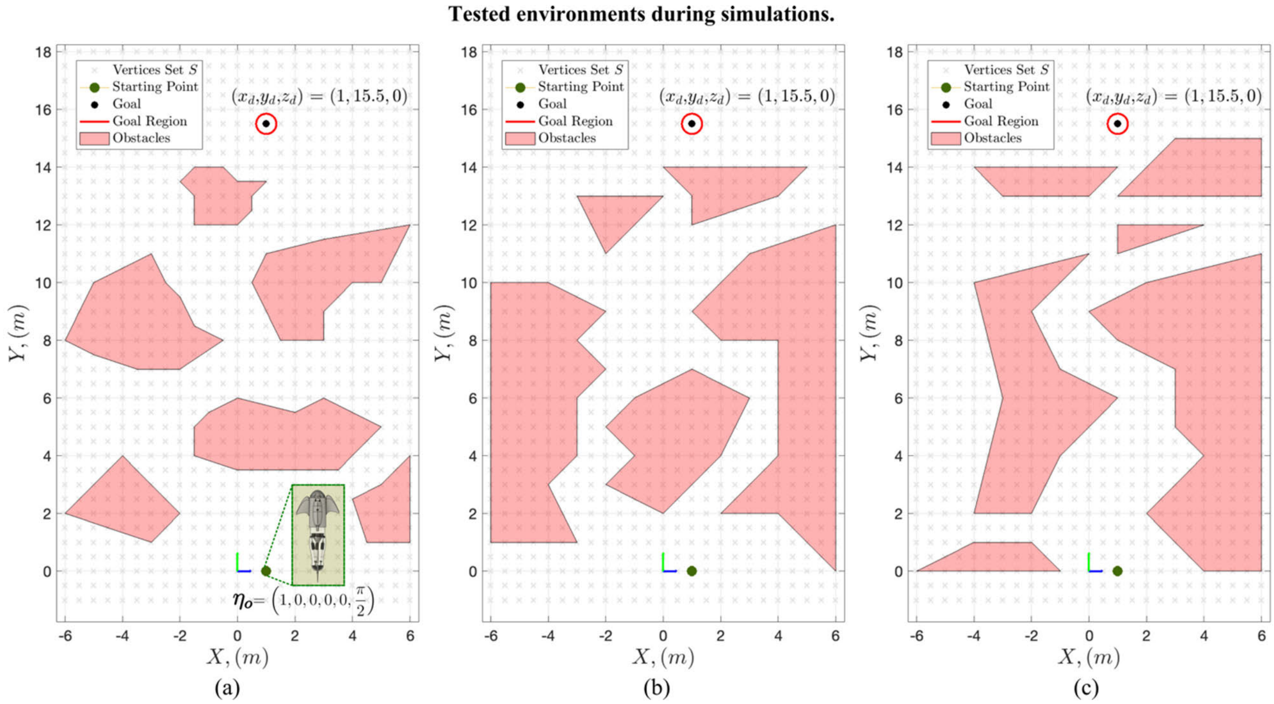

FIGURE 8. Three different scenarios are employed to compare the standard implementation of D* Lite and our adapted version when achieving a safe navigation task. While on (a) the environment was designed to allow the vehicle swim in wider free spaces, (b) presents narrow halls where planning correctly becomes of paramount relevance. Finally, (c) combines both broad spaces with narrow halls.

rect its course. Gray nodes inside the grid are free coordinates that the local planner may select as inner waypoints. Further, all obstacle areas are considered static and will not move during simulated missions. Also, all simulations consider the BAUV's hydrodynamics and the waypoint tracking strategy to tune flapping parameters in the vehicle. Hence, the local planner and BAUV's swimming performances are tested during navigation to visualize the cruising efficiency while trying to reach the final goal.

A. D* LITE ALGORITHM

D* Lite algorithm searches for an optimal path that connects a final fixed goal node and the vehicle's position as a starting node. Once an obstacle inside the optimized path has been detected, D* Lite reuses all previous information and updates it to find a new path. To do so, the planner performs an incremental search by considering all the unknown environment as a free space to recalculate. Since the vehicle is moving towards the goal, the algorithm also considers a moving starting node.

When computing the shortest path, the algorithm computes node-to-node edge costs just as an A* planner (but in the opposite direction). Considering that each node $s \in S$ is a vertex inside the set of possible nodes in the grid map, the g-value $g(s)$ and the one-step lookahead rhs-value $rhs(s)$ are computed to define the optimal path between s_{start} and s_{final}. Both values are cost estimates of the distance between a starting node to a vertex s. The rhs-value of a node $rhs(s)$ is computed based on the traversed cost function $g(s')$ values of its successors in the graph $s' \in Succ(s)$, and the transition costs to reach each successor $0 < c(s, s') < \infty$. Since the search starts from s_{final}, initial distance and cost are set to 0.

The computation of the rhs-value is of great importance since the algorithm uses its minimum as the searching direction to build the path toward the goal. The rhs-value is iteratively computed, and the search expands from node s to all eight possible node's successors s' in the grid by finding the minimum $c(s, s') + g(s')$ until the goal s_{final} is found [13]. $rhs(s)$ is then computed as:

$$rhs(s) = \begin{cases} 0, & s = s_{final} \\ \min\limits_{s' \in Succ(s)} \left(c(s, s') + g(s')\right), & \text{otherwise.} \end{cases} \tag{19}$$

Furthermore, D* Lite uses a priority queue (open list) for expanding nodes when searching for the shortest path. When the g and rhs values of a node are the same, the node is considered consistent. When a series of consistent nodes between the start and goal is found, the shortest path may be obtained. The rhs-value of a node may change once an obstacle is detected, changing the node to an inconsistent state. All inconsistent nodes enter the priority queue for further processing. All nodes inside the open list are ranked based on a key priority $key(s)$ that dictates the expanding direction the algorithm should take when finding the new path. Each node's key priority is computed by considering a heuristic function $h(s, s')$ that measures the distance of the optimal path between the current node and the starting point. The priority key of node s is a vector formed by the minimum of its g and rhs value plus the focusing heuristic and a second tie-breaking key. Then, $key(s)$ is defined as:

$$key(s) = [\min(g(s), rhs(s)) + h(s_{start}, s); \min(g(s), rhs(s))]. \tag{20}$$

For the current implementation, all edge costs and heuristic functions for distances between vertices are computed using the Euclidean distance. When finding the shortest path, the algorithm expands the node with minimum cost inside the open list and updates the g and rhs values of all its surrounding vertices. Then, all nodes that became either under-consistent or over-consistent are processed to update their values and expand such changes to their successors until local consistency is attained. After updating rhs values, the algorithm removes all consistent nodes or adds new inconsistent nodes to the open list. This procedure is iteratively done to elements in the queue until the queue's next top key to expand is no less than the s_{start} key or until s_{start} is locally consistent. Then, the new shortest path from the vehicle's current position and goal is obtained.

Table 3 presents the pseudocode for the implementation of D* Lite, as presented in [13]. In the initialization procedure $\{02'\} - \{06'\}$, g and rhs values of all nodes in the grid are set to infinite. The target node s_{final} rhs-value is set to 0 and changes its state to inconsistent, becoming the first entree of the open list. The target node is permanently fixed throughout the whole task. A first computation for the shortest path is made based on the environment's information, vehicle's starting position s_{start}, and s_{final} $\{10'\} - \{20'\}, \{23'\}$. Typically, the first path estimation does not consider any obstacle since they are outside the vehicle's vision range. Once the shortest path is computed, the vehicle starts moving. The vehicle moves from s_{start} to any of its 8-adjacent neighbor nodes with the smallest estimated values, and its new position becomes the new s_{start} $\{26'\} - \{27'\}$. After reaching a new position, the vehicle should check for updates inside its environs. If an obstacle inside the defined path is found, the cost of edges between the current position and the observed node must be updated accordingly $\{28'\} - \{33'\}$. Transition costs to obstacles are generally set to infinity. Based on the cost change, the rhs-value of the found obstacle must be updated, as well as its key value $\{7'\} - \{9'\}, \{34'\}$. After the found node and its surroundings have been updated, a new computation of the shortest path between the target node and s_{start} must be conducted considering the updated queue's key priorities. The vehicle can then follow the recommended path. This procedure is followed until s_{start} becomes s_{final}.

Figure 9 shows the BAUV performance after implementing a D* Lite path planner inside each simulated environment. Each vertex defined by the planner was considered a waypoint and entered into the path-tracking control strategy detailed in the previous section. For the obstacle detection stage, a surrounding vision range of 1.5 meters from the

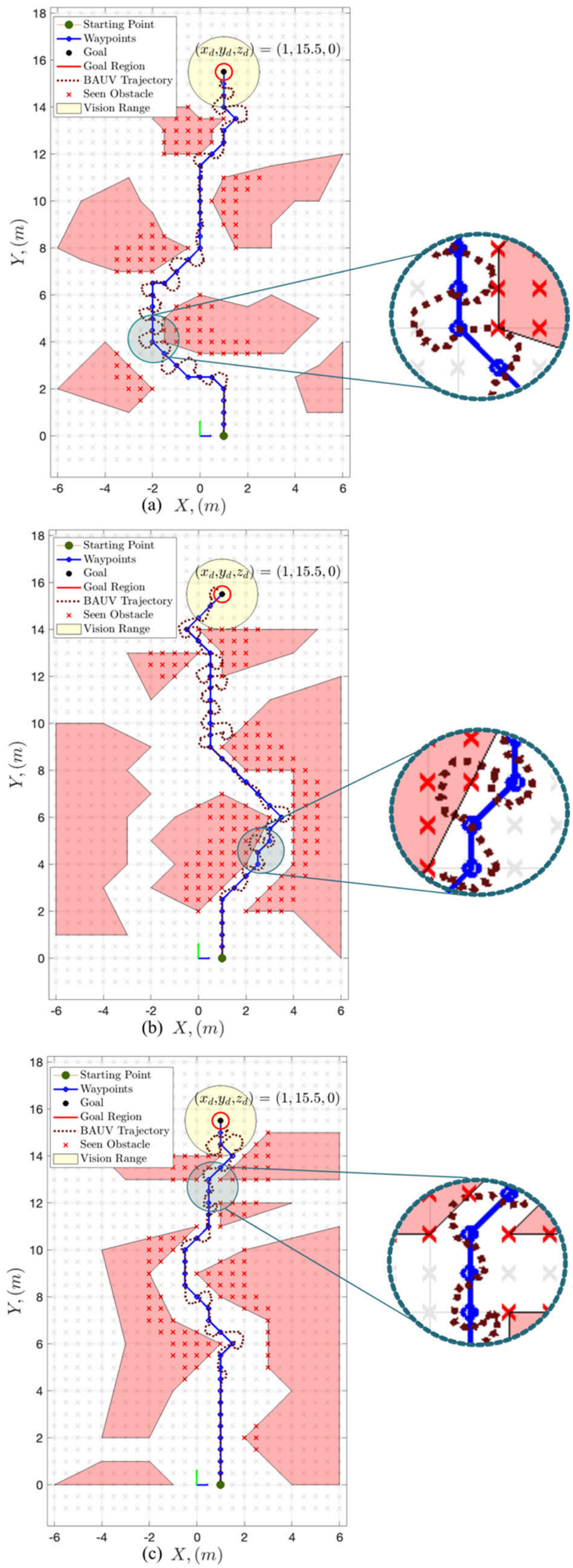

FIGURE 9. Standard D* Lite path planner implemented. On (a), the planner does not consider broader free spaces for replanning. On (b) and (c), the vehicle passes too close to obstacles inside hallways. For all scenarios, the D* Lite planner sets the BAUV to follow risky routes. Some expected collisions are shown inside each navigation task.

vehicle's center of buoyancy is considered. Hence, the vehicle corrects its flapping performance to swim between its current position and the inner waypoint regions (set to $r = 0.20\ m$).

When the standard D* Lite planner is employed to define the navigating path of the BAUV, adjacent waypoints are introduced toward the goal. Then, the waypoint path tracker focuses on reaching each waypoint region, causing the vehicle to present sharp corrections while swimming. The vehicle presents an undesirable swimming performance where even when the path defined by the local planner is obstacles-free, the BAUV collides. Furthermore, making (almost sequentially) sharp turning corrections while swimming results in larger traveled distances and larger inefficient missions in the long run.

In the simulated environment from Figure 9a, broad spaces are not fully considered to plan the route, yet the suggested path passes near all obstacles. The waypoint-based path even describes the shapes of the found obstacles. This phenomenon is visualized in scenarios from Figure 9b and 9c, where the vehicle swims near obstacles due to the defined routes. Furthermore, the standard path planner projects waypoints inside narrow halls since the shortest path is assumed to pass through them (as shown on 9c).

Although the BAUV can swim forward and describe straight paths when oriented toward a waypoint (as seen at the beginning of each simulation), changing its direction requires a transitional swimming stage. In other words, if a new waypoint requires the vehicle to change its heading direction, the BAUV will take some time to reach the required turning moment to achieve such orientation. Because of its hydrodynamics and swimming behavior, if the vehicle is near an obstacle while changing its orientation, it will be expected for the BAUV to collide (as shown in several stages in Figure 9).

B. PROPOSED ADAPTATIONS OVER D* LITE

Although the D* Lite algorithm finds the shortest obstacle-free path toward the goal, the BAUV cannot follow all waypoints smoothly. Then, some adaptations should be introduced to attain the safe navigation required for the BAUV. Some considerations based on the swimming performance of our BAUV should be stated to find such adaptations:

- The BAUV cannot immediately change its orientation. Because of its hydrodynamics and its main propeller's kinematics, the BAUV presents a transitional stage to reach a new heading direction. Then, when a new waypoint is generated, and the vehicle must correct its cruising orientation, a curved path towards such an inner goal is expected. Rough changes on BAUV's ψ_d inside short spaces will result in aggressive described paths. Hence, planning over short distances is not helpful for efficient swimming performances.
- The BAUV only attains a forward thrust. In other words, when the vehicle's propeller starts flapping, the vehicle moves forward. Based on the propeller's flapping bias, a turning moment is generated. Then the planner should prioritize forward paths towards the goal even when the path tracker adjusts BAUV's flapping to make it spin, keep the same orientation, accelerate, or deaccelerate.
- The BAUV presents a limited vision range. The vehicle presents a camera in its bow section with a defined vision range. Since D* Lite uses all the free space to plan, any of BAUV's eight adjacent coordinates may be selected as the best candidate to visit next. However, prior information about the presence of obstacles in the vehicle's surroundings should be attained to select any of such nodes. Hence, detecting all surrounding obstacles is not guaranteed unless the vehicle has previously seen them while swimming. Nodes inside BAUV's visibility should be prioritized over any unseen node when picking the new s_{start}. Then, what the vehicle visualizes when choosing a new node should be considered the safest and most conservative option to avoid collisions.

Based on these constraints, it is desired to adapt a planner that suggests a more robust path so that the vehicle can travel smoothly. The advantages of the fast-planning D* Lite algorithm should be adjusted for such purposes. The main incorporations focus on how the path could be safer once an obstacle has been found. By implementing a strategy for discriminating near-to-obstacles vertices, a conservative selection of waypoints may be attained to avoid collisions.

Moreover, considering all the observed information when picking a new vertex allows planning further than considering only eight adjacent vertices. Then, safer paths may be obtained by adding a conservative node selector to the fast-replanning D* Lite algorithm.

1) DEFINITION OF WARNING NODES

Once the vehicle has reached a vertex, an inspection task is performed to update information from the grid map. If the vehicle has found that the next expected node is an obstacle, a cost correction towards such node should be updated. D* Lite then expands that information to the successive nodes until a local consistency is found. Then, the algorithm suggests a new shortest path based on the computation of the g and rhs values. If the vehicle can visualize further than just the next node and finds an obstacle on its way, replanning is computed as well.

The shortest path generally avoids obstacles by selecting nodes that pass near the found obstacles. After finding a forbidden vertex, the cost update task is modified to avoid planning new routes near obstacles. The definition of warning nodes to those vertices near the obstacle is then employed to find safer paths.

Figure 10 shows a grid diagram where node **O** is a found obstacle during navigation. As specified in the standard development of D* Lite, all directed edge costs from vertices to the observed obstacle will be set to $c(\mathbf{O}, v) = \infty$. However, all surrounding nodes that are not considered obstacles are free options for replanning. In most cases, the planner will

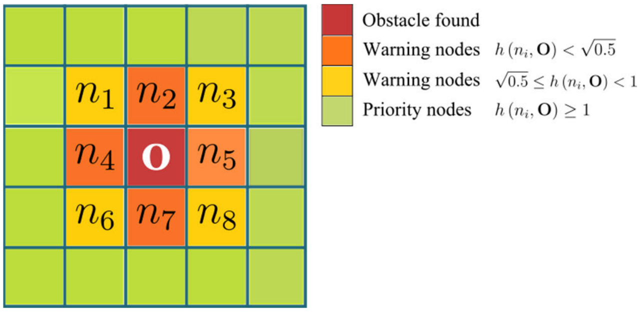

FIGURE 10. Definition of warning nodes after obstacle O is found. A weighted discrimination over adjacent nodes n_i is done based on their closeness to the forbidden vertex. Then, vertices that are far from O are priority for replanning. Nonetheless, warning nodes may be selected in the absence of broader free spaces.

employ such neighboring nodes to avoid passing through the obstacle and head back toward the goal. Nonetheless, all the adjacent vertices' costs should also be modified to avoid a new path passing near **O**. A free vertex may be classified as a warning node according to its Euclidean distance from the detected obstacle. Based on arbitrary thresholds and the heuristics employed to define distances to the obstacle, vertices are considered free or warning nodes while replanning. When a vertex is classified as a warning node, some weights may be added to the traveling costs to such nodes, and their information may be expanded to their local successors. Figure 10 illustrates how warning nodes may be classified based on their closeness to **O**.

By defining a threshold value *thr* for the distance of each adjacent node n_i to **O**, a weight w is added to the already expected edge cost $c(s, n_i)$ as:

$$c(s, n_i) = \begin{cases} w + h(s, n_i), & h(n_i, \mathbf{O}) \leq thr \\ h(s, n_i), & \text{otherwise.} \end{cases} \tag{21}$$

In (21), s represents the vertex of the vehicle's current position (x, y), n_i represents the *ith* vertex from all adjacent nodes to obstacle **O** with coordinates (x_{ni}, y_{ni}), $(x_{\mathbf{O}}, y_{\mathbf{O}})$ respectively, $h(s, n_i)$ is the distance between vehicle's position and the *ith* adjacent node, and $h(n_i, \mathbf{O})$ is the distance between adjacent nodes to the detected obstacle. All computed distances are Euclidean.

Adding weights to traveling costs toward warning nodes helps prioritize the selection of vertices far from the detected obstacle. Then, the searching direction would emphasize such further nodes because of their lower costs when finding a new path. However, if there are no better options, the planner would select the best between all weighted warning vertices. Then, even when the algorithm tries to find the new shortest path to the goal, it would prioritize nodes far from obstacles until the projected path reaches the goal. By adapting the algorithm from Table 3, warning nodes' costs are updated after an obstacle has been found. Table 4 shows this modification inside procedure Main() from D* Lite.

Figures 11, 12, and 13 compare the adapted D* Lite planner that considers warning nodes to the standard version when

TABLE 3. Standard D* lite algorithm.

Procedure CalculateKey(s)
{01′} return $[\min(g(s), rhs(s)) + h(s_{start}, s) + km;\ \min(g(s), rhs(s))]$
Procedure Initialize()
{02′} $U = \emptyset$;
{03′} $km = 0$;
{04′} for all $s \in S$ $rhs(s) = g(s) = \infty$;
{05′} $rhs(s_{final}) = 0$;
{06′} $U.Insert(s_{final}, CalculateKey(s_{final}))$;
Procedure UpdateVertex(s)
{07′} if $(s \neq s_{final})$ $rhs(s) = \min_{s' \in Succ(s)}(c(s, s') + g(s'))$;
{08′} if $(s \in U)$ $U.Remove(s)$;
{09′} if $(g(s) \neq rhs(s))$ $U.Insert(s, CalculateKey(s))$;
Procedure ComputeShortestPath()
{10′} while $U.TopKey() < CalculateKey(s_{start})$ || $rhs(s_{start}) \neq g(s_{start})$
{11′} $k_{old} = U.TopKey()$;
{12′} $s = U.Pop()$;
{13′} if $(k_{old} < CalculateKey(s))$
{14′} $U.Insert(s, CalculateKey(s))$;
{15′} else if $(g(s) > rhs(s))$
{16′} $g(s) = rhs(s)$;
{17′} for all $u \in Pred(s)$ $UpdateVertex(u)$;
{18′} else
{19′} $g(s) = \infty$;
{20′} for all $u \in Pred(s) \cup \{s\}$ $UpdateVertex(u)$;
Procedure Main()
{21′} $s_{last} = s_{start}$;
{22′} Initialize ();
{23′} ComputeShortestPath();
{24′} while $(s_{start} \neq s_{final})$
{25′} /* if $(g(s_{start}) = \infty)$ then there is no known path */
{26′} $s_{start} = \arg\min_{s' \in Succ(s_{start})} (c(s_{start}, s') + g(s'))$;
{27′} Move to s_{start};
{28′} Check for changed edge costs;
{29′} if any edge costs changed;
{30′} $km = km + h(s_{last}, s_{start})$;
{31′} $s_{last} = s_{start}$;
{32′} for all directed edges (s, v) with changed edge costs
{33′} Update edge costs $c(s, v)$;
{34′} UpdateVertex(s);
{35′} ComputeShortestPath()

implemented in each environment. For these cases, only local path planning stages are shown. All previous characteristics were kept to visualize the performance of the modified algorithm, and the vehicle was assumed to have a surround vision range of 1.5 m.

The adapted algorithm replans routes after an obstacle is found but follows the warning node discrimination when possible. The discriminating threshold value was set to $thr = \sqrt{0.5}$, and the added weight for nodes was set to $w = 10$. This algorithm suggests new paths that do not pass near obstacles but prioritize broad free spaces. The final paths are not necessarily the shortest but are safer than those attained from the standard planner.

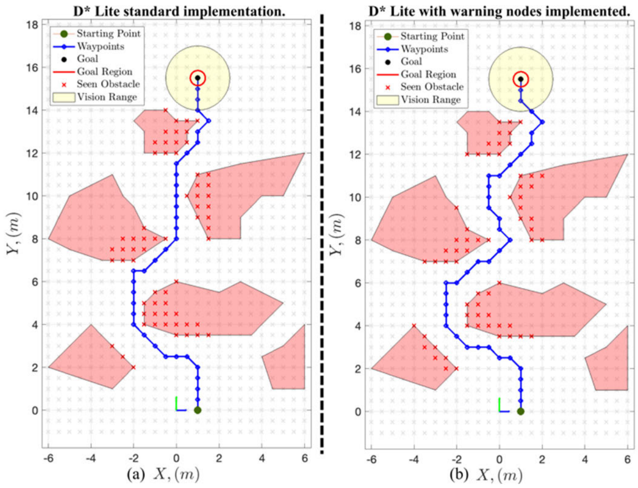

FIGURE 11. Comparison between (a) standard D* Lite and (b) D* Lite considering warning nodes. Because of the presence of wider spaces, the final path from (b) suggests a waypoint-based path that is at least one vertex away from obstacles.

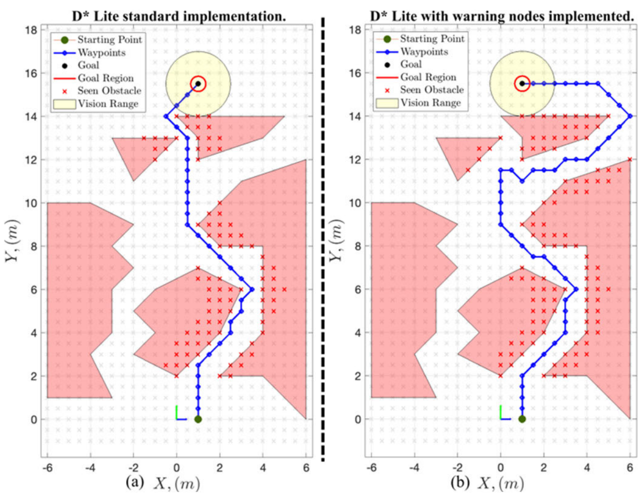

FIGURE 12. Comparison between (a) standard D* Lite and (b) D* Lite considering warning nodes. The adapted algorithm (b) allows planning by keeping safe distances to obstacles and by taking conservative decisions.

Figure 11a shows the path suggested by a standard D* Lite. On 11b, the adapted algorithm shows how near-to-obstacles nodes were discriminated when replanning. It is observed how free spaces were better employed to project a waypoint-based path. Even when the planner adjusted a path with sudden turns, all computed waypoints were at least one vertex away from a detected forbidden node. That was possible because of the environment's vast space.

Figure 12 shows the comparison between algorithms inside the second environment. Unlike the first scenario, the second environment presents halls that the vehicle must traverse to get to the goal. While on 12a, the planner projected a path traversing two hallways, on 12b, the adapted planner only passed through one. In this example, the adapted planner projected a path that passed through the middle of the first

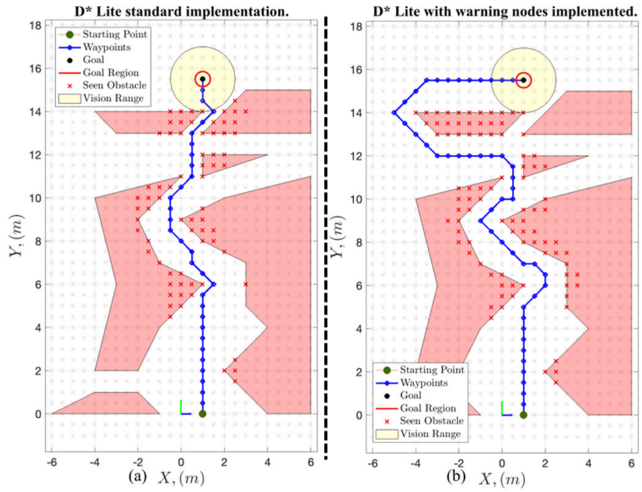

FIGURE 13. On (a), the path is near to obstacles even when wider spaces may have been employed. On (b), the whole path kept a safe distance from obstacles and even surrounded the last object to avoid passing through narrow halls.

hallway. When no wider spaces were able for replanning, the adapted D* Lite picked the less costly warning nodes. Furthermore, a conservative decision was taken in the last section of the navigation task. Since the adapted algorithm prioritizes open spaces over narrow hallways, the planner suggested that going around obstacles would have been better than traversing them.

Figure 13 shows the advantages of planning paths away from obstacles and avoiding traversing narrow hallways. Figure 13a shows the projected path that D* Lite computed during navigation. The suggested path made corrections near obstacles, and the shortest path to the goal was found by traversing halls between obstacles. On 13b, the waypoints projected a line right through the middle of the first two obstacles. In the last section, the conservative decision of avoiding narrow halls set the path around the obstacles, keeping a safe distance between the suggested path and obstacles. Then, the definition of warning nodes as an addition to the standard algorithm will allow finding safer paths than the near-to-obstacles shortest paths suggested by the standard D* Lite.

2) PLANNING LONGER INNER PATHS

Even when adding warning nodes may allow replanning safer paths, defining waypoints too close to each other represents a problem for the BAUV while swimming. The waypoint path tracker iteratively computes corrections over the vehicle's flapping tail to drive the BAUV toward each waypoint. Then, the vehicle's swimming effort to reach each inner waypoint ends up in larger traveled distances. The local planner could project waypoints inside the vehicle's vision range to avoid this. That means that the planner could consider the vehicle's adjacent coordinates and all nodes that the BAUV is visualizing to define a new inner waypoint. Then, by combining the selection of warning nodes with a strategy of picking waypoints away from BAUV's current position, safer paths

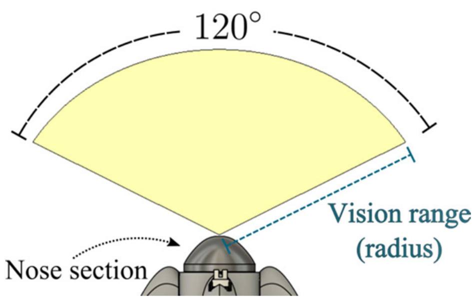

FIGURE 14. Representation of BAUV's perception. The vehicle's vision range is tested for different radii, and its peripheral vision is assumed to be of $120°$ at most.

may be planned, and a smoother swimming performance from the vehicle may be attained.

The BAUV's vision range plays a considerable role when replanning. According to obstacles found, warning nodes are defined, and a safe path is computed with the proper discrimination over vertices. When implementing a standard D* Lite, the vehicle's next position could be any of its eight neighboring coordinates. It is then assumed that the vehicle could reach any such nodes and update information from its surroundings. These assumptions were made when implementing the algorithms inside Figures 9, 11, 12, and 13.

Compared to surround sensor systems, the BAUV counts with a camera incorporated inside its nose section. The camera is considered the only sensor that visualizes the vehicle's environment. That means our BAUV's perception is limited to a frontal vision range, as shown in Figure 14. Moreover, the vehicle's vision range is assumed to be mid-peripheral up to 120° in diameter. These aspects directly affect how the vehicle detects obstacles and should be considered when planning the vehicle's next position. Hence, the vehicle should prioritize swimming forward and toward known and seen areas to attain safe navigation. Based on the information attained from the vision system, a path planner should properly define which nodes are safer to visit. It should be done considering the swimming performance of the vehicle and its safety when passing near detected obstacles.

Based on the limited perception of the environment, the planner should prioritize safer and more conservative paths rather than the shortest. Furthermore, planning over seen and known areas becomes the safe choice based on the assumption that the BAUV is swimming in a static environment (where detected objects will not move). Then, by prioritizing forward planning inside the BAUV's vision range, the path tracker control system will mainly focus on thrusting the vehicle toward the inner waypoint. Moreover, if such a waypoint is far from the vehicle's position, the control stage eases off efforts since corrections may be gradually induced while the vehicle moves. This strategy then focuses on reducing abrupt motions during navigation tasks. Then, more efficient swimming may be attained to follow more conservative, smoother, and safer paths.

On the standard implementation of D* Lite (as established in Table 3), s_{start} is chosen among the vehicle's eight adjacent vertices (line $\{26'\}$). Based on the sum of the travel cost that takes getting to such nodes and their heuristic distances to the goal, the neighboring coordinate with the minimum cost is selected as the new s_{start}. That ends up producing a sequential waypoint-based path formed by adjacent vertices.

The strategy then consists of taking the node information computed by the D* Lite algorithm and defining which of the seen vertices is the best option to visit next. The algorithm is then adapted right at the moment of selecting the new waypoint. The selection priority is set to all nodes that are visualized from the BAUV; that is, all vertices that fall inside the defined vision range (illustrated as a yellow area in Figure 14) must be considered first when choosing the new s_{start}. The next waypoint will be selected by considering the following aspects:

- The planner should always try to pick a waypoint in front of the BAUV and inside its vision range. This will allow keeping the forward thrust and direction of the vehicle while avoiding abrupt changes in its orientation. Then, at first, only those nodes inside the assumed range will be considered candidates. If and only if all seen nodes are obstacles or inaccessible, then the planner should retrieve.
- When retrieving, only the vehicle's adjacent nodes should be considered to avoid significant regression. Then, the best new waypoint selection will be followed as done on the standard D* Lite algorithm. Once the vehicle reaches such retrieving waypoint, the new search will prioritize seen vertices again.
- If there are seen reachable nodes, warning nodes must be identified first. All seen nodes should have a travel cost that measures the vehicle's distance to each possible vertex. All detected warning nodes' travel costs should be updated according to (21).
- Once all costs from seen nodes have been locally updated, three tie-breaking criteria for selecting the next waypoint are considered. First, each seen node's g-value $g\left(s'\right)$ is considered the primary tiebreaker. Second, the Euclidean distance between each candidate and the final goal $h\left(s', s_{final}\right)$ is employed. These two criteria force the planner to pick a new waypoint far from the BAUV's position and closer to the final goal. Then all nodes that are near the BAUV are locally discriminated. Finally, the third tiebreaker is the computed change of direction $\Delta\psi_{s'_i}$ required to reach each seen node considering BAUV's current orientation ψ, where:

$$\Delta\psi_{s'_i} = \left|\psi_{s'_i} - \psi\right| \tag{22}$$

and,

$$\psi_{s'_i} = atan2\left(\left(y_{s_i} - y\right), \left(x_{s_i} - x\right)\right). \tag{23}$$

Figure 15 illustrates how the prioritization of picking a new waypoint is done. Based on the previously stated con-

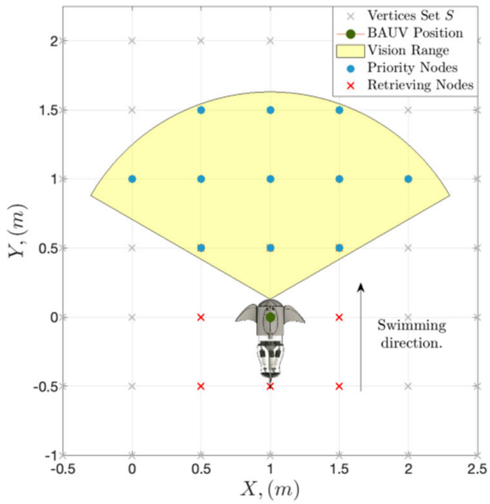

FIGURE 15. Flowchart of the main algorithm from adapted D* Lite implemented for local planning during simulations.

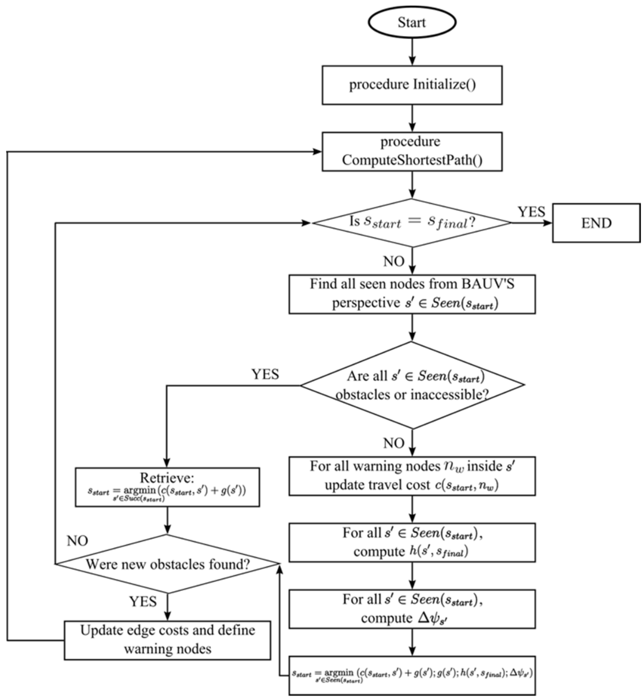

FIGURE 16. Strategy for selecting new s_{start}. All nodes that fall inside a preestablished range are considered as candidates. Based on the information provided by the vehicle's perception, discrimination over such nodes is done. If seen (priority) nodes are inaccessible, the vehicle retrieves by picking one of its surrounding vertices.

siderations, the planner will try to pick further nodes rather than those near the BAUV. Since all vertices are considered as known, the weight assignation may also be computed to help select the safest choice. Based on the proposed criteria, the new coordinate will be inside the vehicle's current vision range. The tracker should focus only on keeping a forward swimming direction and make minor heading corrections. For the selection of the next waypoint when not all seen nodes are obstacles or inaccessible, traveling costs, distance to the goal and change of orientation are all considered, and the new s_{start} will then be defined as:

$$s_{start} = \underset{s' \in seen(s_{start})}{\operatorname{argmin}} \left[\left(c\left(s_{start}, s'\right) + g\left(s'\right) \right); g\left(s'\right); h\left(s', s_{final}\right); \Delta\psi_{s'} \right]. \tag{24}$$

A small retrieving is induced when the vehicle cannot achieve any of the seen nodes. Any adjacent nodes surrounding the vehicle may be selected based on their heuristic values. Then, inside the retrieving stage, the new waypoint will be defined as:

$$s_{start} = \underset{s' \in Succ(s_{start})}{\operatorname{argmin}} \left(c\left(s_{start}, s'\right) + g\left(s'\right) \right). \tag{25}$$

Figure 16 shows the flowchart diagram for selecting a new waypoint based on the proposed strategy. The incorporation of such an algorithm replaces the standard selection of s_{start} as developed on $\{26'\}$ in Tables 3 and 4. This new waypoint-selection strategy will allow planning longer inner paths when possible.

VI. RESULTS

The adapted D* Lite was tested inside the three environments shown in Figure 8. The vehicle's vision range was set to different radii to show how limited vision ranges affect navigation performance and planning. Furthermore, the peripheral vision range was constant for all cases and set to 120°. The experiments consisted of positioning the vehicle at starting point η facing toward the final goal. To regulate swimming, the waypoint path tracker controlled the flapping performance of the BAUV's caudal fin for thrust and moment generation. Then, based on the definition of waypoints by the D* Lite local planner, the vehicle was set to reach all of them until the final coordinate was attained.

Figure 17 shows the results of four missions inside the first environment. The standard D* Lite algorithm was employed to plan paths for missions inside Figure 17a,b, and our adapted algorithm was implemented inside missions from Figure 17c,d. The BAUV's vision radius was set to either 1.5 m (Figure 17a,c) or 2.5 m (Figure 17b,d). For all missions, the limited vision range was set to 120°.

As seen on 17a, the vehicle did not end the mission due to a straight collision in the first obstacle. That occurred because the vehicle did not visualize the obstacle when planning one waypoint before. The D* Lite planner considered such an unseen node a free space for replanning, leading the BAUV to a collision. The main disadvantage of not having a surround vision system is that the standard algorithm expands its search toward all unseen nodes, and misinformed decisions could be made. The algorithm stopped once the BAUV reached the final node since it discovered that the defined vertex was an obstacle.

The BAUV could reach the goal coordinate once its vision range was enlarged to 2.5 m and the standard D* Lite planner

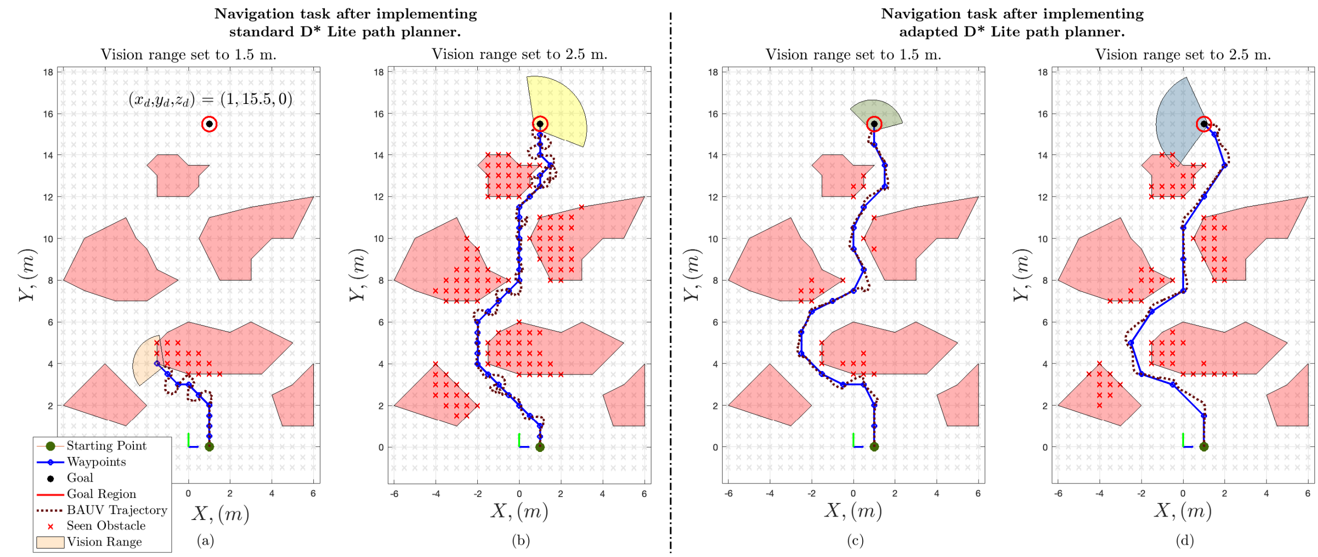

FIGURE 17. Results of missions inside first simulated environment. On (a) and (b), the standard D* Lite path planner was employed for the BAUV with a vision range of 1.5 m and 2.5 m, respectively. On (c) and (d), our adapted D* Lite algorithm was implemented for the BAUV with a vision range of 1.5 m and 2.5 m, respectively. For the first two cases, the vehicle did not finish the mission or presented long traveled distances. For the last two scenarios, the BAUV reached the goal and followed attainable paths.

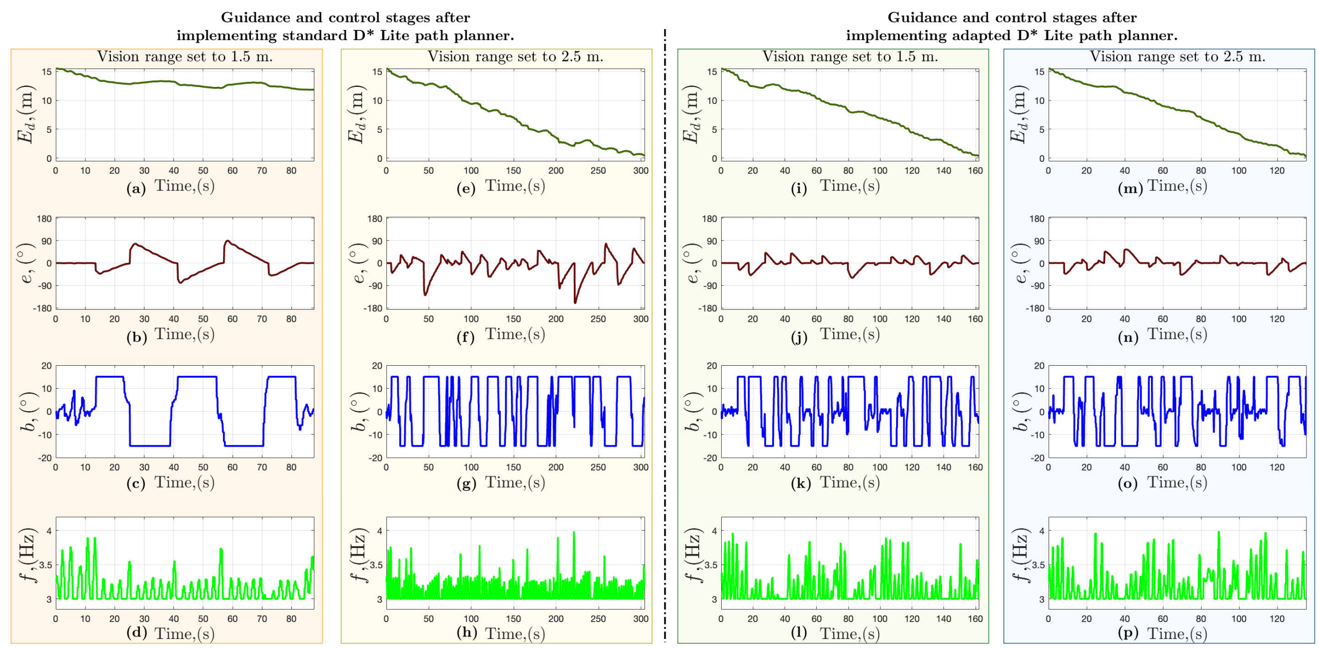

FIGURE 18. Guidance and control stages for simulated missions inside the first environment when the standard (a through h) and adapted (i through p) path planners were incorporated. BAUV's Euclidean distance to goal E_d and heading error deviation e were iteratively tracked to generate control regulations over tail flapping frequency f and bias b.

was still employed. In this case, the vehicle's perception of its environment was enhanced, and more forbidden vertices could be found. Then, the planner was allowed to find a route toward the goal without misclassifying an obstacle node as a free vertex. However, having a non-surround, enlarged, limited vision system does not guarantee that the vehicle will finish its mission.

Furthermore, although the navigation task was drastically enhanced once the vision range was enlarged, the standard D* Lite planner did not help to improve the vehicle's swimming performance. For both cases, in Figure 17a,b, the waypoints induced were sequentially placed, and that caused the BAUV to describe an erratic swimming performance. The implemented control strategy sets the vehicle to reach

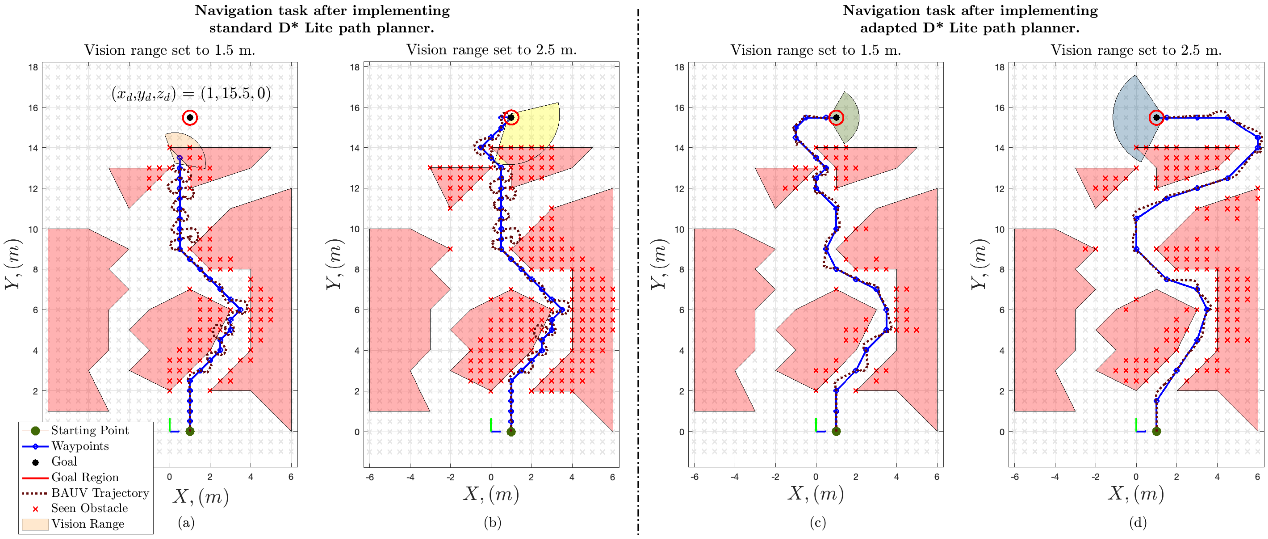

FIGURE 19. Results of missions inside the second simulated environment. On (a), the standard planner misclassified a forbidden vertex due to the short-sight vision range, and the mission could not be finished. On (b), the vision range was enlarged to 2.5 m and the BAUV reached the goal. However, the standard planner suggested a path that produced a poor swimming performance. On (c) and (d), the adapted D* Lite planner was employed, and the vehicle followed smoother routes by keeping a safe distance from obstacles. Conservative paths were attained with a larger vision range of 2.5 m.

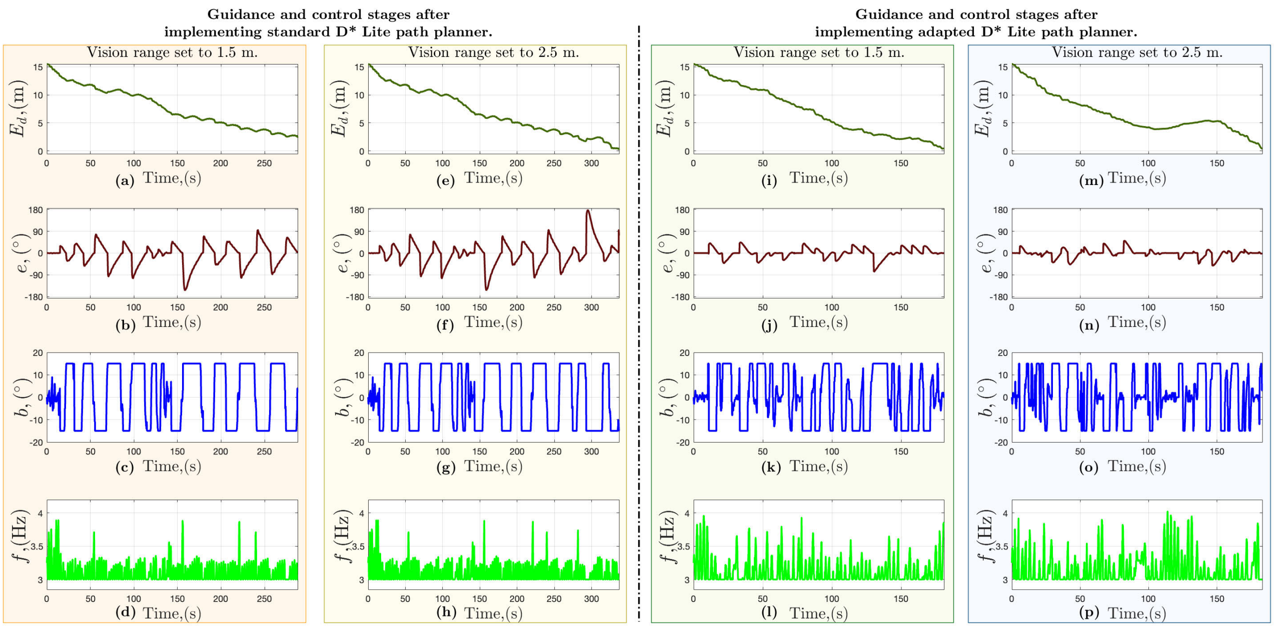

FIGURE 20. Guidance and control stages for missions inside the second environment with: (a through h) the standard and (i through p) our adapted path planners. The adapted D* Lite allowed to reduce distance to goal faster than the standard algorithm, as well as it allowed a better regulation over the vehicle's heading deviation. Control stages were enhanced by allowing the gradual correction over the flapping performance of the BAUV's tail.

all waypoints, inducing abrupt corrections while swimming. Because of such performance, collisions and larger traveled distances are expected. When implementing a standard D* Lite path planner, counting with a surround perception system enlarges the possibility of finishing the mission and reaching the goal. Nonetheless, the standard algorithm ended up suggesting unhelpful paths.

On the other hand, Figure 17c,d shows the navigation results after implementing the proposed adapted D* Lite for path planning. For both cases, the algorithm followed the warning node identification strategy and planned longer inner paths. Then, the waypoint-based computed path presented larger spaces between nodes. While swimming, each waypoint was selected based on

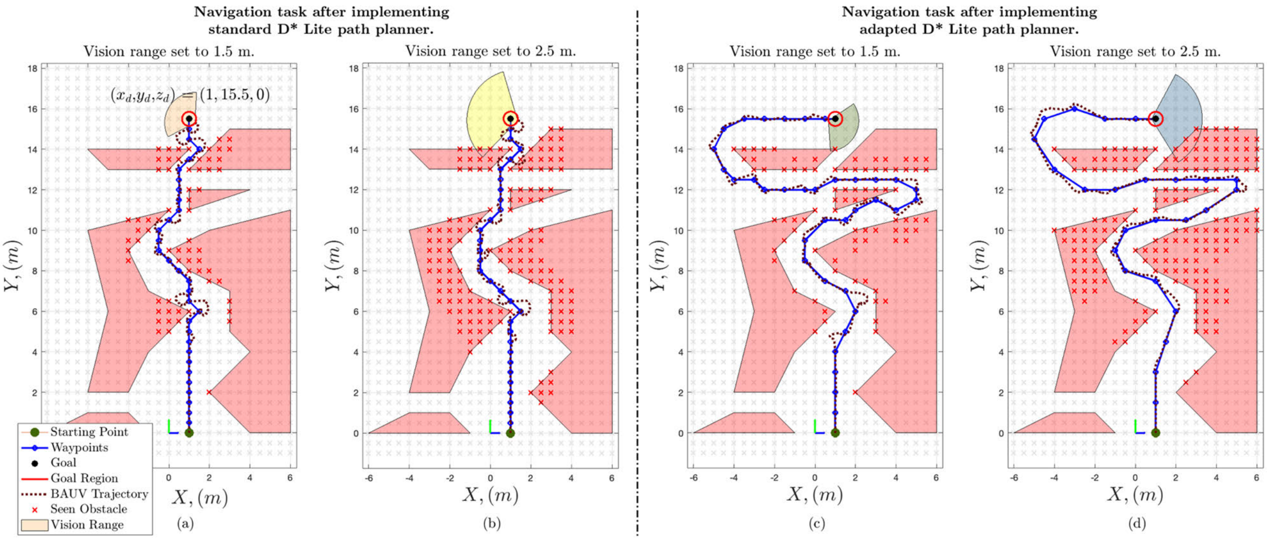

FIGURE 21. Results of missions inside the third simulated environment. On (a) and (b), the standard planner suggested a difficult path for the BAUV conditions. Even when the vehicle was able to reach the goal, collisions are expected due to the poor swimming performance attained. On (c) and (d), our adapted planner suggested a more achievable and safer path. For both cases, the vehicle was capable of reaching the goal by following conservative paths and by keeping a safe distance from obstacles.

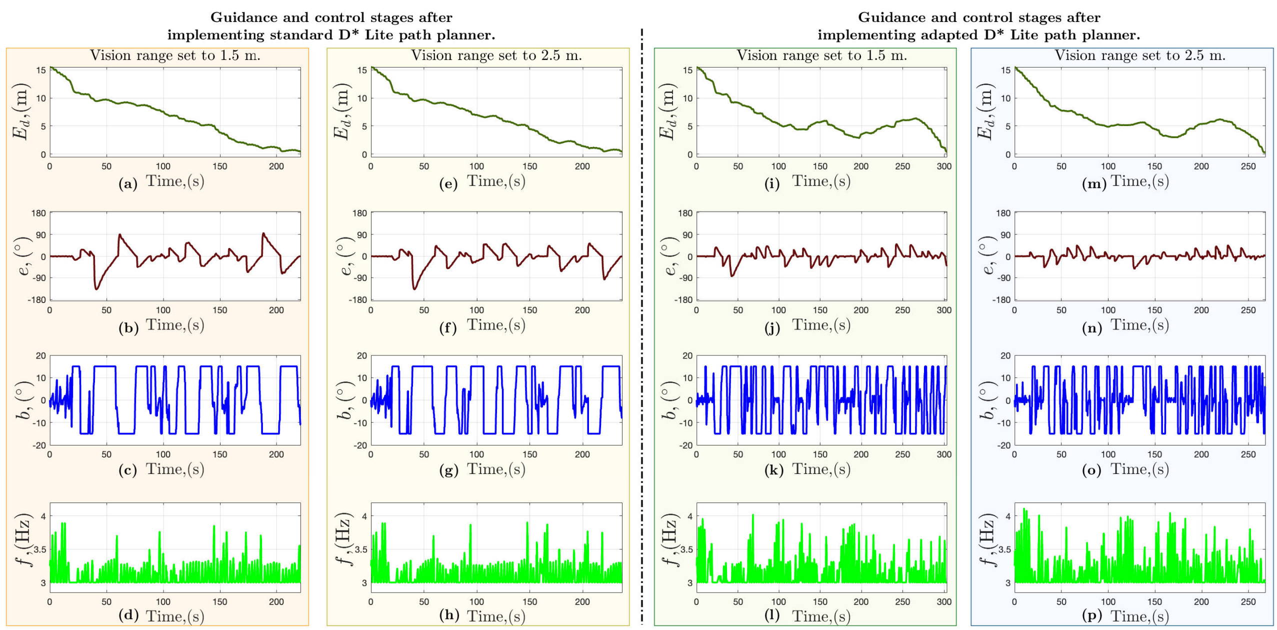

FIGURE 22. Guidance and control stages for missions inside the third environment with: (a through h) the standard and (i through p) our adapted path planners. Even when the adapted algorithm caused longer mission periods due to conservative paths, the vehicle's heading deviation error was largely reduced and handled. This allowed the reduction of the erratic swimming behavior due to gradual control over the tail's flapping performance.

the vehicle's current perception, finishing on a smoother, simpler-to-follow, and safer path. The vehicle reduced its erratic behavior because of larger planning spaces between nodes.

Compared to 17c, results were even enhanced inside the mission in Figure 17d, when the BAUV's vision range was enlarged to 2.5 m, and our adapted planner was employed. The final waypoint-based planned path allowed the vehicle to follow almost straight paths toward the goal. Also, the safe distance between the vehicle's swimming path and obstacles was kept throughout the mission. For the last two cases, the BAUV described a smooth path toward the goal without any risk of collision. All induced waypoints were attainable and properly planned based on the vehicle's current position and heading direction, which allowed performing better-informed decision-making tasks.

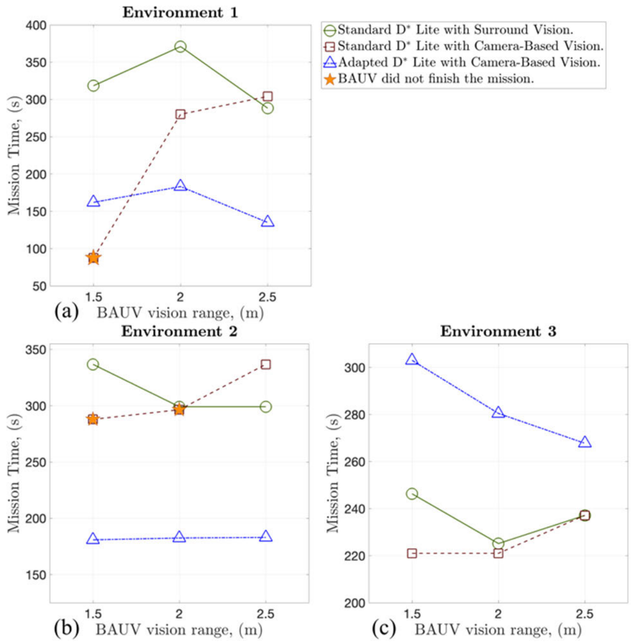

FIGURE 23. Mission times taken by the BAUV when the standard and our adapted D* Lite algorithms were implemented. Surround and camera-based vision ranges were set to 1.5, 2, and 2.5 m for (a) environment 1, (b) environment 2, and (c) environment 3.

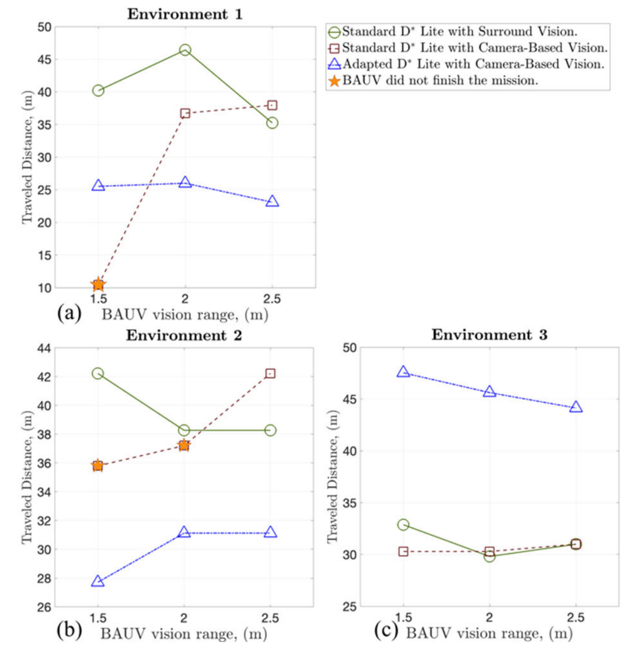

FIGURE 24. BAUV's final traveled distance from each mission after implementing standard and adapted versions of D* Lite on: (a) first scenario, (b) second scenario, and (c) third scenario.

Figure 18 shows the guidance and control stages for each mission developed inside Figure 17. For the guidance stage, the Euclidean distance to the goal E_d and the heading deviation e were iteratively computed to assess the BAUV's swimming performance. For the control stage, the tail flapping frequency and bias were regulated to correct moments and speed produced on the vehicle. Figure 18a through 18d shows the regulation stages when the standard D* Lite planner was implemented for the BAUV with a vision range set to 1.5 m. Figure 18e through 18h shows the same stages but with a vision range enlarged to 2.5 m. On the other hand, Figure 18i through 18l and 18m through 18p detail both stages once our adapted algorithm was incorporated into missions with the BAUV's vision range set to 1.5 and 2.5 m, respectively.

TABLE 4. Warning nodes cost update added to D* lite main procedure.

Procedure Main()
{21'} $s_{last} = s_{start}$;
{22'} Initialize ();
{23'} ComputeShortestPath();
{24'} while ($s_{start} \neq s_{final}$)
{25'} /* if ($g(s_{start}) = \infty$) then there is no known path */
{26'} $s_{start} = \underset{s' \in Succ(s_{start})}{\arg\min} \left(c(s_{start}, s') + g(s')\right)$;
{27'} Move to s_{start};
{28'} Check for changed edge costs;
{29'} if any edge costs changed;
{30'} $km = km + h(s_{last}, s_{start})$;
{31'} $s_{last} = s_{start}$;
{32'} for all directed edges (s, v) with changed edge costs
{33'} Update edge costs $c(s, v)$;
{34'} UpdateVertex(s);
{35'} **for all $n \in Adj(s)$ define warning nodes n_w**
{36'} **for all defined n_w**
{37'} **Update edge costs $c(n_w, v)$;**
{38'} **UpdateVertex(n_w);**
{39'} ComputeShortestPath()

Besides from the first case where the BAUV could not finish its mission, the vehicle traveled larger distances when the standard D* Lite path planner was employed. This resulted in longer mission times as well. This behavior can be shown in the reduction of E_d throughout both missions inside Figure 18a,e. Moreover, erratic behavior was produced when the standard D* Lite path planner was employed. After a new waypoint was introduced, the vehicle's heading direction presented abrupt changes causing large heading deviations. This is visualized on Figure 18b,f, where sudden and large changes during computation of heading deviation e are shown. During the control stage in Figure 18c,g, the propeller was practically set to flap from one side to the other since the vehicle was trying to reach all the sequence of waypoints, producing sharp turns. Frequency regulation throughout the missions is shown in Figure 18d,h.

In contrast, guidance and control stages were enhanced once the adapted local path planner was employed. First, mission times were reduced, and the vehicle could gradually reduce its distance to the goal faster than in the first two cases. This is visualized in Figure 18i,m. Second, few abrupt deviation errors were attained and even reduced (as seen in Figure 18j,n). By planning longer inner paths, the deviation error from the guidance system was better regulated,

TABLE 5. Comparison between D* Lite strategies for planning routes inside missions.

Environment	Vision range (m)		D* Lite version	Mission Time (s)	Waypoints Computed	Traveled Distance (m)	Expected Collisions	Was Mission Accomplished?
1	1.5	Surround	Standard	318.44	34	40.19	3	Yes
		Frontal	Standard	87.57	9	10.44	3	No
		Frontal	Adapted	162.02	18	25.52	0	Yes
	2.0	Surround	Standard	370.92	32	46.42	3	Yes
		Frontal	Standard	280.07	33	36.73	2	Yes
		Frontal	Adapted	183.18	18	26.00	0	Yes
	2.5	Surround	Standard	288.34	31	35.25	1	Yes
		Frontal	Standard	304.04	31	37.95	1	Yes
		Frontal	Adapted	135.09	12	23.09	0	Yes
2	1.5	Surround	Standard	336.73	31	42.22	6	Yes
		Frontal	Standard	287.90	27	35.80	7	No
		Frontal	Adapted	180.90	21	27.72	0	Yes
	2.0	Surround	Standard	299.09	31	38.27	3	Yes
		Frontal	Standard	296.47	28	37.21	6	No
		Frontal	Adapted	182.53	17	31.13	0	Yes
	2.5	Surround	Standard	299.09	31	38.27	3	Yes
		Frontal	Standard	336.73	31	42.22	8	Yes
		Frontal	Adapted	183.05	17	31.13	0	Yes
3	1.5	Surround	Standard	246.28	31	32.87	3	Yes
		Frontal	Standard	221.01	31	30.29	2	Yes
		Frontal	Adapted	302.96	34	47.53	0	Yes
	2.0	Surround	Standard	225.16	31	29.81	3	Yes
		Frontal	Standard	221.01	31	30.29	3	Yes
		Frontal	Adapted	280.42	27	45.62	0	Yes
	2.5	Surround	Standard	237.12	31	30.99	3	Yes
		Frontal	Standard	237.12	31	30.99	3	Yes
		Frontal	Adapted	267.75	25	44.14	0	Yes

decreasing the chances of causing the vehicle to describe abrupt turns. Then, the proper regulation of BAUV's heading direction enhanced the vehicle's overall swimming performance. In the control stage, abrupt changes in the propeller's performance and corrections over its flapping bias and frequency were reduced. A better regulation was achieved since gradual changes could be induced during missions (as seen on bias and frequency control in Figure 18k,l and 18o,p).

The experiment was simulated for the second environment. Figure 19 shows the results after implementing the standard and adapted versions of the D* Lite path planner during the navigation task. While 19a,b shows the navigation task after employing the standard algorithm, 19c,d shows the vehicle's performance when planning with our adapted algorithm.

Once more, the BAUV did not finish its mission when it presented a short vision range of 1.5 m, and the standard planner was implemented. The planner did not see and misclassified an obstacle vertex as the next best candidate. The BAUV collided, and the planner ended its tasks, as seen in Figure 19a. Once the vision range was enlarged inside Figure 19b, the vehicle could finish the mission and reach the final goal. However, the final paths described by the vehicle were not smooth and presented sharp turns. For both paths, collisions with obstacles are expected.

When the adapted D* Lite was implemented for the second scenario, the BAUV finished the mission and described smoother and safer paths. Figure 19c shows the results attained with BAUV's vision range set to 1.5 m. In this case, the BAUV swam right through the middle of all hallways avoiding collisions with obstacles. The distribution of waypoints allowed the vehicle to gradually regulate its orientation and position towards inner nodes and the final goal. Further, the vehicle finished the mission without collisions and described a safe path even inside halls.

Contrasting to results in Figure 19c, Figure 19d shows how the algorithm became a conservative planner when the vision range was enlarged from 1.5 to 2.5 m. The vehicle visualized larger regions than before, and more considerable environmental information was attained. Then, the prior definition of warning nodes provoked the final decision of surrounding the final obstacles. With these results, it can be shown how the adapted D* Lite algorithm will help in planning more achievable paths. Moreover, the decision of surrounding obstacles rather than passing through narrow hallways will be determined by the current information attained from the

BAUV, where conservative decisions are prioritized over the shortest path.

Figure 20 shows each mission's guidance and control stages simulated inside the second environment. When the standard D* Lite path planner was implemented, the BAUV was set to follow waypoints close to each other. This produced a poor swimming performance, presenting a slower reduction of the Euclidean distance to goal and increasing the mission time (Figure 20a,e). Also, the vehicle's heading direction presented large changes while swimming, making the regulation of BAUV's deviation e hard to handle (Figure 20b,f). The detected sudden changes in the BAUV's direction made the control stage induce abrupt corrections over flapping (Figure 20c,d and 20g,h).

When the adapted local planner was employed, the guidance and control stages developed a smoother regulation task. Since corrections were gradually induced, the vehicle enhanced its swimming performance. Moreover, even when the proposed algorithm preferred planning over open spaces rather than traversing narrow halls, the mission times did not increase. This is visualized in the adequate reduction of the Euclidean distance to the goal in Figure 20i,m. Also, a better regulation over heading deviation e could be achieved (as seen in Figure 20j,n). During this set of missions, the control stage (Figure 20k,l and 20o,p) gradually induced corrections over the flapping performance of the vehicle's tail to reduce path tracking errors.

The results for navigation tasks inside the third environment are shown in Figure 21. While Figure 21a,b shows the results after implementing the standard D* Lite, Figure 21c,d shows the results after incorporating our adapted algorithm. The standard planner suggested the shortest path through hallways, causing the vehicle to swim through unsafe narrow spaces, presenting collisions. On the other hand, the adapted algorithm allowed the vehicle to swim over safer paths toward the goal, surrounding narrow hallways and keeping a safe distance between obstacles. The control stage was largely enhanced, and, as in all previous experiments using our adapted algorithm, the swimming performance was also enhanced. The guidance and control stages for all missions inside the third environment are shown in Figure 22.

Figure 23 compares mission times taken by the vehicle when the standard and adapted planners were implemented. These results show that mission times were not enlarged even when our adapted algorithm suggested conservative paths. Surrounding obstacles may initially represent a costly strategy when considering mission times. However, this strategy reduced collision risks and allowed the BAUV to follow smoother paths with a smoother swimming performance, reducing navigation periods.

Further, Figure 24 shows the total traveled distance that the vehicle followed during each mission. The shortest path computed by the standard D* Lite often resulted in a hard-to-follow and inefficient trajectory because of the vehicle's hydrodynamics. The vehicle then tended to travel longer distances than expected, trying to reach all suggested waypoints. On the other hand, the developed, adapted D* Lite algorithm suggested conservative paths that, even when compared to the shortest path were longer, the vehicle would easily and safely follow, resulting in less expensive and efficient traveled distances. Finally, a comparison that summarizes results from simulated experiments between algorithms is shown in Table 5. The final comparison considers all scenarios, vision range radii, mission times, traveled distances, waypoints computed, and expected collisions obtained after each simulated experiment.

VII. CONCLUSION

In this article, the development of a local planner to enhance guidance, navigation, and control of a BAUV was presented. The vehicle swims by flapping a caudal fin, which produces a vectored thrust and turning moments. Based on a waypoint guidance system, the BAUV's heading deviation and distance toward a goal were tracked to regulate the propeller's flapping performance. Moreover, the vehicle's perception of its unknown environs was limited. Unlike surround vision systems, the BAUV could only see what was in front of it. The fast-planning D* Lite algorithm was adapted to the vehicle's conditions to help it reach its goals.

Conservative strategies such as prioritizing open spaces over free vertices near obstacles and planning inside BAUV's limited vision range helped attain achievable paths toward the final target. Furthermore, the adapted algorithm prioritized safer trails over the shortest path frequently computed by the standard algorithm. Then, even when the vehicle had to follow larger distances or swim inside open spaces rather than small shortcuts between obstacles (compared to the shortest paths advised by standard D* Lite), the collision risks were drastically reduced. Furthermore, the BAUV's swimming performance was enhanced since planning longer inner paths helped with the gradual regulation of flapping parameters inside the control stage. Finally, the characteristics of the proposed local planner may be expanded to other types of systems where a surrounding perception of the environment cannot be attained or where more conservative and safer paths are required due to the vehicle's motion constraints.

In future work, the enhancement of the planning stage may consider the vehicle's kinematics and hydrodynamics for the nodes-discrimination task. Further, smart planning techniques may be developed for the definition of paths to excel navigation performance. Also, complementary controllers may be added to enhance the process of tuning swimming parameters inside the BAUV. Then, adaptive and smart path tracking controllers may also be designed to regulate the swimming performance of this type of vehicles. Finally, intelligent algorithms such as deep learning could be incorporated in the control stage to improve travel efficiency.

$$\boldsymbol{M}=\begin{bmatrix} m-X_{\dot{u}} & 0 & 0 & 0 & z_g m & -y_g m \\ 0 & m-Y_{\dot{v}} & 0 & -z_g m & 0 & x_g m-Y_{\dot{r}} \\ 0 & 0 & m-Z_{\dot{w}} & y_g m & -x_g m-Z_{\dot{q}} & 0 \\ 0 & -z_g m & y_g m & I_{xx}-K_{\dot{p}} & 0 & 0 \\ z_g m & 0 & -x_g m-M_{\dot{w}} & 0 & I_{yy}-M_{\dot{q}} & 0 \\ -y_g m & x_g m-N_{\dot{v}} & 0 & 0 & 0 & I_{zz}-N_{\dot{r}} \end{bmatrix}. \tag{A1}$$

$$\mathbf{C}(\boldsymbol{\upsilon})=\begin{bmatrix} 0 & -mr & mq & (y_g q+z_g r)m & -x_g mq & -x_g mr \\ mr & 0 & -mp & x_g mq & z_g mr & 0 \\ -mq & mp & 0 & -z_g mp & -z_g mq & x_g mp \\ -(z_g r+y_g q)m & y_g mp & z_g mp & 0 & 0 & (I_{zz}-I_{yy})\,q \\ x_g mq & -x_g mp & z_g mq & (I_{xx}-I_{zz})\,r & 0 & -z_g mv \\ -x_g mr & y_g mr & x_g mp & 0 & (I_{yy}-I_{xx})\,p & 0 \end{bmatrix}. \tag{A2}$$

$$\mathbf{D}(\boldsymbol{\upsilon})=\begin{bmatrix} X_{u|u|}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{v|v|}|v| & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{w|w|}|w| & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{p|p|}|p| & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{q|q|}|q| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{r|r|}|r| \end{bmatrix}. \tag{A3}$$

$$\mathbf{g}(\boldsymbol{\eta})=\begin{bmatrix} (W-B)\,sin(\theta) \\ -(W-B)\,cos(\theta)sin(\phi) \\ -(W-B)\,cos(\theta)cos(\phi) \\ \left(z_g W-z_b B\right)\cos(\theta)\sin(\phi)+\left(y_g W-y_b B\right)cos(\theta)cos(\phi) \\ \left(z_g W-z_b B\right)\sin(\theta)+\left(x_g W-x_b B\right)\cos(\theta)\,cos(\phi) \\ \left(x_g W-x_b B\right)\cos(\theta)\sin(\phi)+\left(y_g W-y_b B\right)sin(\theta) \end{bmatrix}. \tag{A4}$$

APPENDIX A
BAUV'S HYDRODYNAMICS MODEL MATRICES

Added mass $\boldsymbol{M}$, Coriolis $\mathbf{C}(\boldsymbol{\upsilon})$, damping $\mathbf{D}(\boldsymbol{\upsilon})$,and hydrostatic forces $\mathbf{g}(\boldsymbol{\eta})$ matrices from the BAUV's hydrodynamics model are defined by (A1), (A2), (A3), and (A4), as shown at the top of the page, respectively.

REFERENCES

[1] A. Sahoo, S. K. Dwivedy, and P. S. Robi, "Advancements in the field of autonomous underwater vehicle," *Ocean Eng.*, vol. 181, pp. 145–160, Jun. 2019, doi: 10.1016/j.oceaneng.2019.04.011.

[2] R. Wang, S. Wang, Y. Wang, L. Cheng, and M. Tan, "Development and motion control of biomimetic underwater robots: A survey," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 52, no. 2, pp. 833–844, Feb. 2022, doi: 10.1109/TSMC.2020.3004862.

[3] D. T. Roper, S. Sharma, R. Sutton, and P. Culverhouse, "A review of developments towards biologically inspired propulsion systems for autonomous underwater vehicles," *Proc. Inst. Mech. Eng, M, J. Eng. Maritime Environ.*, vol. 225, no. 2, pp. 77–96, May 2011, doi: 10.1177/1475090210397438.

[4] A. Raj and A. Thakur, "Fish-inspired robots: Design, sensing, actuation, and autonomy—A review of research," *Bioinspiration Biomimetics*, vol. 11, no. 3, pp. 1–23, 2016, doi: 10.1088/1748-3190/11/3/031001.

[5] M. Sfakiotakis, D. M. Lane, and J. B. C. Davies, "Review of fish swimming modes for aquatic locomotion," *IEEE J. Ocean. Eng.*, vol. 24, no. 2, pp. 237–252, Apr. 1999, doi: 10.1109/48.757275.

[6] D. Scaradozzi, G. Palmieri, D. Costa, and A. Pinelli, "BCF swimming locomotion for autonomous underwater robots: A review and a novel solution to improve control and efficiency," *Ocean Eng.*, vol. 130, pp. 437–453, Jan. 2017, doi: 10.1016/j.oceaneng.2016.11.055.

[7] X. Wang, J. Liu, X. Su, H. Peng, X. Zhao, and C. Lu, "A review on carrier aircraft dispatch path planning and control on deck," *Chin. J. Aeronaut.*, vol. 33, no. 12, pp. 3039–3057, Dec. 2020, doi: 10.1016/j.cja.2020.06.020.

[8] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, Aug. 2021, doi: 10.3390/vehicles3030027.

[9] N. Sariff and N. Buniyamin, "An overview of autonomous mobile robot path planning algorithms," in *Proc. 4th Student Conf. Res. Develop.*, Jun. 2006, pp. 183–188.

[10] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proc. Int. Joint Conf. Artif. Intell.*, 1995, pp. 1652–1659.

[11] R. Kot, "Review of collision avoidance and path planning algorithms used in autonomous underwater vehicles," *Electronics*, vol. 11, no. 15, p. 2301, Jul. 2022, doi: 10.3390/electronics11152301.

[12] T. Yao, T. He, W. Zhao, and A. Y. M. Sani, "Review of path planning for autonomous underwater vehicles," in *Proc. Int. Conf. Robot., Intell. Control Artif. Intell.*, Sep. 2019, pp. 482–487, doi: 10.1145/3366194.3366280.

[13] S. Koenig and M. Likhachev, "D* lite," in *Proc. AAAI*, 2002, pp. 476–483. [Online]. Available: www.aaai.org

[14] T. T. Mac, C. Copot, D. T. Tran, and R. D. Keyser, "Heuristic approaches in robot path planning: A survey," *Robot. Auto. Syst.*, vol. 86, pp. 13–28, Dec. 2016, doi: 10.1016/j.robot.2016.08.001.

[15] D. Li, P. Wang, and L. Du, "Path planning technologies for autonomous underwater vehicles—A review," *IEEE Access*, vol. 7, pp. 9745–9768, 2019, doi: 10.1109/ACCESS.2018.2888617.

[16] M. I. Chowdhury and D. G. Schwartz, "The PRM—A* path planning algorithm for UUVs: An application to navy mission planning," in *Proc. Global Oceans, Singapore U.S. Gulf Coast*, Oct. 2020, pp. 1–9, doi: 10.1109/IEEECONF38699.2020.9388987.

[17] M. Ataei and A. Yousefi-Koma, "Three-dimensional optimal path planning for waypoint guidance of an autonomous underwater vehicle," *Robot. Auto. Syst.*, vol. 67, pp. 23–32, May 2015, doi: 10.1016/j.robot.2014.10.007.

[18] M. P. Aghababa, "3D path planning for underwater vehicles using five evolutionary optimization algorithms avoiding static and energetic obstacles," *Appl. Ocean Res.*, vol. 38, pp. 48–62, Oct. 2012, doi: 10.1016/j.apor.2012.06.002.

[19] S. Sedighi, D. Nguyen, and K. Kuhnert, "Guided hybrid A-star path planning algorithm for valet parking applications," in *Proc. 5th Int. Conf. Control, Autom. Robot. (ICCAR)*, Apr. 2019, pp. 570–575.

[20] C. V. Dang, H. Ahn, D. S. Lee, and S. C. Lee, "Improved analytic expansions in hybrid A-star path planning for non-holonomic robots," *Appl. Sci.*, vol. 12, no. 12, p. 5999, Jun. 2022, doi: 10.3390/app12125999.

[21] X. Wang, Z. Deng, H. Peng, L. Wang, Y. Wang, L. Tao, C. Lu, and Z. Peng, "Autonomous docking trajectory optimization for unmanned surface vehicle: A hierarchical method," *Ocean Eng.*, vol. 279, Jul. 2023, Art. no. 114156, doi: 10.1016/j.oceaneng.2023.114156.

[22] X. Wang, B. Li, X. Su, H. Peng, L. Wang, C. Lu, and C. Wang, "Autonomous dispatch trajectory planning on flight deck: A search-resampling-optimization framework," *Eng. Appl. Artif. Intell.*, vol. 119, Mar. 2023, Art. no. 105792, doi: 10.1016/j.engappai.2022.105792.

[23] X. Wang, J. Liu, H. Peng, X. Qie, X. Zhao, and C. Lu, "A simultaneous planning and control method integrating APF and MPC to solve autonomous navigation for USVs in unknown environments," *J. Intell. Robotic Syst.*, vol. 105, no. 2, p. 36, Jun. 2022, doi: 10.1007/s10846-022-01663-8.

[24] T. Praczyk, "Neural collision avoidance system for biomimetic autonomous underwater vehicle," *Soft Comput.*, vol. 24, no. 2, pp. 1315–1333, Jan. 2020, doi: 10.1007/s00500-019-03969-6.

[25] J. O. Gaya, L. T. Gonçalves, A. C. Duarte, B. Zanchetta, P. Drews, and S. S. C. Botelho, "Vision-based obstacle avoidance using deep learning," in *Proc. 13th Latin Amer. Robot. Symp. IV Brazilian Robot. Symp. (LARS/SBR)*, Oct. 2016, pp. 7–12, doi: 10.1109/LARS-SBR.2016.9.

[26] S. Zhao, T.-F. Lu, and A. Anvar, "Multiple obstacles detection using fuzzy interface system for AUV navigation in natural water," in *Proc. 5th IEEE Conf. Ind. Electron. Appl.*, Jun. 2010, pp. 50–55.

[27] 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 *Proc. 1st IEEE Int. Conf. Robotic Comput. (IRC)*, Apr. 2017, pp. 160–163, doi: 10.1109/IRC.2017.52.

[28] J. Peng, I. Li, Y. Chien, C. Hsu, and W. Wang, "Multi-robot path planning based on improved D* lite algorithm," in *Proc. IEEE 12th Int. Conf. Netw., Sens. Control*, Apr. 2015, pp. 350–353.

[29] 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, doi: 10.1109/ACCESS.2020.3021073.

[30] D. F. Myring, "A theoretical study of body drag in subcritical axisymmetric flow," *Aeronaut. Quart.*, vol. 27, no. 3, pp. 186–194, Aug. 1976, doi: 10.1017/S000192590000768X.

[31] J. A. Algarín-Pinto, L. E. Garza-Castañón, A. Vargas-Martínez, and L. I. Minchala-Ávila, "Dynamic modeling and control of a parallel mechanism used in the propulsion system of a biomimetic underwater vehicle," *Appl. Sci.*, vol. 11, no. 11, p. 4909, May 2021, doi: 10.3390/app11114909.

[32] G. Antonelli, "Modelling of underwater robots," in *Underwater Robots*, 3rd ed., B. Siciliano O. Khatib, Eds. Cham, Switzerland: Springer, 2014, pp. 23–31. [Online]. Available: http://www.springer.com/series/5208

[33] J. A. Algarín-Pinto, L. E. Garza-Castañón, A. Vargas-Martínez, and L. I. Minchala-Ávila, "Modeling, trajectory analysis and waypoint guidance system of a biomimetic underwater vehicle based on the flapping performance of its propulsion system," *Electronics*, vol. 11, no. 4, p. 544, Feb. 2022, doi: 10.3390/electronics11040544.

JUAN A. ALGARÍN-PINTO was born in Nuevo Laredo, Mexico, in 1995. He received the engineering degree in mechatronics from Tecnologico Nacional de Mexico, Campus Nuevo Laredo, Mexico, in 2018, and the M.Sc. degree in engineering science from Tecnologico de Monterrey, Monterrey, Mexico, in 2020, where he is currently pursuing the Ph.D. degree in engineering science.

From 2018 to 2020, he was a Research Assistant with the Renewable Energies Laboratory, Tecnologico de Monterrey, working on the assessment and maintenance of transmission line systems, where he is currently a Research Assistant with the Robotics Laboratory. His research interests include mobile robots, image processing, artificial intelligence, and mechatronics.

LUIS E. GARZA-CASTAÑÓN (Member, IEEE) was born in Monclova, Coahuila, Mexico, in 1963. He received the engineering degree in electronic systems, the M.Sc. degree in control engineering, and the Ph.D. degree in artificial intelligence from Tecnologico de Monterrey, Monterrey, Mexico, in 1986, 1988, and 2001, respectively.

From 1999 to 2003, he was a Lecturer with the Physics Department, Tecnologico de Monterrey, where he has been an Associate Professor with the Mechatronics Department, since 2004. He is the author of more than 100 articles, chapter books, and books. His research interests include autonomous vehicles, machine learning, fault detection, diagnosis and control, image processing, and advanced control applications.

ADRIANA VARGAS-MARTÍNEZ received the bachelor's degree in chemical engineering, the M.Sc. degree in environmental systems, and the Ph.D. degree in engineering science with specialty in automation from Tecnologico de Monterrey, Monterrey, Mexico, in 2005, 2007, and 2011, respectively, and the master's degree from Concordia University, Montreal, Canada, in 2013. In 2010, she was a Visiting Researcher with the Polytechnic University of Catalonia. She is currently the Associate Dean of the Digital School and Graduated Studies, Tecnologico de Monterrey. Her research interests include educational innovation, automatic control, intelligent control systems, and fault-tolerant control.

LUIS I. MINCHALA-ÁVILA (Senior Member, IEEE) received the B.S.E.E. degree from Salesian Polytechnic University, Cuenca, Ecuador, in 2006, and the M.Sc. and Ph.D. degrees from Tecnologico de Monterrey, Monterrey, Mexico, in 2011 and 2014, respectively. From Summer 2012 to Summer 2013, he was a Visiting Scholar with Concordia University, Montreal, QC, Canada. From 2017 to 2018, he was a Postdoctoral Fellow with the Climate Change Research Group, Tecnologico de Monterrey. From 2022 to 2023, he was a full-time Researcher with the Department of Mechatronics, Tecnologico de Monterrey, Campus Guadalajara. He has authored or coauthored over 60 indexed publications, including journal articles, conference proceedings, book chapters, and a book. His research interests include fault-tolerant control applied to energy systems, robotics, automation, and process control.