Integrated Autonomous Underwater Vehicle Path Planning and Collision Avoidance Algorithms

Peter Danielis[1], Willi Brekenfelder[1], Helge Parzyjegla[1], Pranavkumar Ghoghari[1], Paul Stube[1], Frank Sill Torres[2]

This paper addresses the growing need for efficient autonomous underwater vehicle (AUV) path planning, used in environmental monitoring, underwater surveillance, and search and rescue operations. The aim was to develop robust path planning strategies that ensure safe and efficient AUV navigation in complex and unpredictable underwater environments. Our approach combines path planning algorithms with the OMNeT++ network simulator. The procedures for the implementation of these algorithms and AUV motion simulation have been outlined. The algorithms were applied not only to single AUV missions but also to scenarios involving multiple AUVs, allowing exploration of cooperative and coordinated mission planning in diverse underwater settings. The results of the study demonstrate the potential of our approach to address real-world challenges encountered by AUVs. AUV behavior was observed in different simulated mission scenarios and environmental conditions. Our findings shed light on the adaptability of AUVs in the face of unexpected obstacles and dynamic ocean currents. In conclusion, this study contributes to the field of AUV applications, offering a path planning and simulation strategy adaptable to diverse AUV mission requirements. By utilizing simulation models, we illustrate AUVs' ability to autonomously adapt their mission plan and avoid obstacles during missions, improving operational efficiency and safety.

KEY WORDS

~ AUVs
~ Path planning
~ Collision avoidance
~ Simulation

[1] University of Rostock, Institute of Computer Science, Rostock, Germany
[2] German Aerospace Center, Institute for the Protection of Maritime Infrastructures, Bremerhaven, Germany
e-mail: peter.danielis@uni-rostock.de

doi: 10.7225/toms.v13.n01.003

Received: 27 Dec 2023 / Revised: 2 Mar 2024 / Accepted: 18 Mar 2024 / Published: 20 Apr 2024

This work is licensed under

1. INTRODUCTION

Our planet is predominantly covered by water, with over 70% of its surface concealed beneath its vast, uncharted depths. These underwater expanses hold immense potential for renewable energy resources, including wind energy, tidal energy, and wave energy. To unlock this concealed energy, comprehensive ocean surveys are essential. Autonomous Underwater Vehicles (AUVs) are emerging as a promising solution for ocean exploration and surveying. AUVs are robotic systems capable of operating underwater independently, equipped with various sensors and actuators to execute a range of tasks (Aschenbruck, et al., 2012).

Recent years have seen a surge of interest in AUV research across diverse fields. This paper aligns with this growing interest by addressing the challenge of path planning for AUVs and simulating their movement during missions. Technological advancements and increasing computational capacity have expanded the horizons of simulation capabilities to include applications from transportation to biology. Notably, the demand for a simulation environment tailored to AUV mission planning has become increasingly urgent. Several research projects have tackled this issue, yielding various simulation platforms (Cook, et al., 2014).

This paper contributes to the field of path planning for both individual and multiple AUVs. This involves the use of appropriate algorithms, the creation of a simulation platform using the OMNeT++ framework, and the incorporation of collision avoidance mechanisms. These efforts enable mission plan modification in response to unexpected obstacles. The significance of path planning is highlighted, as simulations depend on predefined paths. AUV movement was simulated using the BonnMotionMobility model in the OMNeT++ framework INET (OMNeT++ Developer Team, 2023). This model relies on trace-based data, which include time, x-coordinates, and y-coordinates, effectively defining AUV speed based on the distance and time elapsed between coordinates.

The remainder of the paper is organized as follows. First, we look at related research and cover fundamental concepts that contribute to the comprehension of the remaining content in this paper in Section 2. In Section 3, the concept for our simulation models is introduced. The concept includes the formulation of path plans for both individual AUVs and multiple AUVs tailored to a specific mission type. This involves the implementation of suitable algorithms and the capability to autonomously detect and navigate around objects of various shapes, allowing the AUVs to adapt their mission plans independently. Section 4 describes how the concept is implemented in OMNeT++ and INET and gives an evaluation based on case studies. Section 5 gives conclusions.

2. RELATED RESEARCH

There have been numerous studies on AUV path planning, mostly using graph searching-type, sampling-based, and potentially, field-based algorithms (An, et al., 2023). For example, Aguiar et al. propose AUV route optimization through the use of high-resolution ocean models (Aguiar, et al., 2019). This involves the acceleration and speed of the AUV, which are optimized by means of more accurate ocean models which allow currents or other environmental effects to be considered during path planning. Liu et al. present an approach based on the estimation of distribution algorithms (EDA), i.e. the learning fixed-height histogram (LFHH) method, to address path planning challenges in AUVs operating in dynamic environments (Liu, et al., 2019). This approach incorporates a Learning Transformation Strategy (LTS) aimed at improving both accuracy and convergence velocity. Additionally, the utilization of a smoothing technique is implemented to expedite the process of identifying viable navigation paths. Furthermore, the integration of a planning window is proposed to effectively manage dynamic environmental elements. Wei et al. present an Evolutionary Strategy-based Hyperheuristic (ES-HH) algorithm, merging metaheuristics and a selection function for the online heuristic evaluation of AUV mission planning (Wei, et al., 2022). This approach improves computational efficiency and robustness, yielding near-optimal outcomes. Comparative experiments demonstrate its superior convergence and robustness over algorithms like Ant Colony and Biogeography-based Optimization.

With respect to collision avoidance solutions, Han et al. introduce an early warning, obstacle avoidance path planning method for multi-AUV networks in underwater IoT-enhanced marine technology systems (Han, et al., 2023). Utilizing Software Defined Networking (SDN) to improve scalability and control, the proposed Software Defined multi-AUV-based Underwater Network (SD-UWN) architecture employs flexible topology and artificial potential field theories for network control. This approach, focusing on obstacle avoidance, uses SD-UWN data sharing for effective path planning. Shen at al. propose an algorithm for efficient online coverage path planning in unknown terrains using curvature-constrained AUVs (Shen, et al., 2019). The algorithm, an improved ε* algorithm integrating an exploratory Turing machine for supervision, optimizes the so-called Dubins path selection for these AUVs, ensuring effective global coverage despite their limited maneuverability and acceleration capabilities. The research by Heshmati-Alamdari et al. used a robust nonlinear model predictive control (NMPC) method for underactuated AUVs, specifically unicycle-like vehicles, to navigate 3-D trajectories in constrained spaces with obstacles (Heshmati-Alamdari, et al., 2021). It ensures collision-free path tracking in dynamic environments with input and state constraints, adapting to real-time sensory data.

2.1. OMNeT++ and INET

OMNeT++, i.e. Objective Modular Network Testbed in C++, serves as a C++ simulation framework and library primarily tailored for network simulator construction. It acts as a versatile platform for various network types, including wired and wireless communication networks, networks on a chip, and queuing networks. Despite not functioning as a network simulator in itself, OMNeT++ provides essential features such as an Eclipse-based IDE, a graphical runtime environment, and diverse tools. The framework has garnered widespread usage in scientific and industrial circles, boasting a large user community (Varga, 2010).

The INET framework, i.e. Internet Engineering Task Force (IETF) Network Simulation Suite, operates as a modeling library within the OMNeT++ simulation environment. Originating as a toolkit developed by the IETF for network protocol development and testing, INET specializes in Internet protocol and network simulation. It plays a key role in designing and validating new protocols, especially in scenarios involving unfamiliar or novel protocols (Mészáros, et al., 2019). INET extends its capabilities to include the Internet stack (TCP, UDP, IPv4, IPv6, OSPF, BGP, etc.), cellular support, MANET, DiffServ, LDP, and MPLS Protocol with RSVP-TE signaling. Additionally, it provides several application models and serves as the foundation for various simulation frameworks that extend it in specific directions, such as media, overlay/peer or LTE networks.

OMNeT++ and INET combined are a robust foundation for simulating and studying complex communication networks and protocols.

2.2. BonnMotionMobility Model

BonnMotion, a software developed for mobile scenario creation and analysis, enables studying mobile ad hoc network properties. Jointly developed by the Communication Systems Group of the University of Bonn, Germany, the Worker Group of the Colorado School of Mines, Golden, Colorado, USA, and the Distributed Systems Group from the University of Osnabrück, Germany, BonnMotion aids in exporting scripts for network emulators like Ns-2 and Ns-3 (Aschenbruck, et al., 2010). The BonnMotionMobility model, integrated into the INET framework, operates on a trace-based approach. It utilizes pre-recorded trace files, which are either text or XML files containing map execution steps. The native file format of the BonnMotion simulation engine is employed, where each line describes the host's operation. The trace file specifies node trajectories either in triplets (t, x, y) or quadruplets (t, x, y, z), depending on the 2D or 3D nature of the simulation. The trace-based approach ensures the accurate representation of node movements, a crucial aspect for complex scenario simulation. This integration of OMNeT++, INET, and BonnMotionMobility sets the stage for subsequent AUV exploration in the simulated environment.

3. CONCEPTS FOR PATH PLANNING AND COLLISION AVOIDANCE

This section presents the concepts for path planning algorithms and collision avoidance for AUVs to react to unforeseen obstacle during their mission. The discussion is divided into two main parts: path planning algorithms and collision-avoidance mechanisms.

3.1. Path Planning Algorithms

This subsection explains the algorithms employed to plan the paths of a single and multiple AUVs in more detail.

Point to point navigation is a fundamental maneuver required of every underwater vehicle. The complexity of path planning increases with the introduction of obstacles, requiring careful plotting to reach the goal while avoiding obstacles. Path planning algorithms are instrumental for determining the optimal path in complex environments. Motion planning, also known as path planning, addresses the computational problem of identifying valid configurations to move an object from a source to a destination, applicable across computational geometry, computer animation, robotics, and computer games (Jaulin, 2001).

3.1.1. Path Planning for a Single AUV

Path planning for a single AUV involves plotting AUV course from its starting point to its endpoint, optimizing for the best trajectory. Path-planning algorithms, specifically those utilizing search-based methods, were used for offline mission planning. Search-based planning, a motion planning technique, employs graph search methods to compute paths or trajectories within a discrete representation of the problem.

Several algorithms were compared in the same test environment and obstacle setup. The objective of the evaluation was to identify an algorithm that meets the path planning purpose, bearing in mind the costs. Given that more than one algorithm successfully plotted the path in the specified environment, a more intricate comparison became necessary. The evaluated algorithms were subjected to a detailed comparative analysis using performance metrics outlined in (Toma, et al., 2021). The metrics included:

1. Computational time: the time an algorithm takes to plot the path.
2. Success rate: the probability that an algorithm will finding a suitable path in different scenarios.
3. Distance to goal: the distance between the node's last position and the target location in the event of failure.
4. Path length: the overall length of the plotted path to reach the destination.
5. Deviation: the difference between the plotted path and the shortest path plotted by any tested algorithm for the same environment.

The above metrics helped identify Dstar as the algorithm of choice. Among other algorithms, including Bidirectional Astar, Astar, Anytime Dstar, Dijkstra, Best-First Search, Lifelong Planning Astar, Dstar Lite, Real-Time Adaptive Astar, Breath-First Search, Anytime Repairing Astar, Learning Real-Time Astar, and Depth-First Search, the Dstar algorithm earned the best overall ranking. The ranking utilized a low-point system, summing up the ranking positions across all metrics, with the algorithm having the lowest overall sum deemed the most suitable.

Dstar, short for "Dynamic Astar," shares similarities with the well-known Astar path-finding algorithm designed to find the shortest path between two nodes in a graph (Hart, et al., 1968). Astar leverages a heuristic function to calculate the cost of reaching the destination from a particular node, determining the most suitable successor of the current node. Dstar introduces the capability to reconfigure parameters, which improves its suitability for environments with minimal changes, such as when only a single obstacle is added to an environment. It is crucial to clarify that despite its "dynamic" designation, Dstar is an offline planning algorithm and should not be confused with a dynamic online planning algorithm. Dstar and Astar rely on a comparable set of data structures, with Dstar having an additional list for maintenance (Stentz, 1994).

3.1.2. Path Planning for Multiple AUVs

Larger scale missions frequently require the involvement of multiple agents to effectively accomplish mission objectives. For example, when surveying an extensive ocean area, employing a single AUV traversing back and forth would be time-consuming and suboptimal. By contrast, the synchronized operation of multiple AUVs can expedite task completion, improving energy efficiency of each individual AUV. Path planning for multiple AUVs extends the principles of single AUV path planning, aiming to guide each AUV from its starting point to the goal. However, utilizing the Dstar algorithm for planning the paths of multiple AUVs is impractical, as it would treat the starting positions of other AUVs solely as obstacles, potentially resulting in collisions. Instead, the Astar algorithm dynamically adjusts to current positions of other AUVs during path planning to mitigate collision risks.

The Multi-Agent Path Finding (MAPF) problem is often conceptualized as a hybrid of single-agent pathfinding, frequently involving navigation through the same map from distinct starting nodes. Taking this a step further necessitates coordinated efforts among agents. To integrate this coordination, a regression-based method was used to identify all potential collision-free paths. Multiple AUV planning adopts a conflict-based pathfinding method, explained in (Sharon, et al., 2015). Conflict-based search (CBS) emerges as an optimal pathfinding algorithm addressing MAPF problems. CBS operates on a two-level framework, obviating the need for employing the Astar algorithm for multiple AUV path planning. Paths can be planned individually for each AUV using Dstar, as collisions are explicitly handled at the highest level.

At the top level, a conflict search is conducted through a Constraint Tree (CT), a tree rooted in individual agent conflicts. Each constraint defines a specific timeframe during which an agent is prohibited from occupying a particular location. Each CT node embodies a set of restrictions on agent motion. To adhere to the constraints imposed by high-level CT nodes, swift single-agent searches are conducted at the low level, using the Dstar algorithm. CBS defines a set of constraints and subsequently finds paths consistent with those constraints. If these paths intersect, they are deemed invalid, necessitating the addition of new constraints.

The multi-agent CBS starts by identifying potential conflicts between agents using a high-level search algorithm. This involves scrutinizing the start and destination positions of each agent to pinpoint instances when more than one agent is simultaneously present at the same location, leading to a conflict. These conflicts are amalgamated into a conflict set, serving as input for the low-level algorithm. Knowing locations and evasion times makes it possible to plot new paths for each agent until all constraints are met.

3.2. Collision-Avoidance Mechanisms

In this subsection, the collision-avoidance concept is explained. Collision-avoidance mechanisms enable AUVs to function autonomously, which requires the capability to respond to diverse situations and navigate around potential issues, such as unforeseen obstacles not accounted for during mission planning. To facilitate this adaptability, mechanisms must be in place to handle various collision avoidance scenarios, prompting the AUV to deviate from its initially planned mission route when necessary.

To execute an evasive maneuver, several considerations come into play. First, the AUV must detect an obstacle. Once detected, logic is applied to determine the optimal obstacle avoidance path for the AUV. The final step involves the actual avoidance maneuver, specifying the actions the AUV must take to safely navigate around the obstacle. The architecture of the AUV simulation model is modular, comprising three modules, as

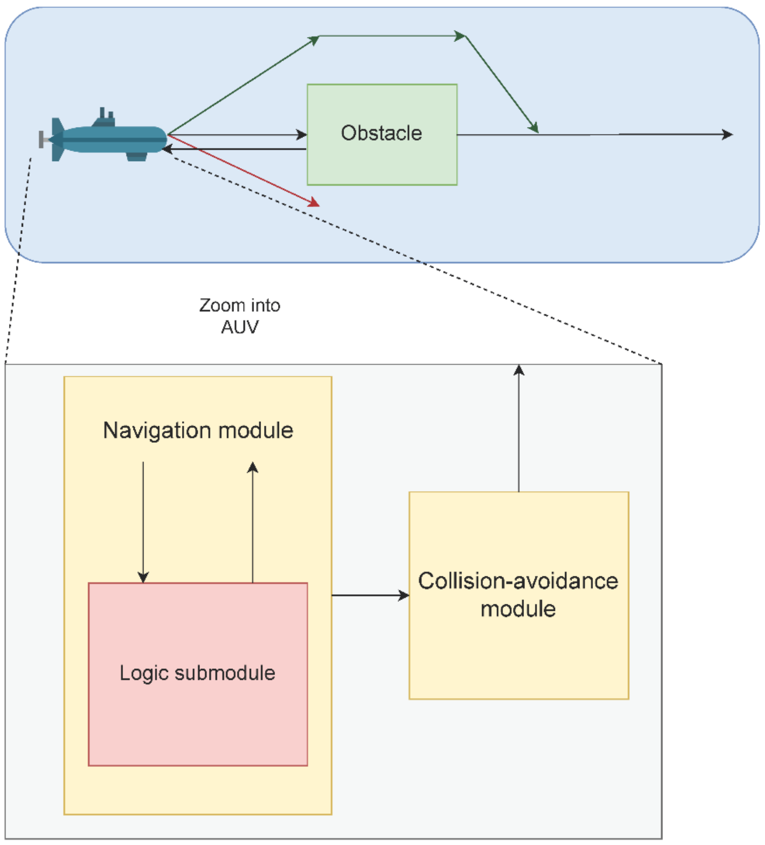
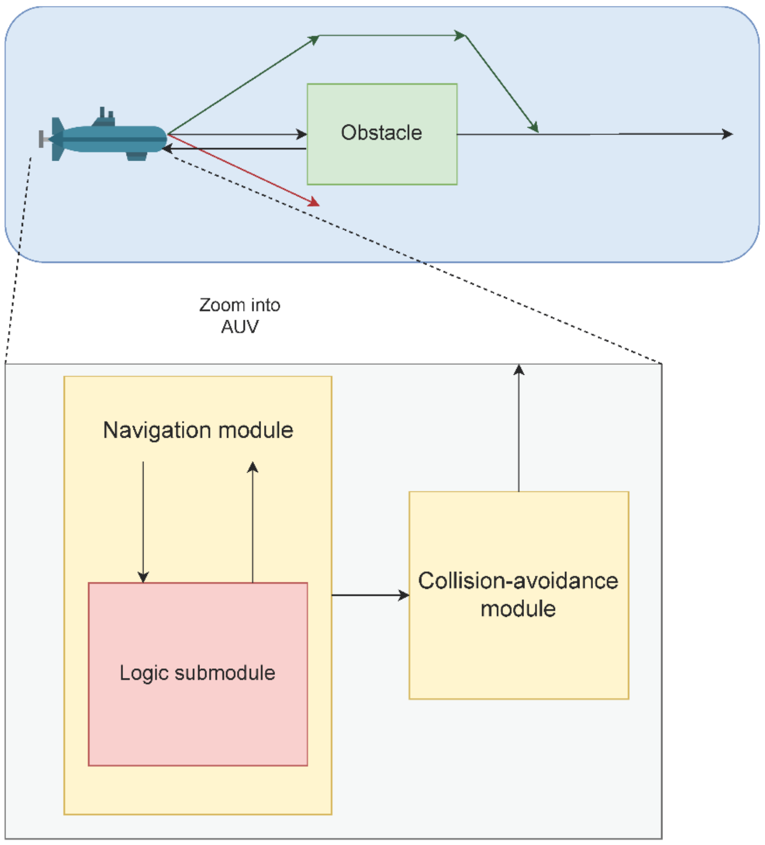


Figure 1. Modular architecture of the AUV



depicted in

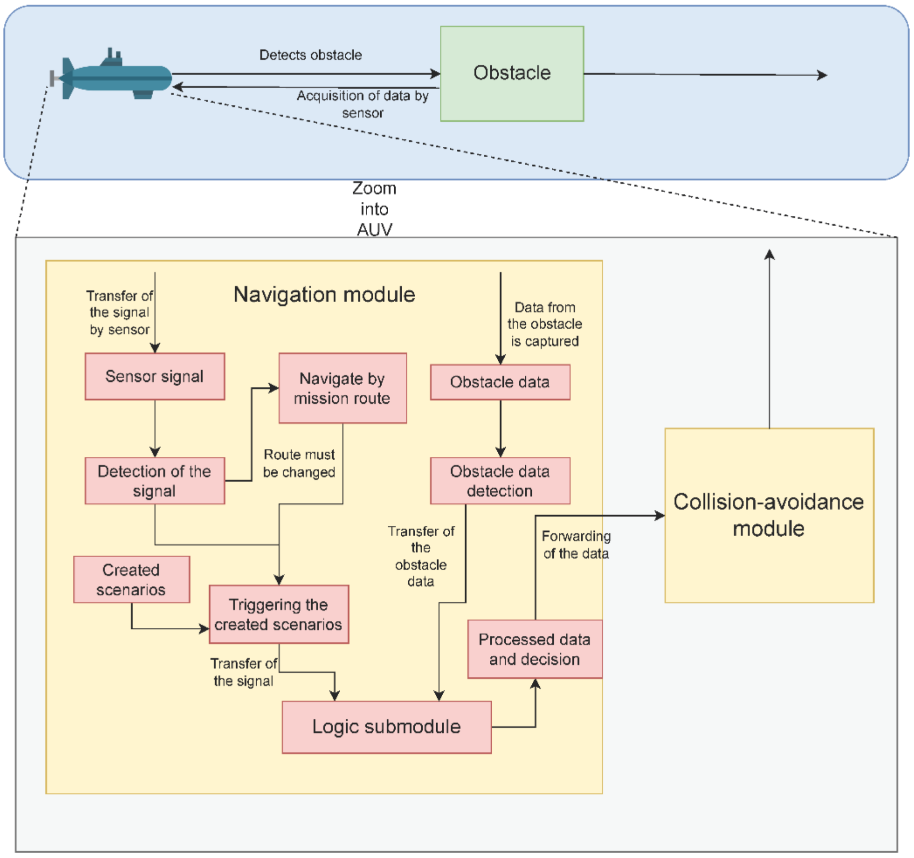
Figure 1. The AUV, moving towards an obstacle, identifies the obstacle and alters its course based on the decision-making process, illustrated by red and green arrows.

The entire avoidance process unfolds across three modules, detailed below.

3.2.1. Navigation Module

As previously noted, the navigation module plays a crucial role in obstacle detection, as well as in the initial processing and transmission of relevant data. It also encompasses the general navigation functions of the AUV. To gain a comprehensive understanding of the navigation module, including the data flow, processing steps, and output, its underlying concepts and involvement in evasion strategies need to be examined.

Navigation module architecture is illustrated in



Detects obstacle
Obstacle
Acquisition of data by sensor
Zoom into AUV
Navigation module
Transfer of the signal by sensor
Sensor signal
Navigate by mission route
Data from the obstacle is captured
Obstacle data
Detection of the signal
Route must be changed
Obstacle data detection
Collision-avoidance module
Transfer of the obstacle data
Forwarding of the data
Created scenarios
Triggering the created scenarios
Processed data and decision
Transfer of the signal
Logic submodule

Figure 2. At the beginning of the maneuver, the AUV follows its regular mission route. If a sensor detects an obstacle in its path, a signal is promptly dispatched, indicating the presence of an obstacle. Simultaneously, pertinent information about the obstacle, such as the absolute coordinates of its sides (points at the height of the AUV), including orientation (left or right of the AUV), is gathered. This data is then conveyed to the navigation module, which, in turn, transmits the obstacle data directly to the logic submodule situated within the navigation module.

Upon receiving the obstacle signal, a sequence of actions is triggered within the navigation module. The mission route must be abandoned, and the AUV must compute an alternative route. To facilitate this, predefined scenarios within the navigation module enable the AUV to respond to various situations. For example, if the

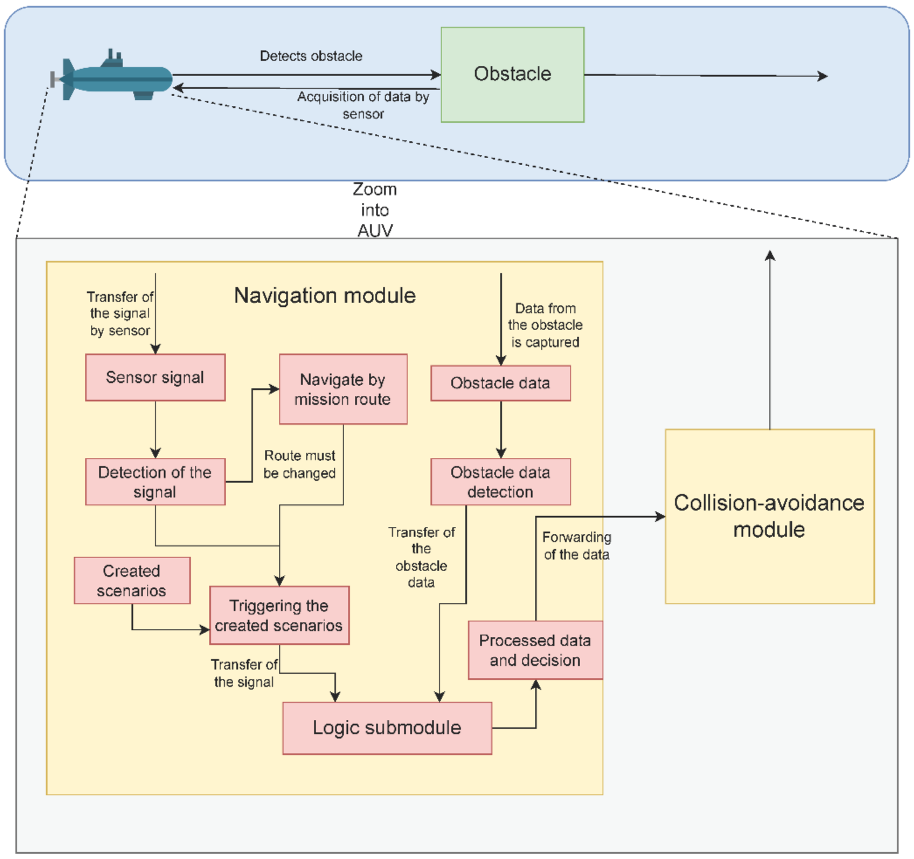

Detects obstacle
Obstacle
Acquisition of data by sensor
Zoom into AUV
Navigation module
Transfer of the signal by sensor
Sensor signal
Navigate by mission route
Data from the obstacle is captured
Obstacle data
Detection of the signal
Route must be changed
Obstacle data detection
Transfer of the obstacle data
Forwarding of the data
Collision-avoidance module
Created scenarios
Triggering the created scenarios
Transfer of the signal
Processed data and decision
Logic submodule

Figure 2. Navigation module architecture

obstacle is situated directly on the AUV's path, the navigation module prompts the logic submodule to conduct necessary calculations to avoid it. After processing the data in the logic submodule, including the avoidance route decision, this information is relayed back to the navigation module. Subsequently, the data is transmitted to the collision-avoidance module.

In essence, the navigation module functions as a distributor, consolidating incoming data and disseminating it to the appropriate modules. This distribution process follows predefined scenarios, allowing for modular extensions by introducing new scenarios to govern other modules. This modular flexibility ensures easy expansion of the navigation module to accommodate new scenarios, improving its adaptability in diverse operational conditions.

3.2.2. Logic Module

Assisted by the navigation module, the AUV is capable of obstacle detection and the collection of necessary obstacle-related information. The data signaling the presence of an obstacle in front of the AUV is transmitted to the logic submodule. Such event initiates the actual evasion maneuver. Upon recognizing an obstacle ahead, the AUV recognizes it needs to deviate from its current mission route and navigate around the obstacle. The crucial question is how the AUV will execute this avoidance, plot a new course. As illustrated in Figure 3, numerous potential routes exist, presenting only a subset of examples since, in principle, there are virtually infinite possibilities. Ideally, the AUV will choose the optimal path with respect to minimal energy consumption, speed or distance. Presently, the optimal path is determined based on the absolute distance from the AUV to one of the two outermost points of the detectable obstacle.

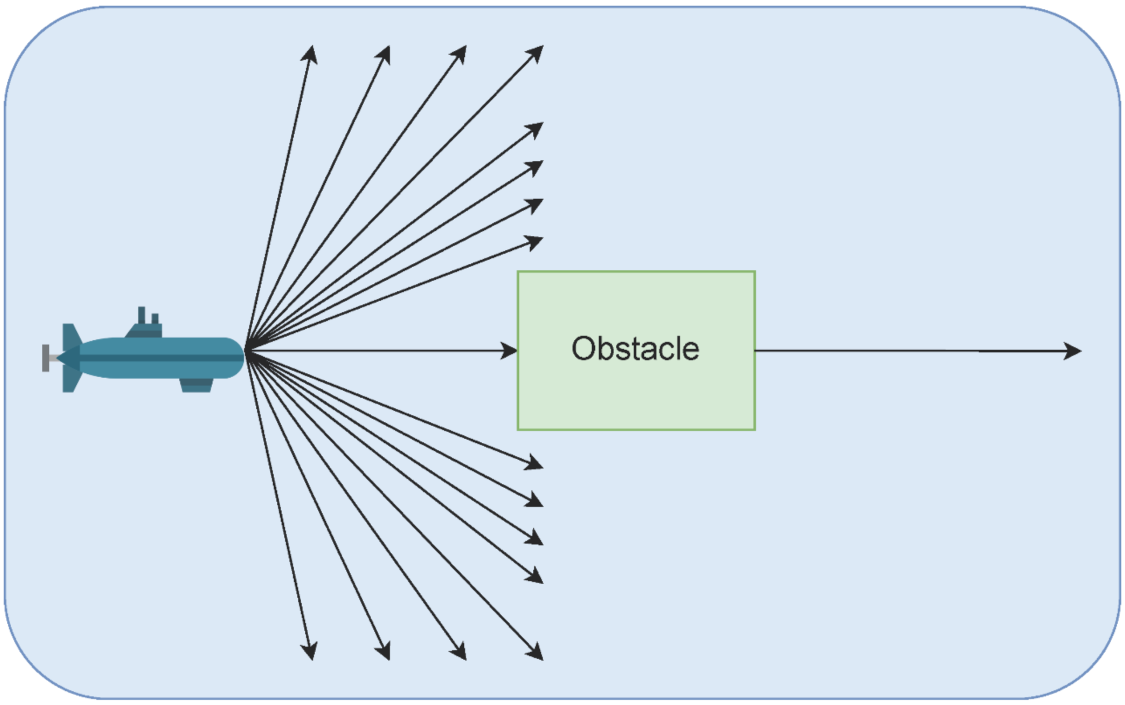


Figure 3. Cutout of potential evasion maneuver routes

The logic submodule needs two types of data to plot the path. First, sensor signals to indicate the presence of an obstacle. Second, AUV information on the obstacle, specifically the absolute coordinates of the outermost sides of the obstacle visible to the AUV, along with the orientation of these coordinates (right or left). Upon receiving this data, the logic submodule commences its calculations, illustrated in

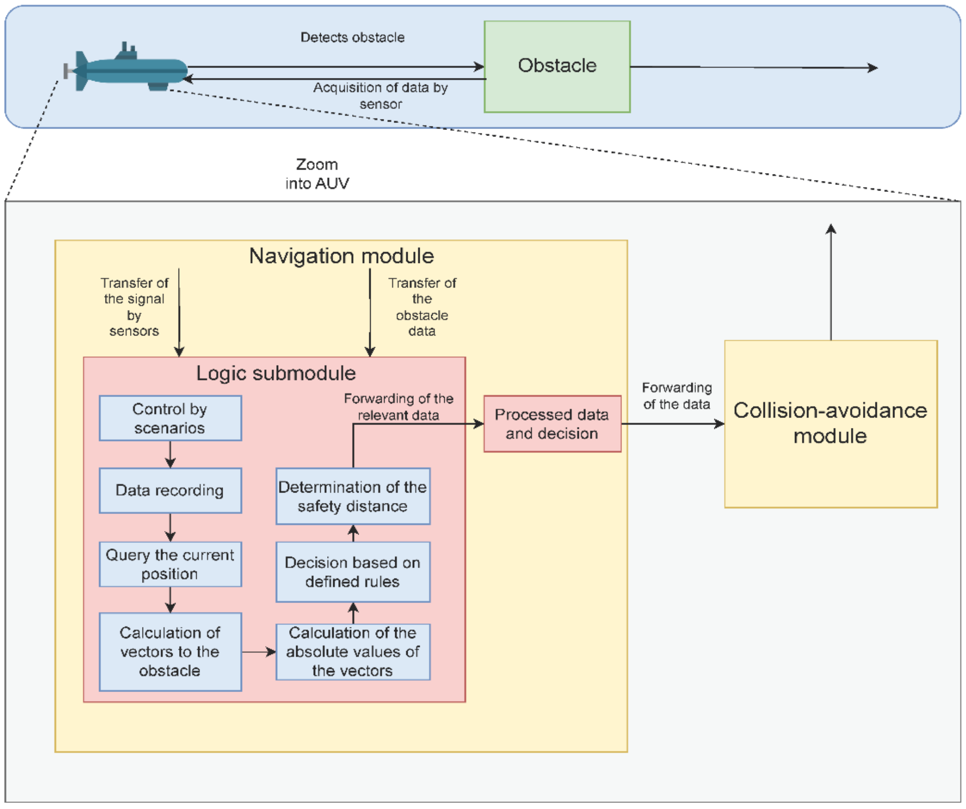

Detects obstacle
Obstacle
Acquisition of data by sensor
Zoom into AUV
Navigation module
Transfer of the signal by sensors
Transfer of the obstacle data
Logic submodule
Control by scenarios
Forwarding of the relevant data
Processed data and decision
Forwarding of the data
Collision-avoidance module
Data recording
Determination of the safety distance
Query the current position
Decision based on defined rules
Calculation of vectors to the obstacle
Calculation of the absolute values of the vectors

Figure 4. Data is transferred from the navigation module to the logic submodule, initiating the decision-making process. The submodule retrieves additional data, including the current position of the AUV, crucial for calculating the absolute distance to the outermost edges of the obstacle.

Absolute distances are determined by calculating vectors from the current position of the AUV to the outermost visible edges of the obstacle. Subsequently, the magnitude of these vectors is computed, serving as the absolute distance. This value guides the decision on the direction in which the obstacle will be avoided. Decisions are made based on predefined rules, prioritizing the smaller absolute value of vectors, signifying the shorter path. Consideration is given to the orientation of obstacle edges, ensuring correct conclusions are drawn from the rules, as depicted in Figure 5. The rules involve ratios of absolute vector values, determining whether to move right or left based on edge orientation. Upon reaching a decision, it is saved, and the distance between the AUV and the obstacle is determined in the final step. A safe distance, comprising the absolute value of the vector on the side where the obstacle is to be avoided and a user-defined fixed value, is then transmitted to the navigation module along with the route decision. Subsequently, this data is forwarded to the collision-avoidance module, detailed in the subsequent section.

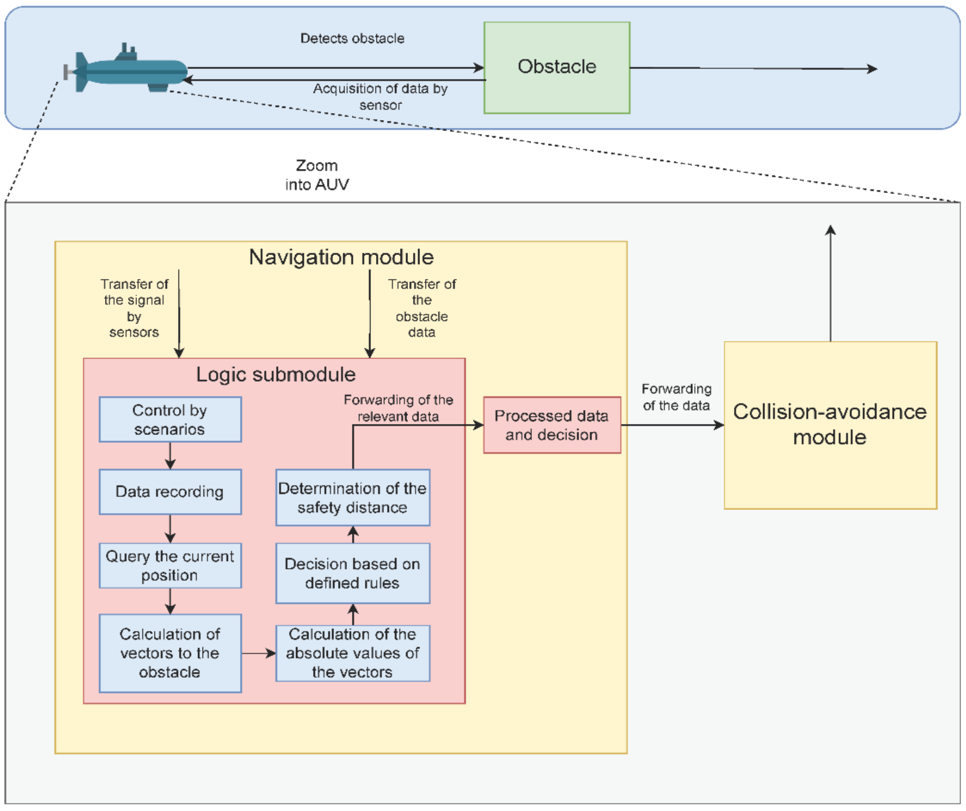
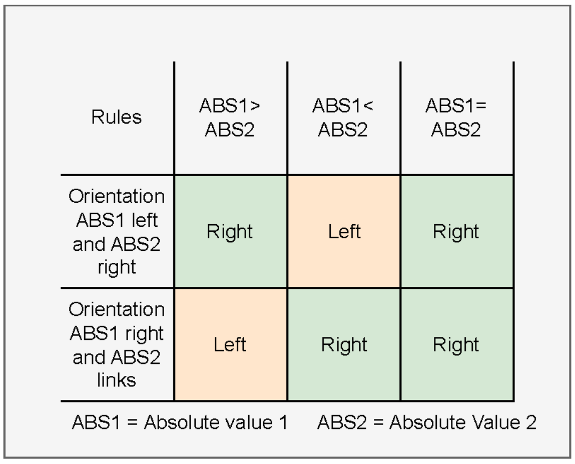

Detects obstacle
Obstacle
Acquisition of data by sensor
Zoom into AUV
Navigation module
Transfer of the signal by sensors
Transfer of the obstacle data
Logic submodule
Control by scenarios
Forwarding of the relevant data
Processed data and decision
Forwarding of the data
Collision-avoidance module
Data recording
Determination of the safety distance
Query the current position
Decision based on defined rules
Calculation of vectors to the obstacle
Calculation of the absolute values of the vectors

Figure 4. Architecture of the logic module

3.2.3. Collision-Avoidance Module

With the help of the logic submodule, the AUV efficiently detects obstacles, processes obstacle data, and makes informed evasion route decisions. This subsection focuses on describing the movements constituting the core of the evasive maneuver itself, as illustrated in Figure 3. As previously emphasized, the different ways to execute an evasion are outlined, with the current discussion centered on determining the movement characteristics, how the AUV deviates from its original route, maintains a safety distance, and eventually continues along the mission route.



Rules	ABS1> ABS2	ABS1< ABS2	ABS1= ABS2
Orientation ABS1 left and ABS2 right	Right	Left	Right
Orientation ABS1 right and ABS2 links	Left	Right	Right

ABS1 = Absolute value 1 ABS2 = Absolute Value 2

Figure 5. Simplified representation of collision avoidance rules

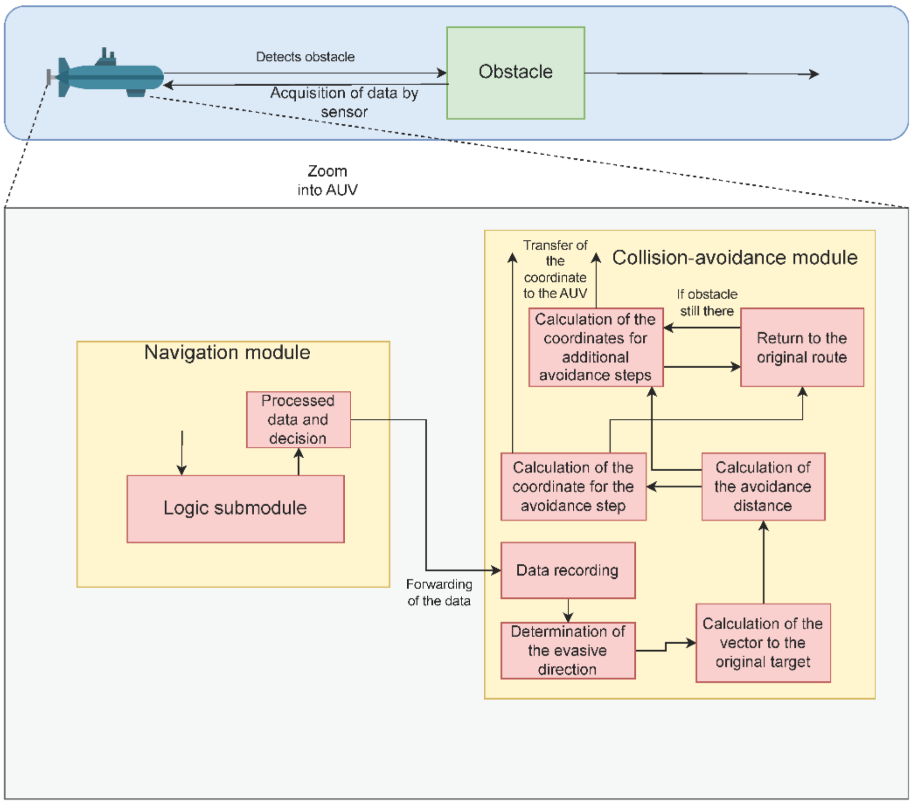

Detects obstacle
Obstacle
Acquisition of data by sensor
Zoom into AUV
Navigation module
Processed data and decision
Logic submodule
Forwarding of the data
Collision-avoidance module
Transfer of the coordinate to the AUV
If obstacle still there
Calculation of the coordinates for additional avoidance steps
Return to the original route
Calculation of the coordinate for the avoidance step
Calculation of the avoidance distance
Data recording
Determination of the evasive direction
Calculation of the vector to the original target

Figure 6. Architecture of the collision-avoidance module

At the core of this maneuver is the chosen motion model, BonnMotionMobility. As a route-based model, it sticks to prescribed paths, a feature advantageous for collision avoidance. This model facilitates collision avoidance by supplying the AUV with a new set of coordinates, thus defining an alternative route. Figure 6 depicts the architecture of the collision avoidance module. It initiates its operation by receiving data from the navigation module, encompassing the decision on the evasion direction and pertinent details about the obstacle, along with the predefined safety distance. Once this data is processed, the module evaluates the avoidance direction decision and identifies the direction for subsequent calculations. The next step involves avoidance route plotting, commencing with the calculation of the vector from the current position to the original destination, i.e., the current obstacle location. Subsequently, the final avoidance distance is determined, incorporating the absolute value of the vector to the original target and safe distance.

This calculated distance is employed to compute the coordinate for the collision-avoidance step, generally ensuring that the vector to this coordinate is approximately perpendicular to the vector to the original target. Following this calculation, the evasion coordinate is conveyed to the AUV, prompting the AUV to maneuver to this point. Upon reaching this point, the AUV returns to its initial route. However, the obstacle may still be close to the AUV, prompting the sensors to detect its presence and trigger a signal. In this scenario, a new decision on the swerving direction is not necessary. Instead, the data is relayed to the logic submodule, which modifies the data, signaling the avoidance module to execute a distinct move. Subsequently, the avoidance module utilizes the avoidance distance for calculations, akin to the initial avoidance step.

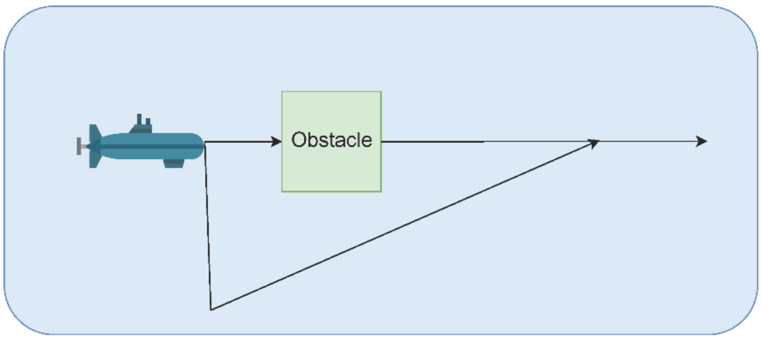


Figure 7. Representation of a simple maneuver

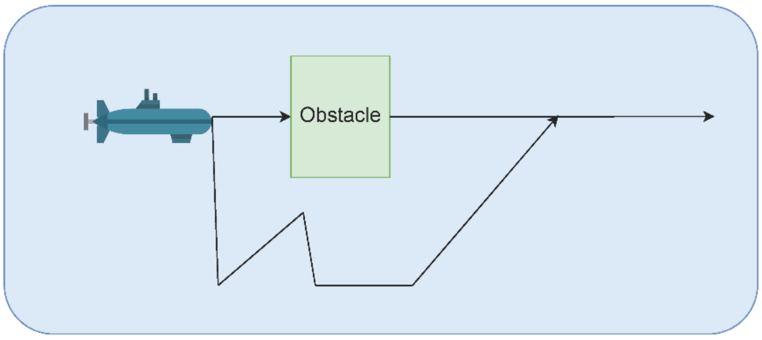


Figure 8. Representation of an advanced maneuver

A new coordinate is then computed for the additional collision-avoidance step, with a position akin to the first coordinate but shifted in the direction of the mission route. Consequently, the AUV follows a route parallel to the obstacle, equivalent to the length of the avoidance distance. This new coordinate is transmitted back to the AUV, initiating a repeated movement, enabling the AUV to return to the original route and complete its mission. This cycle may be repeated multiple times until the AUV safely resumes its original course.

This process of coordinate calculations yields various evasive maneuvers, two of which are depicted in Figure 7 and

Figure 8. Figure 7 illustrates a simple maneuver where the AUV, upon detecting an obstacle, maneuvers towards a point nearly perpendicular to the original route. Having reached this point, the AUV smoothly returns to the original route, completing the maneuver with a single evasive step. In

Figure 8, a more complex scenario unfolds, necessitating multiple evasive steps. Even after the first attempt to return to the original route, the obstacle persists in the AUV's path, prompting further steps. Consequently, the AUV moves a greater distance parallel to the obstacle before successfully reaching its original route, ultimately completing the entire evasive maneuver.

4. IMPLEMENTATION AND EVALUATION

This section introduces our architecture, which has been implemented to compute feasible AUV routes and simulate their movements. Additionally, the implementation and evaluation of the collision-avoidance mechanisms used are described in more detail.

4.1. Path Planning

This subsection presents architecture designed to plot viable AUV routes and subsequently simulate their trajectories. The solution used is evaluated through comprehensive case studies.

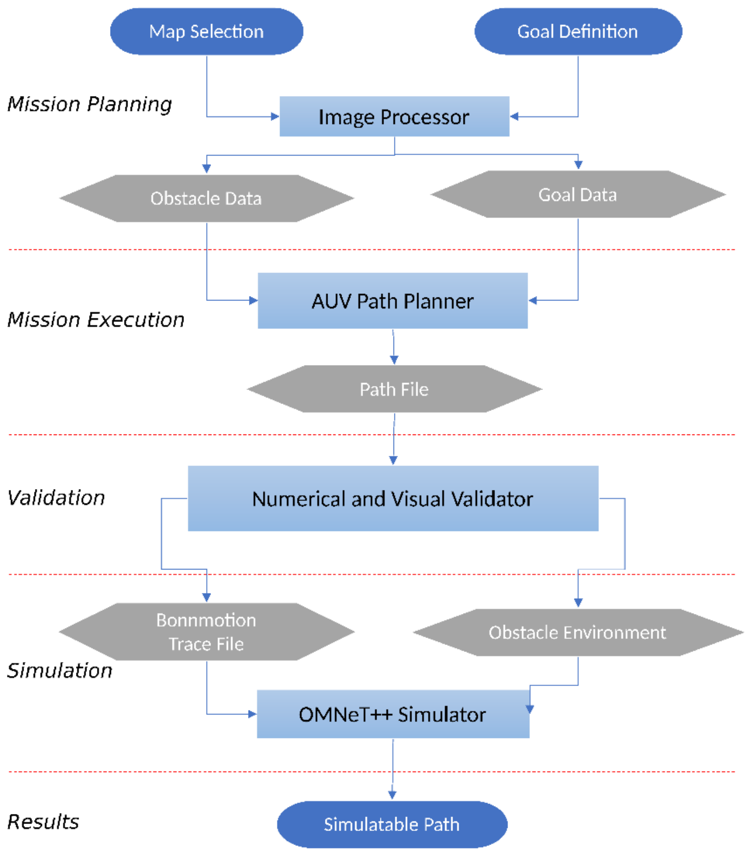


Figure 9. Representation of path planning architecture

4.1.1. Implemented Architecture

The architecture, as illustrated in

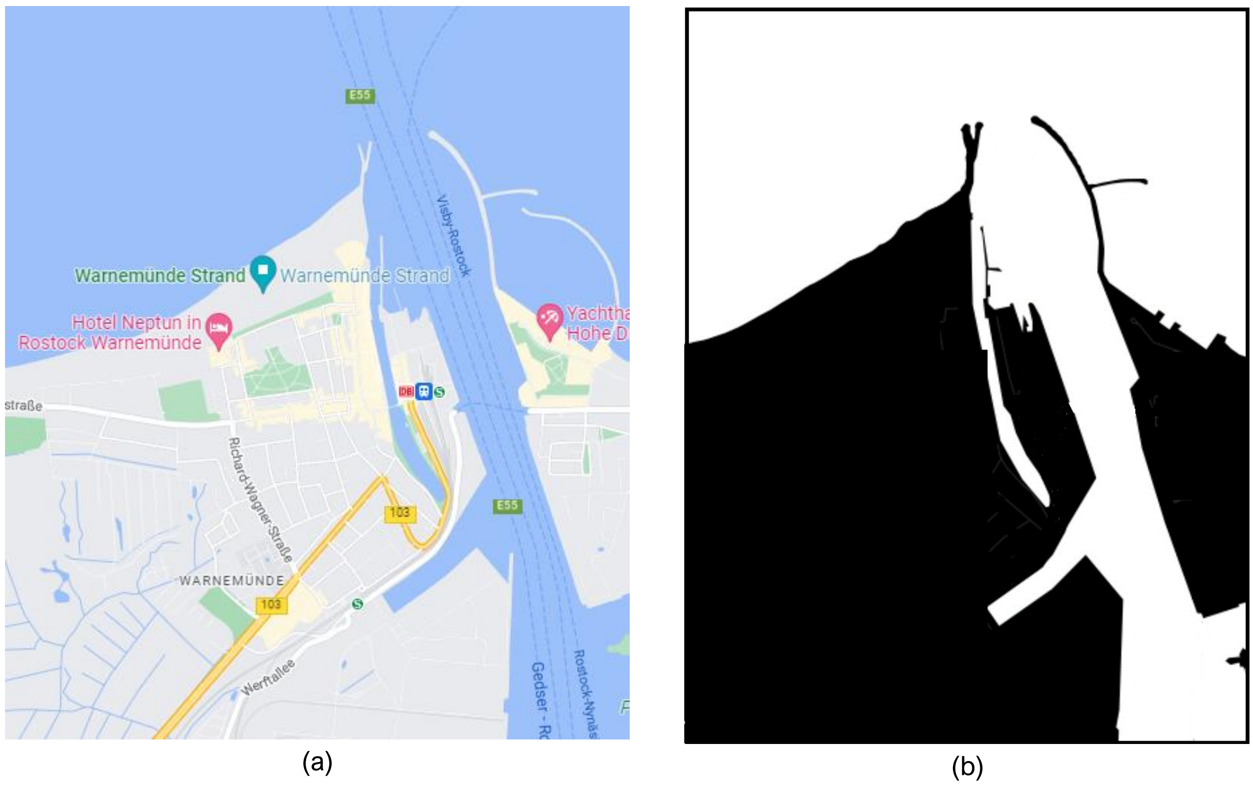


Figure 10. (a) Screenshot of Warnemünde, Germany, from Google Maps. (b) Warnemünde with land in black and water in white

Figure 9, illustrates the overarching objective to generate executable paths in line with AUV mission requirements. To this end, an intricate structure was devised, consisting of five distinct stages executed sequentially and iteratively if the desired outcome was not attained. The stages included mission planning, mission execution, validation, simulation, and results.

Mission Planning

The mission planning stage is pivotal, focusing on defining the primary objectives, location, and nature of the mission, be it a single AUV or multiple AUVs. An image processor is employed to convert the selected map and goals into obstacle and goal data, crucial inputs for subsequent path planning. The process involves converting a screenshot from Google Maps, such as depicted in Figure 10a), into a binary representation shown in Figure 10b). This processed map, with white representing water and black representing land obstacles, is subjected to various image processing steps, resulting in two txt files for obstacle and goal data.

Mission Execution

The core of the architecture is the mission execution stage, where the AUV planner utilizes algorithms based on obstacle and goal data to compute the main path. Using the procedures described in Section 3, this stage generates a txt file containing the planned path. Figure 11 visually depicts Warnemünde, Germany, with the planned path connecting the blue starting point and the green finishing point, calculated using the Dstar algorithm.

Validation

Validation is a critical step, divided into numerical and visual aspects. Numerical validation verifies key data points, such as starting and finishing points, and ensures that the length deviation from the optimal path falls within acceptable limits. Visual validation checks for collisions with obstacles or other AUVs. A successful validation results in the generation of files that describe the route as a Bonnmotion trace and an obstacle environment, which is essential for route simulation in OMNeT++.

Simulation

The simulation stage involves simulating the calculated route using the BonnMotion Mobility model, leveraging additional functionalities of the OMNeT++ simulator. OMNeT++ facilitates the addition of communication between multiple AUVs, crucial for cooperative scenarios. Given its discrete event simulator nature primarily designed for building network simulators (OMNeT++ Developer Team, 2023), OMNeT++ proved to be a rational choice for simulating AUV movements.

Results

Following the successful completion of the aforementioned stages, a satisfactory result indicates that a route is ready for real life AUV testing. In conclusion, our implemented architecture offers a comprehensive and systematic approach to AUV path planning and simulation, and is validated through case studies as follows.

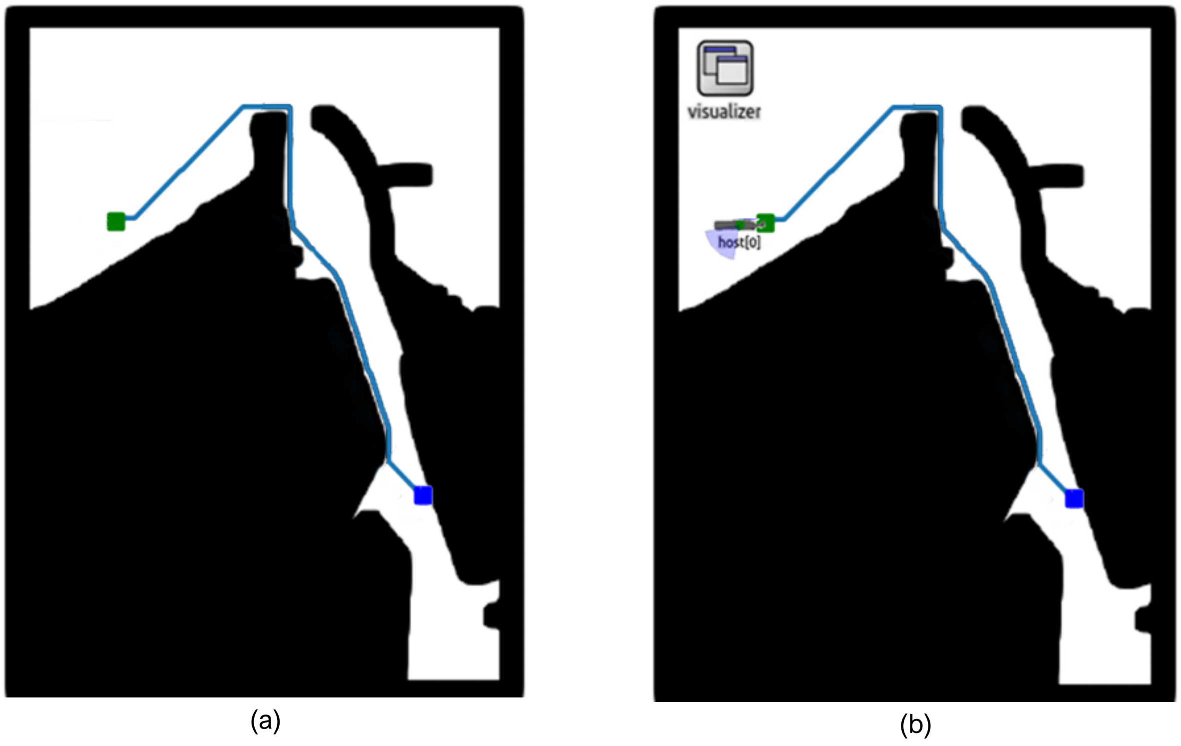


Figure 11. (a) Route at Warnemünde planned by Dstar from blue starting point to green finishing point, (b) Planned path simulated in OMNeT++

4.1.2. Use Cases

Following are case studies illustrating AUV path planning, featuring scenarios with both single and multiple AUVs. The decision to use one or more AUVs is contingent on the mission's nature and requirements. For instance, missions involving traversing from a starting to a finishing point while exploring the seafloor might be suitable for a single AUV. On the other hand, missions involving underwater exploration or mapping could benefit from deploying multiple AUVs in a coordinated manner to cover expansive areas of the ocean floor, thereby gathering comprehensive data on seafloor topography, geology, and marine biodiversity.

Single AUV

The initial case study focused on the main AUV objective, namely, navigating from one point to another. This offline planning process entails guiding the AUV from a specified starting point to an exact location while maneuvering to avoid any obstacles. The chosen mission location was Warnemünde, situated at the confluence of the Warnow River and the Baltic Sea (coordinates 54.18348, 12.09044). The AUV was tasked with navigating from the interior of the Warnow River channel (coordinates 54.1742, 12.09651) to a point in the open sea (coordinates 54.18322, 12.07709), simulating scenarios involving navigating through tight corners and irregular boundaries.

The physical environment included features such as a port, a river channel, and a vast open ocean, as depicted in Figure 10a. The map was processed by the image processor which converted it into a two-tone black-and-white image (Figure 10b). The Dstar algorithm, as detailed in Section 3, was used for path planning, and successfully generated the optimal route shown in Figure 11a. Validation ensures that the route adheres to

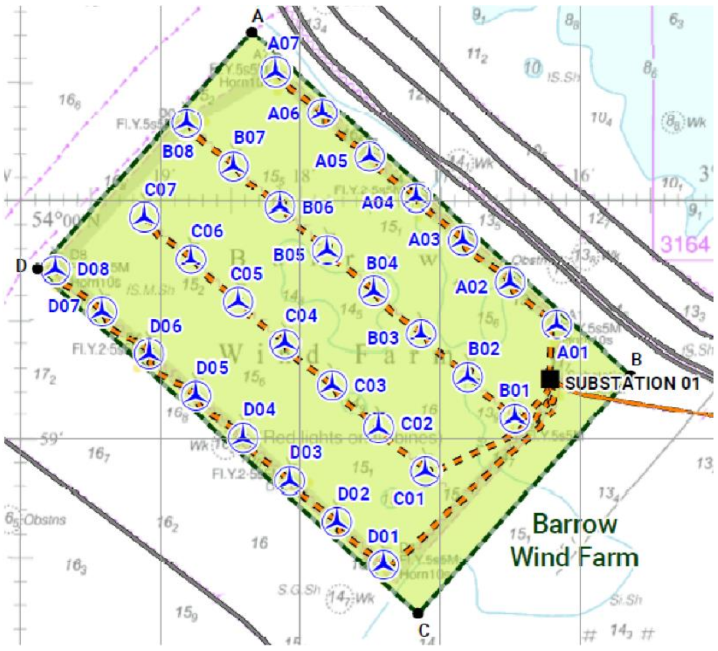


Figure 12. The Scottish Barrow Wind Farm with turbines and cabling positions

numerical correctness and visual criteria, paving the way for simulation in OMNeT++ (in Figure 11b), and demonstrating the efficiency of the architecture in single AUV missions.

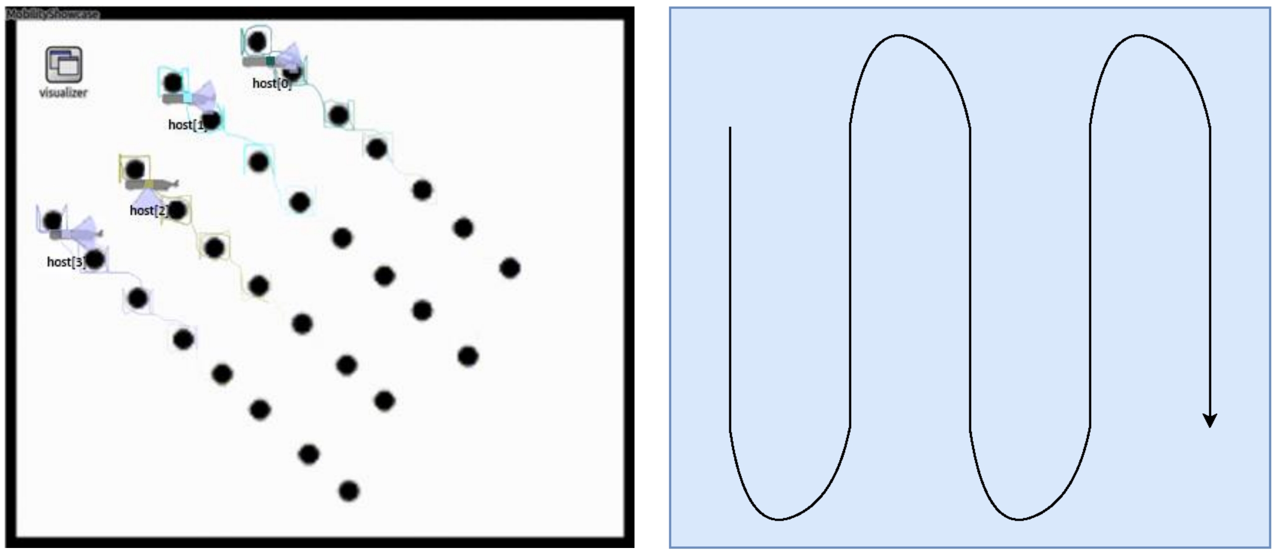
Multiple AUVs

The initial mission involved the deployment of multiple Autonomous Underwater Vehicles (AUVs) in a simulated scenario aimed at testing collision avoidance algorithms. In this setup, the AUVs were programmed to navigate to distinct targets strategically positioned to intersect each other's paths. This deliberate configuration created scenarios that facilitated the evaluation of collision avoidance algorithms, offering a valuable opportunity to assess and improve AUV performance when operating in close proximity to other vehicles.

The subsequent mission used four AUVs to perform a comprehensive site survey of individual wind turbines situated at the Barrow Wind Farm in the East Irish Sea. The primary objective of the mission was to highlight the potential of AUVs in analyzing the structural integrity of wind turbines, a task traditionally conducted manually using diving equipment for tasks such as maintenance and damage inspection. The integration of AUVs in this mission aimed to automate the wind turbine surveying process, and thus lower effort and increase overall efficiency. This mission serves as a demonstration of the capability to plan and execute intricate operations involving multiple AUVs.

Figure 12 shows an extract from a sea map, indicating the positions of each individual wind turbine. Notably, the image processor required supplementary manual assistance to generate the necessary black-and-white image for further processing. The goal definition involved a single starting point located at substation 01, assumed to be a safe spot for deploying AUVs into the water. As the finishing point for each AUV was not predetermined, all 30 wind turbines were designated as potential route endpoints.

The mission was executed using the CBS (Conflict-Based Search) method, due to its suitability for scenarios involving multiple AUVs. The requirement that AUVs cover multiple routes increased the mission's planning complexity. Offline planning was crucial for determining the subsequent steps of AUVs after surveying a wind turbine, accounting for potential casualties. The optimized sequence of execution for each AUV, as



(a) (b)

Figure 13. (a) Simulated inspection of wind farm turbines, (b) Planned mission route of the AUV

calculated by the route planner, dictates that each AUV surveys a distinct line of wind turbines (A to D in Figure 12). Additionally, four waypoints are designated around each wind turbine to ensure comprehensive inspection from all sides. The validation process was even more important in this mission due to the multitude of waypoints targeted by each AUV. Numerical validation, akin to previous missions, had to be conducted multiple times to account for the 30 wind turbines and the four waypoints around each turbine, resulting in a total of 120 sets of start and goal locations that require validation.

Figure 13a is a visual representation of the simulated inspection of the wind turbines at the Barrow Wind Farm, illustrating the AUVs immediately after completing the inspection of the last wind turbine. The sequential course of each AUV reveals the optimized sequence followed to reach its destination point.

4.2. Collision Avoidance

This section deals with the implementation and evaluation of collision-avoidance mechanisms.

4.2.1. Implementation

The implementation of AUV collision-avoidance mechanisms involves the utilization of OMNeT++, with motion simulation facilitated by the INET framework. Within OMNeT++, the network definition was distributed across two network description (ned) files, while an additional initialization (ini) file specified the motion model, environmental properties, and AUV characteristics.

Overall architecture was organized into three distinct modules: the navigation module, the logic submodule, and the collision-avoidance module. Notably, the navigation module was responsible for obstacle recognition and data distribution, while the logic submodule performed calculations crucial for determining the optimal direction for evasive maneuvers. The collision-avoidance module then computed the avoidance route, completing the tripartite structure of the implemented architecture.

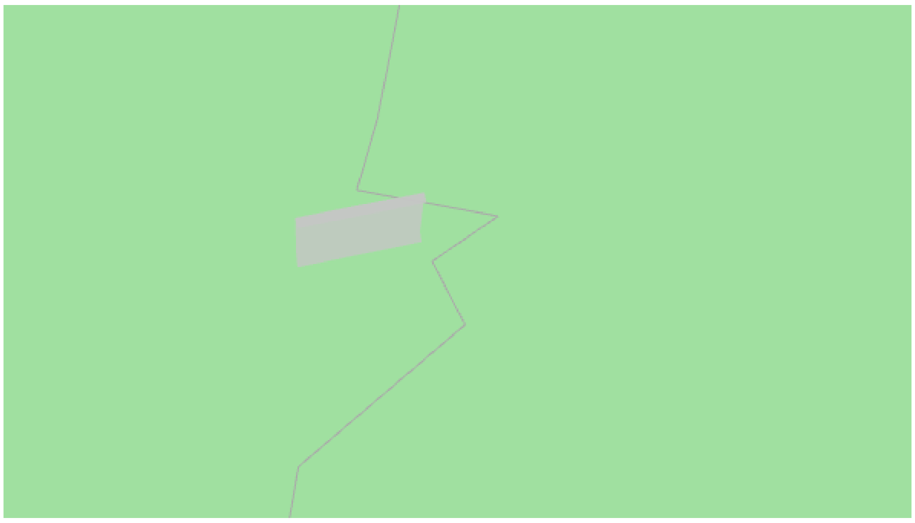


Figure 14. Screenshot of a standard maneuver from OMNeT++

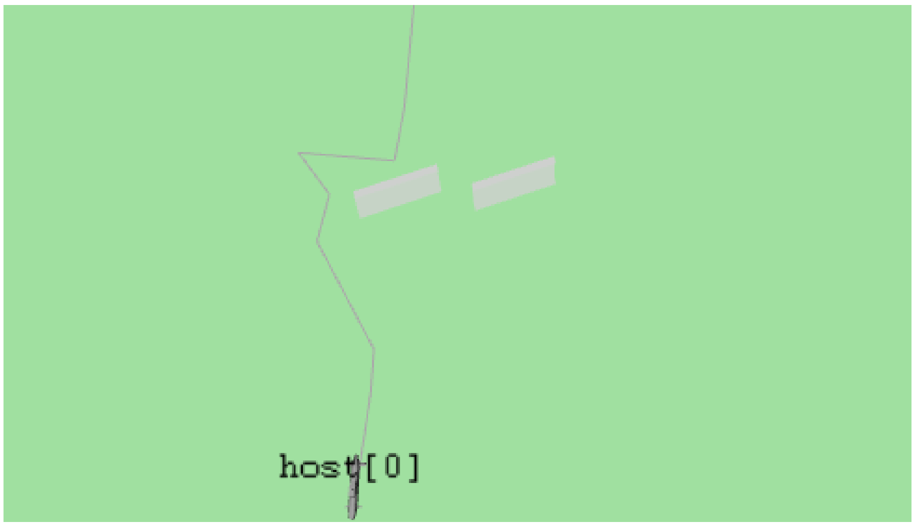


Figure 15. Screenshot of a maneuver with two objects from OMNeT++

4.2.2. Evaluation

Finally, the implemented simulation model was assessed to validate the functionality of the collision-avoidance concept, which involved conducting case studies and checking whether the observed outcomes aligned with anticipated results. Two distinct case studies were outlined, each featuring a predetermined mission route characterized by an undulating pattern, as depicted in Figure 13b.

In the first case study, a comprehensive scenario was envisaged to validate main functionalities. This involved a situation where a single object obstructed the AUV's mission route. Following obstacle detection, the AUV initiated avoidance route calculation, directing its path to the left of the object from its perspective. This maneuver led the AUV to deviate from its original route by nearly 90°. Subsequently, upon reaching the new avoidance point, the AUV reverted to its original route, necessitating an additional avoidance step due to obstacle proximity. The AUV consistently selected the optimal route, minimizing the distance to its outermost corner. A screenshot of this maneuver was given and evaluated in Figure 14, illustrating the AUV's approach, detection, and successful evasion in OMNeT++.

The second case study introduced a more complex scenario featuring multiple objects within AUV's field of vision. An object obstructed the mission route, and another object was positioned adjacent to it. The AUV, following detection, computed a new route to circumvent the obstacle, initially opting to pass on the left side. However, recognizing the proximity of the second object on that side, the AUV modified its decision and chose the longer route, passing the obstacle on the right side to avoid collision. The maneuver, akin to the first case study, was visualized and evaluated in Figure 15, demonstrating AUV's successful negotiation of the complex scenario in OMNeT++.

In both case studies, the AUV executed evasive maneuvers in accordance with expectations, showcasing the effectiveness of the implemented collision-avoidance mechanisms.

5. CONCLUSION

In conclusion, this paper explored the viability of AUV path planning, emphasizing two scenarios: single AUV path planning and path planning for multiple AUVs. In single AUV scenarios, the Dstar search algorithm efficiently calculated suitable paths within reasonable timeframes. By contrast, multiple AUV path planning involved a more complex approach, requiring the use of a conflict-based search algorithm to ensure simultaneous collision avoidance by multiple AUVs. Simulation in the OMNeT++ framework INET of planned paths underscores the usability of the proposed architecture, showcasing its effectiveness in different stages. Furthermore, the paper gives a significant contribution to the field by extending the simulation model to facilitate AUV obstacle detection and avoidance. The development of modular architecture lays the foundation for the implementation of various mechanisms, including obstacle detection, logic for decision-making, and consideration of safe distances and obstacle shapes. The simulation model was validated through case studies, demonstrating the AUV's capability to detect and navigate around obstacles that were unknown during mission planning.

REFERENCES

Aguiar, M. et al., 2019. Optimizing autonomous underwater vehicle routes with the aid of high resolution ocean models. *OCEANS 2019 MTS/IEEE SEATTLE*, pp. 1-7. Available at: https://doi.org/10.23919/oceans40490.2019.8962569

An, D. et al., 2023. Intelligent Path Planning Technologies of Underwater Vehicles: a Review. *J Intell Robot Syst*, 107(2). Available at: https://doi.org/10.1007/s10846-022-01794

Aschenbruck, N. et al., 2010. Bonnmotion: a mobility scenario generation and analysis tool. *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques. Available at: https://doi.org/10.4108/icst.simutools2010.8684*

Aschenbruck, N. et al., 2012. *Future Security: 7th Security Research Conference.* 318 ed. Bonn, Germany: Springer. Available at: https://doi.org/10.1007/978-3-642-33161-9

Cook, D., Vardy, A. and Lewis, R., 2014. A survey of AUV and robot simulators for multi-vehicle operations. *IEEE/OES Autonomous Underwater Vehicles (AUV)*, pp. 1-8. Available at: https://doi.org/10.1109/auv.2014.7054411

Engelhardtsen, Ø., 2007. *3D AUV Collision Avoidance,* Norwegian University of Science and Technology: Norwegian University of Science and Technology, Department of Engineering Cybernetics.

Flenker, T. and Stoppe, J., 2021. MARLIN: An IoT Sensor Network for Improving Maritime Situational Awareness. *MARESEC 2021*,

Han, G. et al., 2023. Early Warning Obstacle Avoidance-Enabled Path Planning for Multi-AUV-Based Maritime Transportation Systems. *IEEE Transactions on Intelligent Transportation Systems*, 2, pp. 2656-2667.

Hart, P. E., Nilsson, N. J. and Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, pp. 100-107.

Jaulin, L., 2001. Path planning using intervals and graphs. *Reliable computing*, pp. 1-15. Available at: https://doi.org/10.1023/a:1011400431065

Liu, R.-D. et al., 2019. Intelligent Path Planning for AUVs in Dynamic Environments: An EDA-Based Learning Fixed Height Histogram Approach. *IEEE Access*, pp. 185433-185446. Available at: https://doi.org/10.1109/access.2019.2960859

Mészáros, L., Varga, A. and Kirsche, M., 2019. Inet framework. *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*, 21 May, pp. 55-106. Available at: https://doi.org/10.1007/978-3-030-12842-5_2

OMNeT++ Developer Team, 2023. *BonnMotionMobility.* Available at: https://doc.omnetpp.org/inet/api-current/neddoc/inet.mobility.single.BonnMotionMobility.html.

Sharon, G. et al., 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence 219*, pp. 40-66. Available at: https://doi.org/10.1016/j.artint.2014.11.006

Shen, Z., Wilson, J. P. and Gupta, S., 2019. *An Online Coverage Path Planning Algorithm for Curvature-Constrained AUVs.* Seattle, WA, USA, s.n., pp. 1-5. Available at: https://doi.org/10.23919/oceans40490.2019.8962629

Stentz, A., 1994. Optimal and efficient path planning for partially-known environments. *1994 IEEE international conference on robotics and automation*, pp. 3310-3317. Available at: https://doi.org/10.1109/robot.1994.351061

Toma, A.-I. et al., 2021. Pathbench: A benchmarking platform for classical and learned path planning algorithms. *18th Conference on Robots and Vision (CRV)*, pp. 79-86. Available at: https://doi.org/10.1109/crv52889.2021.00019

Varga, A., 2010. OMNeT++. In: K. Wehrle, M. Günes & J. Gross, eds. *Modeling and Tools for Network Simulation.* Berlin, Heidelberg, Germany: Springer, p. 35–59. Available at: https://doi.org/10.1007/978-3-642-12331-3_3

Wei, D. et al., 2022. A Hyperheuristic Algorithm Based on Evolutionary Strategy for Complex Mission Planning of AUVs in Marine Environment. *IEEE Journal of Oceanic Engineering*, 47(4), pp. 936–949. Available at: https://doi.org/10.1109/joe.2022.3177858

ToMS WebFirst