Mission Planning and Motion Models for Autonomous Underwater Vehicles in OMNeT++

Willi Brekenfelder, Peter Danielis, Gunnar Kulat, Mohammad Salouh,
Finn Ole Stadtaus, Lara Tauch, Helge Parzyjegla
Institute of Computer Science, University of Rostock
Rostock, Germany
{willi.brekenfelder, peter.danielis, helge.parzyjegla}@uni-rostock.de

***Abstract*—Autonomous Underwater Vehicles (AUVs) are well suited to perform underwater work that is too dangerous or inaccessible for humans. In order to better plan, test, and optimize AUV missions, simulations with adequate models are essential. In this paper, we review existing motion models for the OMNeT++ simulation environment. We determine their ability to store and navigate to the waypoints of a planned mission. Based on the review results, we present a tool and workflow (i) to extract waypoint data from AUV mission plans created in the Neptus software, (ii) to check the planned trajectories for plausibility, and (iii) to convert the data into the appropriate formats for selected motion models. Finally, we demonstrate and evaluate our tool and workflow in a case study.**

I. Introduction

Autonomous Underwater Vehicles (AUVs) are unmanned, self-propelled vehicles that are well suited for exploring and developing underwater areas. AUVs can be equipped with different kinds of tools, instruments, and sensors (e. g., sonar systems, video cameras, CTD profilers, Doppler current meters) enabling them to complete various tasks and missions. Once deployed, usually from a surface vessel, an AUV executes its tasks independent of direct human control and without a cable connection to its mother ship. This way, it can safely operate in areas that are too dangerous for manned submersibles or inaccessible for Remotely Operated Vehicles (ROVs), e. g., AUVs can operate under large ice sheets in polar regions, can inspect small underwater caves or man-made structure, or can safely search ocean floors for unexploded ordnance.

Because of the non-existent permanent connection to the mother ship, an AUV has to fulfill additional requirements. On the one hand, an AUV has to follow a planned course moving in a predetermined way in order to safely navigate to its operation area as well as back again to the mother ship for pick up. On the other hand, it has to react independently to unforeseen circumstances and difficulties such as an obstacle blocking the planned course. Obstacles need to be autonomously avoided in order to prevent collisions. Furthermore, in critical situations, the AUV needs always to be able to cancel a mission and return to the mother ship, for example, if it runs low on energy.

The AUV's ability of adapting tasks whenever necessary makes the planning of an AUV mission a non-trivial endeavor [1]. To support AUV operators in this planning phase, simulations of the AUV's mission are a very valuable tool. Simulations enable operators to better study the vehicle's behavior in specific or even dangerous situations without putting the actual AUV at any risk. In addition, alternate mission plans can be analyzed and compared or even optimized for certain objectives, e. g., to set waypoints that maximize the search area covered by the AUV for a given energy budget. Furthermore, AUV engineers and researchers also benefit greatly from simulations. Simulations can help to develop and investigate search strategies or cooperation schemes in detail. In particular, for the latter, movement patterns and communication protocols need to be coordinated and synchronized between multiple AUVs on cooperative missions. This is especially important since underwater communication by acoustic modems tends to be costly, slow, and error-prone.

In this paper, we present a toolchain enabling AUV operators, engineers, and researchers to extract the movement data of AUVs as planned in the Neptus command and control software [2] and to simulate these routes and trajectories in the OMNeT++ simulation environment [3]. The key component of the toolchain is a conversion program that parses a Neptus mission plan, extracts planned waypoints and routes, checks those for physical plausibility w.r.t. the AUV's maneuvering capabilities, and creates a corresponding movement trajectory as input for the simulation models. For the latter, our tool supports the motion models `BonnMotionMobility` and `TurtleMobility` that are part of the INET framework [4] in OMNeT++. Hence, our toolchain offers a convenient way to make real-world AUV missions available for detailed simulations and a further analysis in OMNeT++.

The remainder of the paper is structured as follows: Section II introduces the Neptus software showing how to create and export AUV mission plans. In Sect. III, an overview about the OMNeT++ simulator is given and existing motion models are briefly reviewed that are shipped together with the simulation framework. In this process, two suitable mobility models for AUV missions are identified and further characterized. Section IV presents our conversion tool, explains its features, and demonstrates the whole workflow leveraging our toolchain in a case study. Finally, Sect. V concludes the paper.

II. Mission Planning in Neptus

Neptus [2] is a command and control software for unmanned vehicles developed by the Underwater System and Technology

Table I
COMMON MANEUVERS SUPPORTED BY NEPTUS.

Maneuver	Description
Goto	Navigate towards a waypoint.
Loiter	(Circular) holding pattern keeping the AUV in a specific area.
Station Keeping	Keep the AUV within an area without enforcing a holding pattern.
Rows	Lawn mower pattern for surveying an area.
RI Pattern	Reacquire-Identify (RI) search pattern composed of three rows patterns.
Cross Hatch Pattern	Another search pattern composed of two rows patterns.
Follow Path/Trajectory	Follow a given path (with time constraints).
Elevator	Change depth while staying within an area.

Laboratory (LSTS) at the University of Porto. It supports AUV operators throughout the whole mission lifetime including the planning of the mission, monitoring and controlling the AUV during mission execution, and analyzing the mission logs and results afterwards [5]. Furthermore, Neptus is an integral part of the LSTS open source software toolchain [6] that additionally features a communication protocol [7] and API [8] as well as a development framework for the AUV's on-board software [9]. This software stack has matured over the years and is successfully deployed on various AUVs sent on real-world missions [10]–[12]. However, in this paper, we solely focus on the planning phase. Our objective is to use such realistic mission plans created with the Neptus software as the basis for simulations in the OMNeT++ framework.

There are several approaches to create an AUV mission plan [13]. Neptus mission plans are based on *primitives* which are commands and/or tasks the AUV can execute autonomously, e. g., operating a sensor or navigating towards a waypoint. Basic motion primitives are called *maneuvers* [13]. Table I gives an overview about common maneuvers that can be planned with Neptus. More complex maneuvers are build from simpler ones, e. g., all search patterns to cover a specific area are eventually determined by a series of waypoints to visit. Please note that, depending on its autonomous capabilities, the AUV may not support all of the listed maneuvers. Neptus then enables human AUV operators to directly specify the mission plan by arranging and composing these maneuvers in a convenient Graphical User Interface (GUI).

Figure 1 shows a screenshot of the mission planning view in Neptus. The major part of the window (top left) is occupied by the map panel. The map shows the river Warnow in Rostock at the city harbor. On this map, the planned course of the AUV is displayed as an overlay. As an example, we have planned a small mission to demonstrate some of the maneuvers above. From its deployment zone, the AUV is sent by Goto maneuvers to two waypoints to the left in order to reach the area to be surveyed. For surveying, we use the Rows maneuver resulting in a course of five parallel lines (called rows) by which the area is covered. Finally, the AUV travels to three more waypoints in order to return to its starting zone for pick up. The bottom

Table II
COMMON MOBILITY MODELS PROVIDED BY OMNET++/INET.

Mobility Model	Description
Stationary	Stationary deterministic or random placement.
StaticGrid	Stationary placement in a rectangular grid.
StaticConcentric	Stationary placement in concentric circles.
Linear	Linear movement with constant speed.
Circle	Circular movement with constant speed.
Rectangle	Rectangular movement with constant speed.
Tractor	Movement according to a rows pattern.
Vehicle	Curves when turning, constant speed.
Turtle	Programmable, linear movement.
Facing	Stationary, but orients towards the position of another mobility model.
BonnMotion	Replays BonnMotion trace files.
Ns2Motion	Replays ns2 trace files.
Ansim	Replays ANSim trace in XML.
RandomWaypoint	Moves to/with random waypoint/speed.
GaussMarkov	Variable randomness up to Brown motion.
Mass	Random movement of a mass (physical laws).
Chiang	Uses probabilistic transition matrix.
Superpositioning	Combines several other mobility models.
Attached	Static offset from another mobility model.

panel lists each maneuver of the mission plan, whereas the panel right of the map allows to edit further parameters of the selected maneuver, e. g., the speed with which the AUV navigates towards a waypoint. The rightmost panel contains the mission tree that stores the mission elements: the home reference, defined beacons, and created plans. In Neptus, a mission may have multiple plans for one or even more AUVs operating in parallel.

Furthermore, Neptus allows to simulate the execution of a mission plan. For this purpose, Neptus computes and visualizes the current position of the AUV on the map in fast motion. In particular, the simulation considers the maneuvering capabilities of the AUV that are defined by the type of vessel deployed. Figure 2 shows a section of the map zoomed in towards the top right corner of the rows pattern. The simulated course, depicted by green dots, deviates from the planned path, represented as yellow line, since the AUV does not immediately turn by 90°. Unfortunately, the more realistic, simulated trajectory cannot be exported from Neptus. Instead, it is only possible to save the created mission plan in a `.mis` file containing all maneuvers and their original waypoints.

III. MOBILITY MODELS IN OMNET++

OMNeT++ [3] is an extensible, component-based C++ framework for discrete event simulations. It is well-proven for simulating communication networks and networked systems. It comes with its own Integrated Development Environment (IDE) based on the Eclipse platform [14] that is well suited for developing own simulation models [15]. For this purpose, OMNeT++ also features the Network Topology Description (NED) language enabling developers to easily combine and compose new components from existing building blocks and

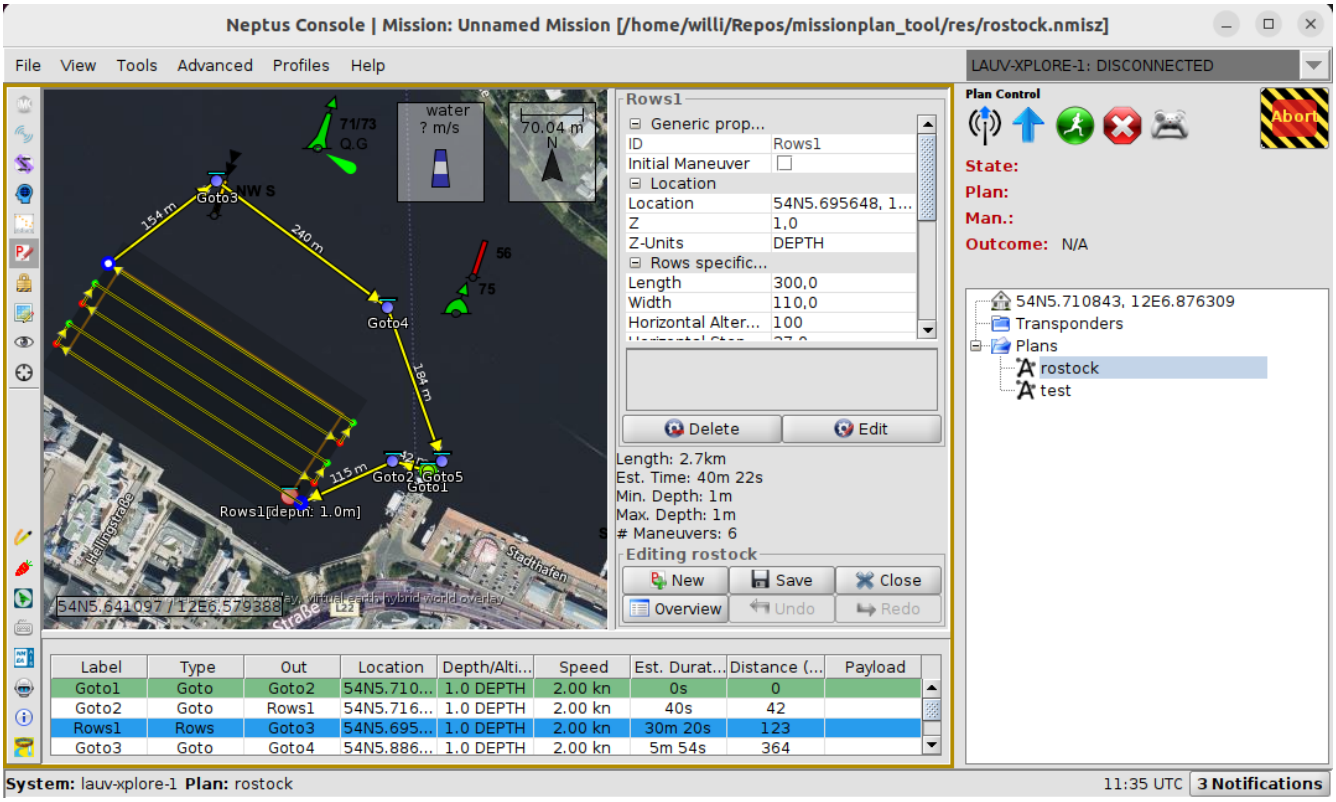

Figure 1. Neptus mission planning view with example route in the city harbor of Rostock.

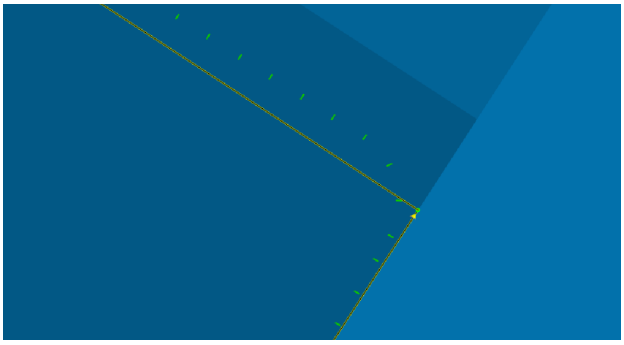

Figure 2. Planned path (yellow solid line) vs. simulated course (dotted green line) in Neptus.

new modules. The graphic capabilities of the IDE, including 2D and 3D animations, allow to visualize and even explore the state of running simulations and, thus, help new users and non-simulation experts to better familiarize with the simulation framework and the interworking of its components.

In addition, many well-tested simulation models are already available for OMNeT++. The INET framework [4] is one of the oldest and largest collections of simulation models covering the Internet protocol stack as well as closely related protocols such as Ethernet and Wireless LAN. Over the years, it thus became a natural choice as basis for many more specific simulation models (e. g., SimuLTE, Veins) [15]. Likewise, we also build our AUV (communication) models on top of OMNeT++/INET.

Table II gives an overview about mobility models that are already available in INET [16]. In particular, we are interested in those models in which it is basically possible to specify the path of an AUV as planned by Neptus. For this purpose, the mobility model has to offer the notion of a waypoint or a way to somehow emulate it. Furthermore, it needs to be deterministic in order to follow a planned path. After reviewing the mobility models, two suitable candidates remained, i. e., `BonnMotionMobility` and `TurtleMobility` that are discussed next.

A. *BonnMotion Mobility Model*

BonnMotion [17] is a mobility scenario generation and analysis tool developed at the University of Bonn, Germany, in order to study mobile multi-hop networks for disaster areas. However, the `BonnMotionMobility` model in OMNeT++ does not re-implement any of the original logic. Instead, it is able to parse trace files generated by the external BonnMotion tool. Compared to the other trace-based mobility models in INET (i. e., `Ns2MotionMobility` and `AnsimMobility`) that provide similar functionality, BonnMotion uses a much simpler file format. In fact, it is a plain text file in which each line contains all waypoints of a mobile actor (e. g., an AUV). Each waypoint is defined by four values "$t\ x\ y\ z$" where t specifies the time when the actor has to be at the position (x, y, z) in the simulation's coordinate system. The segments

```
1 <movement>
2  <set x="10" y="10" z="0" speed="2"/>
3  <forward d="60"/>
4  <wait t="10"/>
5  <set speed="1"/>
6  <turn heading="90" elevation="-10"/>
7  <forward t="40"/>
8  <moveto x="10" y="10" z="0" t="80"/>
9 </movement>
```

Listing 1. Example of a `TurtleMobility` program.

between waypoints are then interpolated by straight lines that are travelled with constant speed.

B. Turtle Mobility Model

The turtle is probably the most well-known feature of the educational programming language Logo [18]. By controlling the movement of the Logo turtle that is equipped with a retractable pen, one can produce line graphics. Likewise, using similar movement commands, the `TurtleMobilityModel` in INET allows to specify the motion of an actor. By executing a corresponding turtle program, we can thus make a simulated AUV follow a preplanned path in OMNeT++.

Listing 1 shows a program example written in the XML-based dialect used by the `TurtleMobilityModel`. The first command `set` (line 2) specifies the initial position using the coordinates x, y, and z. Additionally, the initial *speed* is defined too. The next command `forward` (line 3) makes the actor travel in the direction it is currently facing. It stops after the distance d is covered. Alternatively, it can stop after a time interval t (line 7). A `wait` command (line 4) is required to let the actor stay at its current position for the time t, whereas another `set` command is necessary to adjust the *speed* (line 5). The command `turn` (line 6) makes the actor turn by an angle of *heading* clockwise within the horizontal plane and by *elevation* vertically. Finally, the command `moveto` (line 8) makes the actor face and move towards the position (x, y, z). If the optional parameter t is given, the current speed is adjusted so that the actor arrives at the specified position after time t.

IV. Conversion Tool and Toolchain

Simulating AUV missions before putting them into action has many advantages. In particular, alternative mission plans, different mission parameters, varying environmental conditions, etc. can be studied safely beforehand without putting the AUV at any risk. Based on the simulation results, the AUV operator can then decide on the best option. Our workflow for simulating AUV missions has the following three steps:

1) *Mission planning.* The mission for the AUV is planned in Neptus and exported into a `.mis` file. It is important that, for mission planning, the same software tools are used independent of whether the mission is simulated or carried out in reality.
2) *Converting the mission plan.* From the saved mission file, input for the simulation's mobility models has to be generated. For `BonnMotionMobility`, a `.movements` file with waypoints is created. For `TurtleMobility`, a `.xml` file with movement commands is produced.
3) *Mission simulation.* The generated mobility files are used as input for the corresponding mobility models in an OMNeT++ simulation. Depending on the simulation's objectives, the degree of flexibility required, and personal preferences, we leave it to the user (e. g., researcher, AUV operator) which mobility model to use.

Sect. II and Sect. III already discuss the planning and simulation, respectively. Challenges and details of the intermediate conversion step, that glues the former together, are covered in the following. Afterwards, we demonstrate the developed conversion tool and the toolchain in a small case study.

A. Conversion Tool

Besides the mechanical, yet tedious parsing and generation of different file formats, the conversion of a planned course of an AUV into a motion description for simulation purposes include some challenges that are not obvious at first glance.

Geographic coordinates vs. Cartesian coordinates. Mission plans created by Neptus use geographic coordinates, whereas many simulation environments such as OMNeT++/INET work with Cartesian coordinates. In fact, the coordinate system is not a property of the simulation framework, it is rather an assumption of the simulation models that are actually used. Nevertheless, we decided to translate the waypoint coordinates of the mission plan rather than adapting the simulation models. For this purpose, we calculate the distances between Neptus waypoints according to [19]. Choosing a waypoint as a reference, we are then able to transform the other waypoints into Cartesian coordinates.

Trace-based mobility models. Trace-based models are well suited to replay an observed activity. When recording the movements of an actor, it is a natural choice to store the current time and position either periodically or on an event-driven basis, e. g., whenever a waypoint is reached. However, an AUV mission plan does not have timestamps yet. Instead, the plan sends the AUV towards the next waypoint with a given speed. But knowing the distance to the next waypoint allows to estimate the time of arrival. Please note that this is only an estimation (although usually quite exact) because the AUV may deviate slightly from the planned path (e. g., when turning) or the specified speed (e. g., when accelerating).

Absolute waypoints vs. relative waypoints. Most mobility models specify a waypoint using absolute coordinates. However, it is also possible to define the next waypoint by an offset relative to the current one. Interestingly, `TurtleMobility` offers both specification variants. With the help of the `moveto` command, the AUV is sent to absolute coordinates, whereas a sequence of a `turn` and a `forward` command defines the next waypoint relative to the current position. We implemented both variants that, except for numerical errors, lead to identical results. The relative variant might be better suited when extending the AUV's autonomous behavior in future simulation models so that it can leave its planned path (e. g., for collision avoidance).

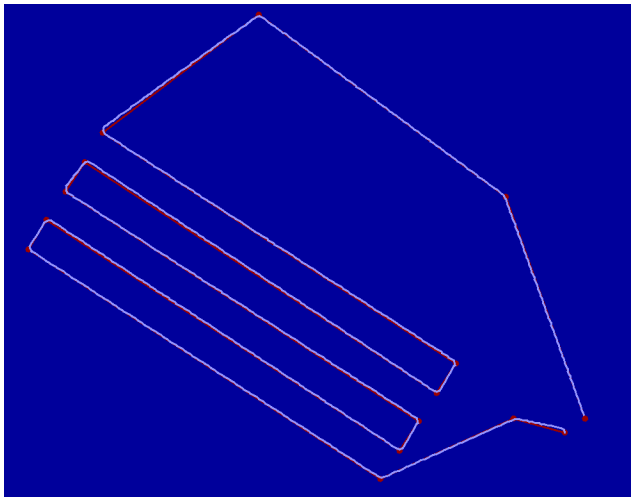

Figure 3. Mission plan for rostock habour in developed tool.

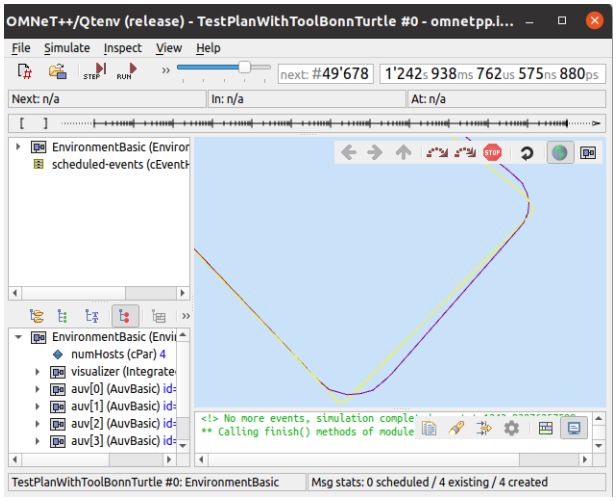

Figure 4. Mission plan for rostock habour in simulated in OMNeT++.

Turning at waypoints. Neither Neptus' mission plans nor the selected mobility models support curves. Hence, the turning behavior at waypoints (i. e., immediately moving towards next waypoint in a straight line) is not realistic at all. Neptus solves this issue via an internal simulation when visualizing the planned mission. We take the same approach by pre-simulating the AUV's path trajectory in the developed tool. Please note that we take kinetic constraints as well as constraints related to the AUV's design into account. For example, there is a margin defining when the AUV considers a waypoint to be reached and starts turning. Furthermore, the maximal turning velocity as well as the speed with which it can dive down or up are limited. All constraints are customizable for the type of AUV used. By comparing the computed trajectory with the planned path, we can warn if the deviation exceeds a certain threshold indicating that the mission plan is at least unsuited for the particular AUV. Checking the mission plan w.r.t. feasibility is thus a side effect of our pre-simulation. Finally, we interpolate the curve segments in the path trajectory by small line segments as required by the mobility models used in our OMNeT++ simulation.

B. Case Study

To demonstrate our tool and the toolchain, we simulate the example mission that we planned for an AUV in the Rostock city harbor in Sect. II. After finishing the work in Neptus and saving the created mission plan, the file can seamlessly be processed by our conversion tool. The tool parses the mission plan, transforms waypoint coordinates, checks the plan's feasibility by pre-simulating the AUV's path trajectory, interpolates resulting curve segments with lines, and outputs the final result either as a trace-based `.movement` file for `BonnMotionMobility` or as a `.xml` file with turtle program for `TurtleMobility`.

Figure 3 shows the originally planned path by Neptus (red line) as well as the pre-simulated trajectory (light blue line) in our conversion tool. First, both lines red and light blue look similar to what has originally been planned in Neptus (cf. Fig. 1). Second, the simulated light blue trajectory follows very closely the planned red path. Third, exceptions are only near waypoints where the computed trajectory shows a more realistic turning behavior.

The files, generated by our conversion tool, can then be used in simulations without further modifications. They describe the movement of an AUV based on a realistically planned mission. Moreover, no special mobility models are needed since both `BonnMotionMobility` and `TurtleMobility` are automatically shipped with any OMNeT++/INET version.

Figure 4 shows the OMNeT++ GUI during a simulation. The resulting trajectory from our motion models is displayed as purple line, whereas the originally planned path is drawn as yellow line for comparison. The main visualization panel is zoomed to two waypoints of the rows maneuver in order to better exemplify the smooth interpolation of the trajectory when the AUV starts turning towards the next waypoint.

V. Conclusions

In this paper, we developed a conversion tool and presented a toolchain to ease the simulation of planned AUV missions. The developed tool parses the mission plan of an AUV mission created in the Neptus software. It extracts waypoints of planned maneuvers and generates corresponding input files for the OMNeT++ motion models `BonnMotionMobility` and `TurtleMobility` that are shipped with the INET standard library. Furthermore, it pre-simulates the AUVs path trajectory, checks it for feasibility, and interpolates curve segments as needed. The presented workflow better integrates the mission planning and simulation of AUV missions, thus, supporting AUV operators and researchers alike. For future work, we want to also extract more data from Neptus files, e. g., when which sensors are used, and make these information available in OMNeT++ simulations. These will help us to better integrate the individual models within the simulation.

References

[1] M. P. Brito, R. Lewis, N. Bose, P. Alexander, G. Griffiths, and J. Ferguson, "The role of adaptive mission planning and control in persistent autonomous underwater vehicles presence," in *Proceedings of the 2012 IEEE/OES Symposium on Autonomous Underwater Vehicles (AUV)*. Southampton, UK: IEEE, Sep. 2012, pp. 1–9.

[2] LSTS. (2022, Jun.) Neptus source code repository. [Online]. Available: https://github.com/LSTS/neptus

[3] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (SIMUTools '08)*. Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Mar. 2008. [Online]. Available: https://doc.omnetpp.org/workshop2008/omnetpp40-paper.pdf

[4] L. Mészáros, A. Varga, and M. Kirsche, "INET framework," in *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*, A. Virdis and M. Kirsche, Eds. Cham, Switzerland: Springer, 2019, ch. 2, pp. 55–106.

[5] P. S. Dias, G. M. Gonçalves, R. M. F. Gomes, J. B. Sousa, J. Pinto, and F. L. Pereira, "Mission planning and specification in the Neptus framework," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA 2006)*. Orlando, FL, USA: IEEE, May 2006, pp. 3220–3225.

[6] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, and J. Sousa, "The LSTS toolchain for networked vehicle systems," in *Proceedings of 2013 MTS/IEEE OCEANS – Bergen*. Bergen, Norway: IEEE, Jun. 2013, pp. 1–9.

[7] R. Martins, P. S. Dias, E. R. B. Marques, J. Pinto, J. B. Sousa, and F. L. Pereira, "IMC: A communication protocol for networked vehicles and sensors," in *Proceedings of OCEANS 2009 – Europe*. Bremen, Germany: IEEE, May 2009, pp. 1–6.

[8] LSTS. (2022, Jun.) IMC source code repository. [Online]. Available: https://github.com/LSTS/imc

[9] ——. (2022, Jun.) DUNE source code repository. [Online]. Available: https://github.com/LSTS/dune

[10] L. Madureira, A. Sousa, J. Sousa, and G. Gonçalves, "Low cost autonomous underwater vehicles for new concepts of coastal field studies," *Journal of Coastal Research*, vol. 1, pp. 238–242, Apr. 2009, Special Issue No. 56. Proceedings of the 10th International Coastal Symposium (ICS 2009).

[11] L. Madureira, A. Sousa, J. Braga, P. Calado, P. Dias, R. Martins, J. Pinto, and J. Sousa, "The light autonomous underwater vehicle: Evolutions and networking," in *Proceedings of 2013 MTS/IEEE OCEANS – Bergen*. Bergen, Norway: IEEE, Jun. 2013, pp. 1–6.

[12] M. Costa, J. Pinto, M. Ribeiro, K. Lima, A. Monteiro, P. Kowalczyk, and J. Sousa, "Underwater archaeology with light auvs," in *Proceedings of OCEANS 2019 – Marseille*. Marseille, France: IEEE, Jun. 2019, pp. 1–6.

[13] M. Kothari, J. Pinto, V. S. Prabhu, P. Ribeiro, J. B. de Sousa, and P. Sujit, "Robust mission planning for underwater applications: Issues and challenges," *IFAC Proceedings Volumes*, vol. 45, no. 5, pp. 223–229, Apr. 2012, proceedings of the 3rd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles.

[14] Eclipse Foundation. (2022, Jun.) Eclipse platform overview. [Online]. Available: https://www.eclipse.org/eclipse/eclipse-charter.php

[15] A. Varga, "A practical introduction to the omnet++ simulation framework," in *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*, A. Virdis and M. Kirsche, Eds. Cham, Switzerland: Springer, 2019, ch. 1, pp. 3–51.

[16] INET Developers. (2022, Jun.) INET user's guide – INET 4.4.0 documentation. [Online]. Available: https://inet.omnetpp.org/docs/users-guide

[17] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "BonnMotion: A mobility scenario generation and analysis tool," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques (SIMUTools '10)*. Torremolinos, Malaga, Spain: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Mar. 2010.

[18] B. Harvey, *Computer Science Logo Style, Volume 1: Symbolic Computing*, 2nd ed. Cambridge, MA, USA: MIT Press, 1997.

[19] N. de Lange, *Geoinformatik*. Berlin / Heidelberg, Germany: Springer, 2013.