Path Planning Algorithms For Autonomous Underwater Vehicles

Pranavkumar Ghoghari*, Willi Brekenfelder*, Helge Parzyjegla*, Frank Sill Torres† and Peter Danielis*
*Institute of Computer Science, University of Rostock, Rostock, Germany
Email: {willi.brekenfelder;helge.parzyjegla;peter.danielis}@uni-rostock.de
†German Aerospace Center (DLR),
Institute for Protection of Maritime Infrastructures, Bremerhaven, Germany
Email: frank.silltorres@dlr.de

***Abstract*—The increasing use of autonomous underwater vehicles (AUVs) for environmental monitoring, underwater research, and search and rescue operations highlights the importance of efficient path planning for AUVs in unknown areas. In this paper, we propose a path planning and simulation strategy for AUVs. Our approach involves path planning algorithms and the OMNeT++ network simulator to simulate the motion of AUVs. The algorithms are applied to mission planning for a single AUV as well as for multiple AUVs in specific scenarios. The results demonstrate the potential of the approach to solve real-world problems and provide insightful information about the behavior of AUVs in different situations.**

***Index Terms*—AUVs, path planning, simulation.**

I. Introduction

Most of the earth is covered by bodies of water, and much of that water has yet to be discovered. The reason for the shift in focus from land to water is not only in the energy sector, but also has a major impact on it. The oceans hold an immense amount of renewable energy in the form of wind energy, tidal energy and wave energy. To harness this vast amount of hidden energy, the ocean must first be surveyed. The emerging technology of autonomous underwater vehicles (AUVs) may be perfect for the goal of surveying the ocean. AUVs are robots capable of moving underwater without human assistance to perform tasks. Depending on their purpose, AUVs are equipped with a variety of sensors and actuators to assist them in their mission [1].

In recent years, many different fields have shown interest in AUV research. This paper also contributes to this interest and attempts to solve the task of path planning for AUVs and simulate their movement while following it. The field of simulation has expanded significantly due to technological development and increasing computing capacity. The capabilities of simulation have made it possible to test unexpected situations and events. The spectrum ranges from transportation and aerodynamics to more sophisticated disciplines such as biology. The need for a simulation environment for mission planning for AUVs is imminent. Many researchers have been working on this issue and several projects have been developed [2]. Since OMNeT++ (Objective Modular Network Testbed in C++) can be used to simulate movement and is well suited for lather testing of different communication methods, it was chosen as simulation environment [3].

The main contribution of this paper is to plan the path of a single AUV as well as paths for multiple AUVs for a specific type of mission with suitable algorithms as well as to explore the possibilities of setting up a simulation platform in OMNeT++. The path-planning for single or multiple AUVs is at least as important as simulating the motion, since no simulation would be possible without a predetermined path. The motion of AUVs is simulated using the BonnMotionMobility model in the OMNeT++ framework INET [4]. The BonnMotionMobility model in INET is trace-based. Trace-based means the trajectories of the nodes are taken from generated trace files, which include triple consisting of time as well as x-coordinate and y-coordinate of the AUV at the given time. The speed of the AUV is therefore implicitly defined by the distance between two consecutive coordinates and the elapsed time between being at these coordinates.

II. Path Planning Algorithms

This section aims to explain the algorithms used for path planning and is divided into two parts: path planning for a single AUV and path planning for multiple AUVs. A basic maneuver every underwater vehicle needs to perform is to go from one point to another. The complexity in path planning increases after the introduction of an obstacle. This increase in complexity is due to the decision of the path to be chosen to reach the goal while avoiding the obstacle. To find the optimum path in a complex environment, path planning algorithms are used. Motion planning (also called path planning) is the computational problem of finding a set of valid configurations that move an object from a source to a destination. The term is used in computational geometry, computer animation, robotics, and computer games [5].

A. Path Planning for a Single AUV

Path planning for a single AUV means planning a path for one AUV to take from start to finish using the best path. This paper mainly focuses on offline path-planning algorithms using search-based algorithms. Search-based planning is a motion planning method that uses graph search methods to compute paths or trajectories through a discrete representation of the problem. Many novel algorithms have been developed and tested, and a few of them have proven to yield excellent results.

To select an algorithm for path planning, an evaluation was performed. The evaluation was done by using a project that uses algorithms in the same test environment with the same obstacle setup, resulting in different paths and hence a better understanding of the behavior of algorithms. These tests aimed to find an algorithm that can fulfill the purpose of path planning while considering a cost function. Since more than only one algorithm was able to plan a path at the given environment, it was necessary to perform a more advanced comparison between the different algorithms. To do so the tested algorithms were compared using different performance metrics like described in [6]. The following metrics were used:

1) Computational Time: It represents the time an algorithm takes to compute a path.
2) Success Rate: The probability an algorithm finds a suitable path for 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 total length of the calculated path to reach the destination.
5) Deviation: The difference between the calculated path and the shortest path calculated by any tested algorithm for the same environment.

With these metrics it is possible to deduce that the algorithm of choice is Dstar. Besides other algorithms like 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 has reached the best overall ranking. The algorithms were ranked with a low-point system, where the ranking positions of all metrics were added up and the algorithm with the lowest overall sum is chosen as most suitable.

Dstar stands for "Dynamic Astar." It is similar to the popular path-finding algorithm Astar, which used to find the shortest path between two nodes in a graph [7]. Astar uses a heuristic function to calculate the cost of reaching the destination from a certain node to determine the most suitable successor of the current node. The Dstar algorithm adds the ability to reconfigure parameters. This makes it more viable for environments with little changes, e. g. adding one obstacle to the whole environment. This is useful in case the environment for the path-planning is at a fixed spot, but there is not all information about all obstacles known. Nevertheless, it must be made clear that although the algorithm is called "dynamic", it is an algorithm for offline planning and should not be confused with being a dynamic online planning algorithm. Dstar follows a similar set of data structures as Astar, with an additional list to be maintained [8].

B. Path Planning for Multiple AUVs

Larger missions often involve the use of multiple agents to complete the mission. For instance, to scan a larger area of an ocean, a single AUV going back and forth would often require a lot of time and would not be an ideal solution. Instead, if multiple AUVs can work in synchronization, the task can be finished in less time with better energy savings for an individual AUV. Planning for multiple AUVs is an extended version of path planning. The goal is similar as single AUV, i.e., to travel from start to goal point. Unfortunately, it is not possible to use the Dstar algorithm for planning the path of multiple AUVs, because it would only consider the starting position of the other AUVs as obstacle, which might lead to collisions. The Astar instead reconsiders the new positions of the AUVs during the path planning to avoid collisions. The Multi Agent Path Finding (MAPF) problem is frequently treated as a hybrid of single agent path finding. This often involves traversing the same map through different starting nodes. To take it a step further, the coordination of agents is needed. To incorporate this coordination, a regression-based method is used to check for all possible paths that would result in no collision or conflict. The multiple AUV planning was done using a conflict-based path finding method as explained in the following [9]. Conflict-based search (CBS) is a new optimal path-finding algorithm for solving MAPF problems. CBS is a two-level algorithm. This two-level procedure eliminates the necessity of using the Astar algorithm for planning the paths of multiple AUVs because the paths can again be planned for every AUV by its own using Dstar, since the collisions are handled explicitly at the highest level. At the highest level, a conflict search is conducted by means of a Constraint Tree (CT) which is a tree based on individual agent conflicts. A constraint therefore describes a specific time when an agent is not allowed to be at a certain place. Each node in the CT represents a set of restrictions on the agents' motion. To satisfy the constraints imposed by the high-level CT node, quick single agent searches are conducted at the low level, for which we use the Dstar algorithm. CBS's approach is to create a set of constraints and then find paths that are consistent with those constraints. If these paths intersect, they are considered invalid and would necessitate the addition of new constraints.

This multi-agent CBS starts with detecting possible conflicts between agents using a high-level search algorithm. It is done by analyzing the start and destination positions of each agent and determining the points in time when more than one agent is at the same place. This state would lead to a collision and therefore it leads to a conflict. Each of these conflicts are combined into a conflict set for the low-level algorithm as an input. Since the low-level path-planning algorithms know which places to avoid at which time, a new path can be planned for every agent until all constrained satisfied.

III. Implementation and Evaluation

This section first introduces our architecture implemented in such a way as to compute feasible paths for the AUVs and simulate them. Subsequently, we evaluate our solution by means of case studies.

A. Implemented Architecture

The overall goal of the algorithm discussed so far is to devise an executable path adhering to all mission requirements. In order to convert the input parameters map and target data to the final path, an architecture shown in Fig. 1 has been implemented that is divided into 5 stages to be executed in

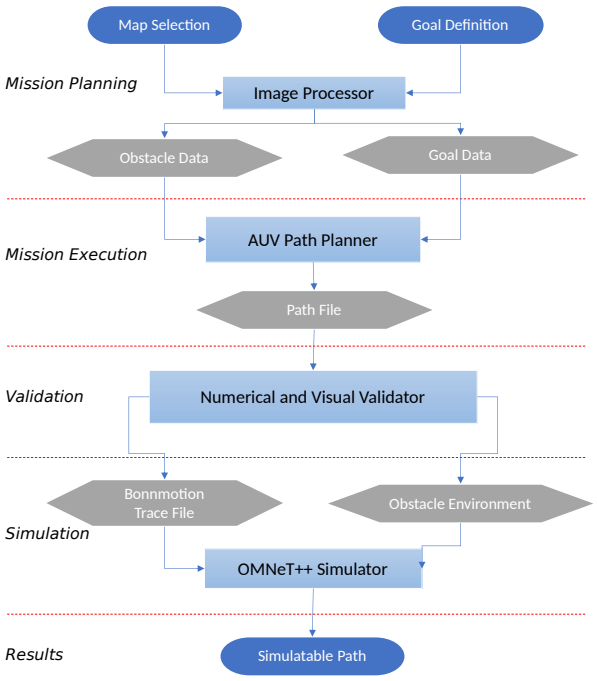

Figure 1. Representation of the architecture.

sequence and also to be repeated when the desired result is not yet achieved. The five stages of the architecture comprise mission planning, mission execution, validation, simulation, and results, which will be detailed in the final paper.

1) Mission Planning: This stage focuses on defining the main motive of the mission. The location of the mission and the exact goals are decided at this stage. The nature of the mission is also of prime importance, i.e., single AUV or multiple AUV. In this step an image processor converts the selected map as well as the goals into obstacle data and goal data which can be used as input for the path planner. A screenshot from Google Maps showing Warnemünde like depicted in Fig. 2 will be converted to a picture like shown in Fig. 3. The converted map consists of only black and white. The white area represents water and the black area represents the land and therefore a big obstacle. The image processing has to go through different steps like converting to 2-color image, removing water areas which are not irrelevant like the little streams in the south-west of Fig. 2. All image processing is implemented using Python and the map and goals are input as jpeg image files. As an output the image processor creates two txt-files for obstacle and goal data.

2) Mission Execution: This is the main part of the architecture as it calculates the path for the next steps. The AUV planner comprises the algorithm based on the obstacle and goal data from mission planning and outputs a basic path for further steps. To calculate the path, the described procedure from Sec. II is used to find the most suitable plan. This stage of our architecture generates only one txt-file, which contains the planned path. Fig. 4 shows the image of Warnemünde with the blue starting-point and the green finishing-point. The blue path that connects these two points was planned by the mission execution stage and shows an optimal path between these points that was calculated by the Dstar algorithm.

3) Validation: The validation step determines whether a solution for the path has been found and the accuracy of the solution. When goals for an optimal path are not met, the output path needs to be checked for accuracy or even recalculated. The validation is split into numerical and visual validation.

The numerical validation checks key data like if the starting point of the planned path is the same as the starting point of the goal data. The same is done for the finishing point. On the other hand, it is checked if the length deviation from the optimal path is within a certain maximum or whether the previous stage must be repeated.

The visual validation checks if there are any points of the path that collide with an obstacle or with other AUVs. If the validation is successful, a file which describes the path as Bonnmotion trace as well as an obstacle environment are generated to simulate the generated path in OMNeT++.

4) Simulation: The simulation stage is intended to simulate the calculated path with the BonnMotionMobility model while also utilizing the OMNeT++ simulator's additional functionalities. The OMNeT++ simulator can calculate external parameters such as congestions for traffic simulations or conflicts during multi-agent simulations.

The simulation is done in OMNeT++ because it enables the lather addition of communication between multiple AUVs to make them cooperate with each other. Since communication is essential for sophisticated cooperation between more than one AUV, it is reasonable to use OMNeT++ to simulate the movement of those, because OMNeT++ is an discrete event simulator which is primarily meant for building network simulators [10].

5) Results: If all the above-mentioned steps show a satisfactory result, a path is ready for test of a real AUV.

B. Case Studies

Following there will be one case study with a single and one with multiple AUVs. It highly depends on the mission and itself if it is worth to use more than one AUV. A mission with a job like moving from a starting to a finishing point with investigating the seafloor might be a job for only one AUV. A possible mission that could use multiple AUVs comprises underwater exploration or mapping. The AUVs could be deployed in a coordinated manner to cover a large area of the ocean floor, gathering data on the seafloor topography, geology, and marine biodiversity.

1) Single AUV: The first case study aimed to test the most basic target of an AUV, i.e., to travel from one point to another. The AUV is instructed to travel from a start point to an exact location while trying to avoid obstacles encountered during the journey. It is an offline planning process with information on all the obstacles available during planning. The plan only considers static obstacles and does not consider dynamic obstacles. For this mission, a test location is selected at Warnemünde. The location is the meeting point of the Warnow River and the Baltic Sea. This location can be found at the coordinates 54.18348, 12.09044.

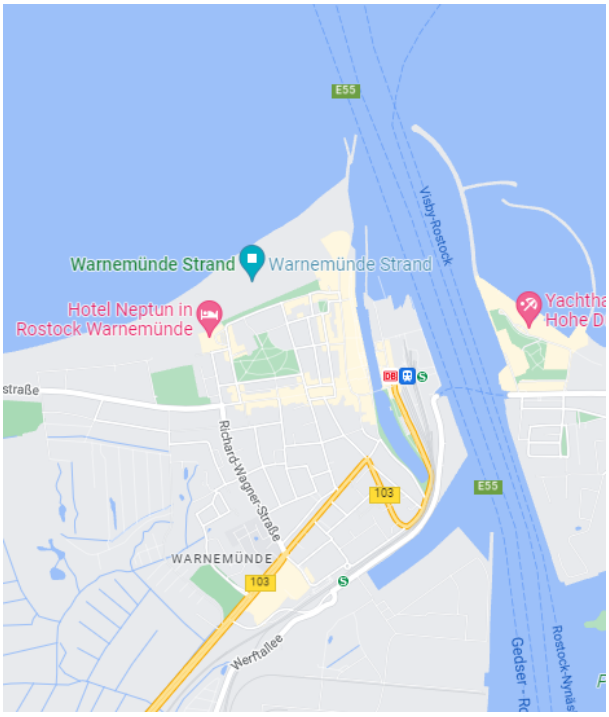

Figure 2. Screenshot of Warnemünde from Google Maps.

Figure 3. Screenshot of Warnemünde with land in black and water in white.

The mission attempts to navigate the AUV from the Warnow River channel's interior (coordinates 54.1742, 12.09651) to a point in the open sea (coordinates 54.18322, 12.07709).This would imitate the situation of a mission that required the AUV to navigate through tight corners as well as irregular boundaries.

The physical environment for this mission has features like a port and a river channel, as well as a wide open ocean. Fig. 2 shows the map view of the mission location. The start position is at 54.1742, 12.09651, and the target location is at co-ordinates 54.18322, 12.07709, which corresponds to X and Y locations of 200, 175 and 50,107, respectively.

The image processor converts Fig. 2 into Fig. 3 with removing canals, guidelines and streets into a two-tone black-and-white image. This image-information is needed by the path planner to set every black pixel as an obstacle.

Since the mission only requires a single AUV the Dstar algorithm from Sec. II can be used to calculate the shortest possible path from the starting-point to the destination without colliding with any obstacle. Fig. 4 shows the generated path directly inside the black-and-white obstacle image. It gets clear, that the Dstar algorithm has found the only way of leaving the Warnow to reach the finishing point. Also it is the shortest path possible.

The Validation of the planned path is also done and it is considered as numerically correct, because the first element of the path consists of the same coordinates as the starting-point, as well as the last coordinate of the path is the same as the finishing-point. This shows that the AUV starts at the correct position and finishes at the required target location. The visual validation was also successful, since there were no

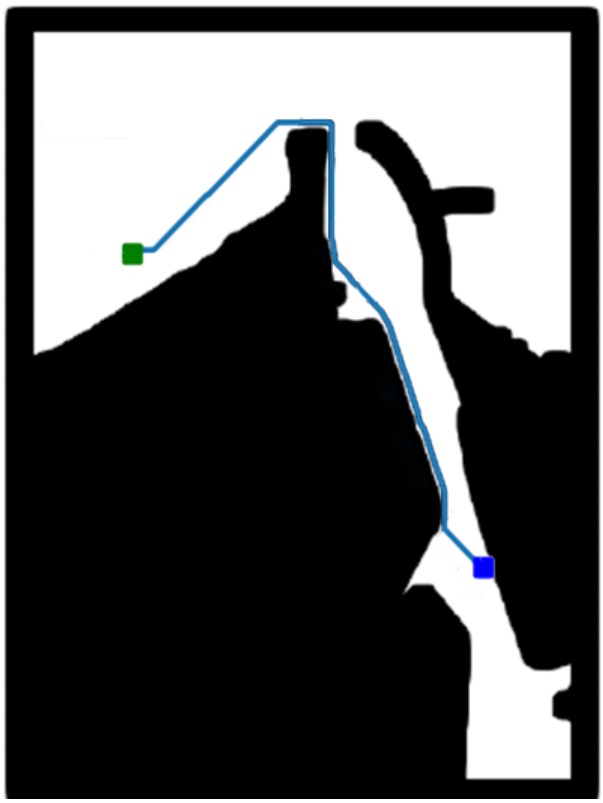

Figure 4. Path at Warnemünde planned by Dstar from blue starting-point to green finishing-point.

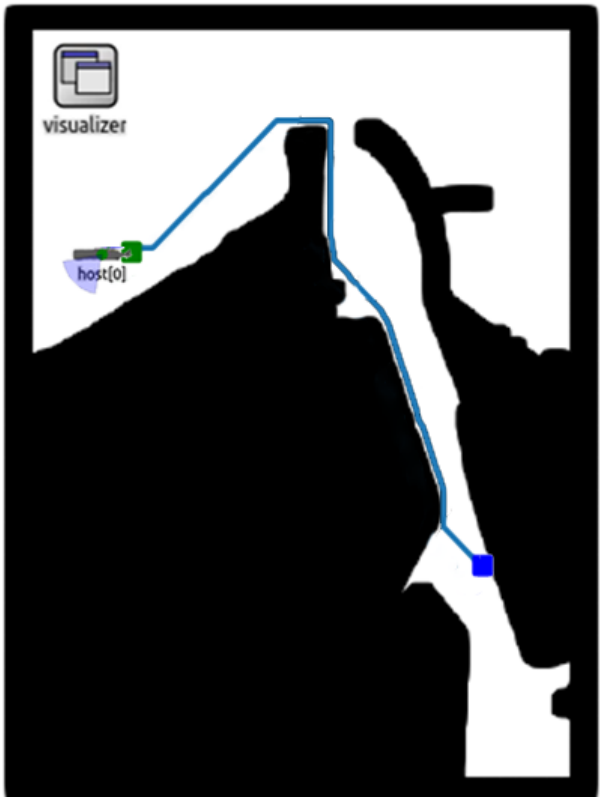

Figure 5. Planned path simulated in OMNeT++.

coordinates from the path of the AUV which are on land and would lead to a collision.

After successful completion of all the above steps, the mission is ready to be simulated in OMNeT++. The simulation involves the usage of OMNeT++ as the simulation framework, the Bonnmotion Mobility Model for the movement, and Qtenv for the visualization of the simulation. The output path from the validation is already in the right format for OMNeT++ to serve as a Bonnmotion trace file. On the other hand the environment has to be used by OMNeT++, since, without it, it would not be possible to determine if the simulation with OMNeT++ generated the desired output.

Fig. 5 shows the 2D-visualization of OMNeT++ with the given path and environment. It shows the AUV right after reaching the destination at the Baltic Sea. This case study makes it apparent that the developed architecture works for single AUV missions.

2) Multiple AUVs: The first mission involves deploying multiple AUVs in a simulated scenario to test collision avoidance algorithms. The AUVs are programmed to travel to different targets that have been set up in such a way as to cause them to cross each other's paths, creating situations where the collision avoidance algorithms can be evaluated. This kind of mission is good for testing and improving the performance of AUVs when they have to work close to other vehicles.

The second mission involves using four AUVs to conduct a site survey of individual wind turbines at the Barrow Wind Farm located in the East Irish Sea. The purpose of this mission is to showcase the potential of AUVs in analyzing the structure

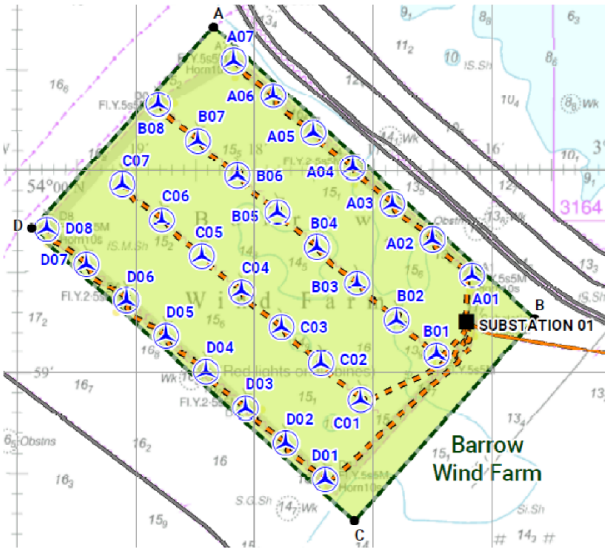

Figure 6. The Scottish Barrow Wind Farm with positions of turbines and cabling.

of wind turbines for tasks such as maintenance and damage inspection, which are traditionally carried out manually using diving equipment. By using AUVs, the mission aims to automate the task of surveying wind turbines, which would reduce effort and increase efficiency. This mission demonstrates the ability to plan and execute a complex operation involving multiple AUVs.

Fig. 6 shows an extract from a seamap which includes the position of every single wind turbine. Unfortunately the image processor is not able to generate the black-and-white image exclusively by itself. It is necessary to draw a picture that makes it more clear where wind turbines are and where there is water.

The goal definition consists of a single starting-point, which is located at the point of Fig. 6 that is called substation 01, because this is assumed as a safe spot to bring the AUVs into water. Since the finishing-point for every single AUV is not known before the processing, all 30 wind turbines are defined as potential end-points of the paths.

To execute this mission the CBS method is used as it involves multiple AUVs. Since the AUVs need to cover multiple waypoints the planning for of this mission is fairly complex. To calculate the next step of AUV after surveying a wind turbine needs to be decided by offline planning in order to account for any casualties. The sequence of execution for each AUV calculated by the path planner, came to the clue, that it is the best case if every AUV surveys one line (A to D in Fig. 6)of wind turbines. Also, there are set four waypoints around each wind turbine because of the purpose of surveying these. To check the whole wind turbine it is necessary to inspect it from every side.

The validation of this mission is highly crucial due to the multiple waypoints for each AUV being targeted. The procedure for numerical validation is similar to previous missions but has to be done multiple times. The mission has 30 wind turbines and 4 waypoints around each wind turbine, resulting

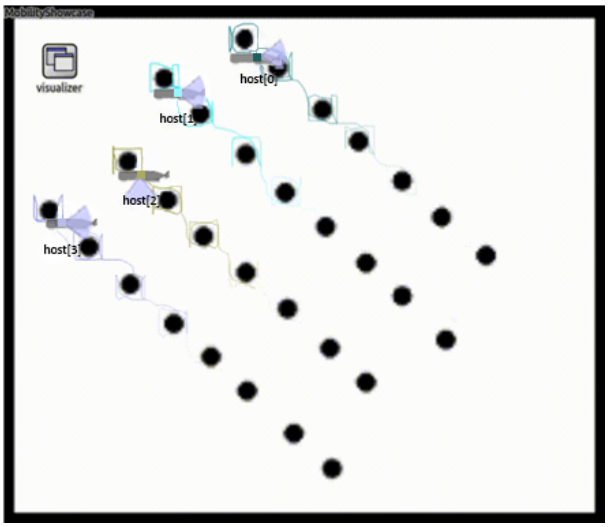

Figure 7. The simulated inspection of the turbines at the wind farm.

in a total of 120 sets of start and goal locations that need to be numerically validated.

Fig. 7 shows the simulated case study. It shows the AUVs right after finishing the inspection of the last wind turbine. From the past course can be seen which sequence every AUV processed to reach its destination point.

IV. Conclusion

The main goal of this paper is to investigate the viability of planning paths for AUVs and testing these in a simulation platform utilizing OMNeT++ motion models.

It has been shown, that path planning for a single AUV and path planning for multiple AUVs are the two scenarios for the route planning that need different approaches. Planning the paths for a single AUV the Dstar search algorithm delivers suitable paths with reasonable time effort. Whereas, planning the paths for simultaneous moving AUVs is more complex. Since, two or more AUVs cannot be in the same place at the same time, it is necessary to use a conflict-based search algorithm. Such an algorithm does also find a usable solution for the shown case studies. It can be used to locate the best course through a complicated environment. The simulation of the planned paths shows the usability of the whole architecture with its different stages.

Future research may examine the use of different path planning algorithms in more complicated situations. And the simulation in OMNeT++ might be used to add a beneficial communication for cooperative missions with multiple AUVs.

References

[1] "Future security: 7th security research conference, future security 2012, bonn, germany, september 4-6, 2012. proceedings." [Online]. Available: http://link.springer.com/10.1007/978-3-642-33161-9

[2] D. Cook, A. Vardy, and R. Lewis, "A survey of AUV and robot simulators for multi-vehicle operations," in *2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*. IEEE, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/document/7054411/

[3] A. Varga, "OMNeT++," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Springer, pp. 35–59. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_3

[4] OMNeT++. BonnMotionMobility. [Online]. Available: https://doc.omnetpp.org/inet/api-current/neddoc/inet.mobility.single.BonnMotionMobility.html

[5] L. Jaulin, "Path planning using intervals and graphs," vol. 7, no. 1, pp. 1–15. [Online]. Available: https://doi.org/10.1023/A:1011400431065

[6] A.-I. Toma, H.-Y. Hsueh, H. A. Jaafar, R. Murai, P. H. J. Kelly, and S. Saeedi, "PathBench: A benchmarking platform for classical and learned path planning algorithms." [Online]. Available: http://arxiv.org/abs/2105.01777

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," vol. 4, no. 2, pp. 100–107, conference Name: IEEE Transactions on Systems Science and Cybernetics.

[8] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 3310–3317 vol.4.

[9] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," vol. 219, pp. 40–66. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370214001386

[10] OMNeT++. OMNeT++ discrete event simulator. [Online]. Available: https://omnetpp.org/