Autonomous Search and Rescue: Path Planning

Todd Cooper[1], Callam Hartley[1], James Hudson[1], Daniel Jacobsen[1]

[1] Griffith University, Southport QLD, Australia

Abstract. The use of unmanned autonomous vehicles for search and rescue operations offers several advantages, including enabling access to hazardous areas without endangering human lives and faster detection using powerful sensors. This paper presents an approach for autonomously navigating a submarine in a simulated disaster scenario to retrieve survivors while avoiding hostile objects. The approach integrates the Robot Operating System (ROS) and Process Analysis Toolkit (PAT) to simulate and control the robot's behaviour within a grid environment. By utilizing PAT's path planning capabilities, optimal paths can be determined which are then interpreted and executed in ROS. The model successfully simulates a 2D environment, allowing the submarine to explore and update its knowledge of its surroundings while dynamically replanning routes when a new behaviour is required. The simplified representation of the environment exposed limitations of the BFS algorithm, even in a quantised world characterised by simplicity. However, it also highlighted the feasibility of the approach in search and rescue tasks with smaller state spaces. Findings highlighted areas of interest which requires improvement, as well as logical areas for extension of the experiment, laying the groundwork for future investigations in the integration of ROS and PAT for robot control in search and rescue tasks.

Keywords: Autonomous-Vehicle, Robot, Submarine, PAT, ROS

1 Introduction

In recent years, autonomous robotic systems have become vital tools for search and rescue operations in challenging environments. The ability to deploy robots capable of efficiently navigating through hazardous areas, such as the middle of the ocean or disaster-stricken regions, has emerged as a critical area of research. Robots with these capabilities play a pivotal role in locating and rescuing survivors in situations where human intervention may be limited or unsafe.

One commonly employed model is the use of unmanned aerial vehicles (UAVs) and autonomous underwater vehicles (AUVs). These drones offer the advantage of aerial or underwater mobility, allowing them to cover large areas quickly and survey environments from a safe distance. Equipped with sensors, such as cameras and thermal imaging devices, AUVs and UAVs have been utilised in various search and rescue scenarios, including the search for the debris of Malaysia Airlines flight MH370 [1].

Ground-based robots also play a crucial role in search and rescue operations. These robots are designed to traverse difficult terrain, such as rubble, debris, or flooded areas, where human access may be restricted or unsafe. Equipped with sensors, like LIDAR and cameras, along with specialised tools for physical interaction with the environment, these robots enhance navigation and survivor detection capabilities.

This paper focuses on the use of robotics in performing search and rescue within aquatic environments. The proposed approach involves the conjunction of ROS and PAT to simulate a robot capable of navigating and searching a grid area for survivors. By leveraging the capabilities of ROS, the robot's actions are simulated within a 2D environment, where ROS reads the commands generated by PAT and executes them accordingly. The primary objective of the proposed approach is to enable the robot to search the grid area comprehensively until all survivors are found and return them home. PAT plays a central role in the proposed model by performing the path-planning operations for the robot. It employs the depth-first and breadth-first search algorithms to determine the optimal path for the robot to follow in its search for survivors. In some cases, the optimal path may be the shortest route. In other circumstances, the shortest route may be too computationally expensive to calculate, so the easiest route to calculate becomes optimal.

This report presents a detailed exploration of the proposed approach, including the integration of ROS and PAT, the algorithms used for obstacle detection and avoidance, and the path planning strategies employed. An in-depth look at the experimental design and its results is provided. An evaluation of the merits and potential drawbacks of the created system for simulating aquatic search and rescue operations concludes the paper.

2 Related Work

Unmanned robotic search and rescue (SAR) systems offer advantages such as agility, mobility, endurance, adaptability, and scalability over traditional human-operated methods. However, existing literature highlights various challenges and open problems in designing and deploying such systems. Communication, coordination, navigation, perception, decision making, and human-robot interaction must all be considered. Academic papers discuss these issues and propose potential solutions. We will discuss three recent papers that present novel solutions for different aspects of unmanned robotic SAR missions. The first paper presents a deep reinforcement learning robot for exploration in unknown cluttered environments [2]. The second introduces an innovative approach for using autonomous underwater vehicles for SAR at sea [3]. The third proposes a model checking based planning and goal reasoning framework for autonomous systems [4]. This report discusses the main contributions, methods, results, and implications of each paper for the field of unmanned robotic SAR.

2.1 Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments

The paper "Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments" by Farzad Niroui et al. [2] proposes a new approach to robot exploration in unknown cluttered environments. The approach, known as Frontier-based Deep Reinforcement Learning (FDR), combines the traditional frontier-based exploration approach with deep reinforcement learning (RL).

Frontier-based exploration is a simple but effective approach for exploring unknown environments. It involves identifying unexplored regions in the environment, referred to as frontiers, and guiding the robot towards these areas. This iterative process is repeated until the entire environment has been explored. Alternatively, deep RL is a powerful machine learning technique used to train agents to perform complex tasks. In the context of search and rescue, RL can enable robots to learn how to navigate through cluttered environments in order to find survivors.

The FDR approach combines the strengths of frontier-based exploration and deep RL. Frontier-based exploration provides a simple and efficient way to identify unexplored regions in the environment. Deep RL allows the robot to learn how to navigate through these regions safely and efficiently.

The proposed approach was evaluated in a simulated environment and showed a greater ability to explore more of the environment and find more victims than traditional approaches. This approach is a promising new method for robot exploration in unknown cluttered environments. Its potential is recognised to be used in a variety of search and rescue applications, such as exploring collapsed buildings or finding survivors in natural disasters.

2.2 AUV for Search & Rescue at Sea – An Innovative Approach

Sathyaram Venkatesan's paper "AUV for Search & Rescue at Sea – An Innovative Approach" explores the utilisation of AUVs for search and rescue missions at sea [3]. AUVs are unmanned vehicles that can operate autonomously for extended durations, making them ideal for SAR missions that can be hazardous or time-consuming for human operators.

The paper focuses on the design and development of an AUV that is specifically designed for SAR missions. The AUV is equipped with a variety of sensors, including sonar, cameras, and a magnetometer, which allow it to detect and identify objects underwater. The AUV is also equipped with a communication system that allows it to transmit data back to a shore station.

To assess the AUV's performance in SAR missions, a series of experiments were conducted. The results demonstrated the AUV's successful detection and identification of underwater objects, as well as its capability to transmit data back to the shore station in real-time.

The paper concludes by highlighting the potential AUVs to revolutionize SAR missions at sea. Their ability operate autonomously for extended periods makes them ideal for SAR missions that can be perilous or time-consuming for human operators.

2.3 GRAVITAS: A Model Checking Based Planning and Goal Reasoning Framework for Autonomous Systems

The paper "GRAVITAS: A Model Checking Based Planning and Goal Reasoning Framework for Autonomous Systems" by Bride et al. proposes a framework for autonomous systems that utilise model checking to reason about goals and plans [4].

The paper presents the "Goal Reasoning And Verification for Independent Trusted Autonomous Systems" (GRAVITAS) framework, which uses model checking to verify the correctness of plans and goals, and generate new plans when necessary. The paper also presents a case study of the GRAVITAS framework applied to a robotic system, demonstrating its effectiveness in handling complex scenarios.

The main contribution of the GRAVITAS framework is its ability to provide planning and goal reasoning capabilities through model checking, ensuring plan and goal accuracy. The framework is also able to handle complex scenarios, as demonstrated in a case study of a robotic system. The paper addresses the challenges of autonomous systems, recognising the need for autonomous decision-making capabilities while ensuring safety, reliability, and adherence to predefined goals and constraints. It also acknowledges the scalability challenge in planning and goal reasoning for complex autonomous systems and discusses techniques to address and judge the feasibility and scalability of the model checking approach.

The paper concludes by highlighting the importance of future work to improve the level of trustworthiness of GRAVITAS by extending the verification from high-level plans to low-level plans and conducting realistic simulations and real-world demonstrations.

3 Proposed Approach

The proposed approach aims to achieve the objective of autonomously navigating a submarine through a simulated disaster scenario, retrieving all survivors while avoiding potential hostile objects. This approach integrates ROS and PAT to simulate a robot within a grid environment. By utilizing PAT's path planning capabilities, the robot can determine optimal paths, which can be interpreted and executed in ROS. The robot's actions are simulated in a 2D environment, allowing it to explore and update its knowledge of the grid. Whenever obstacles are encountered, the robot dynamically replans its route to ensure continued progress towards the mission's goals.

3.1 Environment Modelling and Sensors

In the model a grid world is generated representing a simulated disaster scenario with a number of survivors and hostiles placed randomly. The submarine is initially placed at the top left corner of the grid at the ‘home’ position (0,0). The submarine is equipped with two sensors, a survivor sensor (e.g. infrared), and a hostile sensor (e.g. sonar) which can detect their respective objects in north, south, east, and west directions. These sensors were designed to have variable ranges and can be set within the program. To build an accurate understanding of its surroundings, the submarine maintains an internal representation of the map. This representation will be continually updated as the submarine's sensors gather information, allowing it to accrue knowledge about the environment. An example of the internal representation of the submarine's world can be seen in Figure 1, marked as "known world".

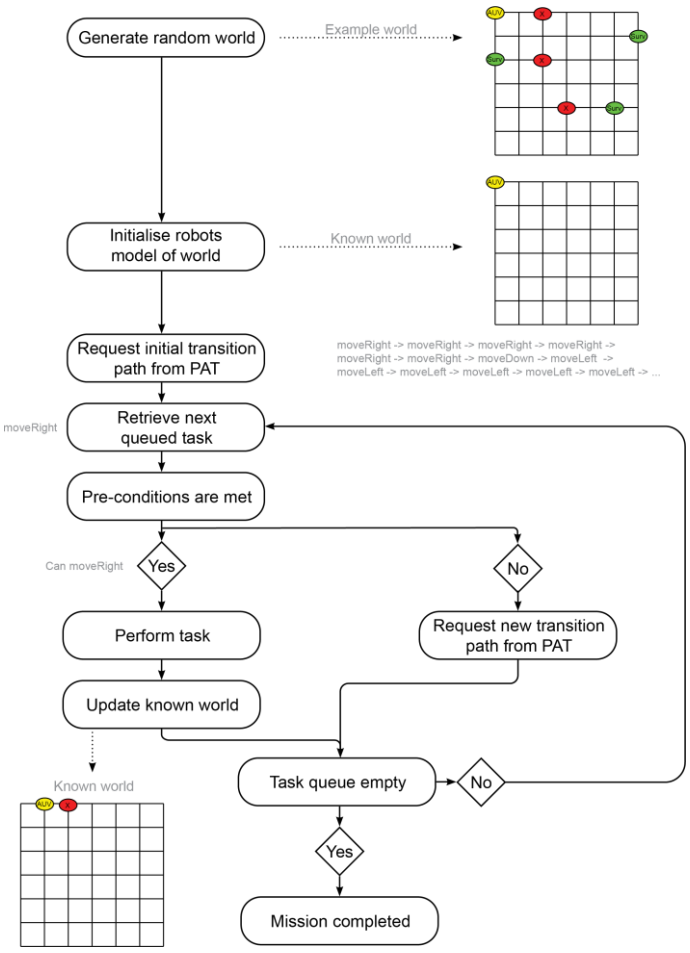

Fig. 1. Overall workflow of the program. A sample world has been generated and the known world updated after the first moveRight has been performed.

When the submarine reaches a position where a survivor is present it will pick them up (the default capacity of the submarine is two, however, this can be modified). The goal of the model is to safely return all survivors to the home position, while avoiding collisions with hostiles. Figure 2 shows an example of the submarine in a simulated environment, and an example of a hostile triggering a sensor.

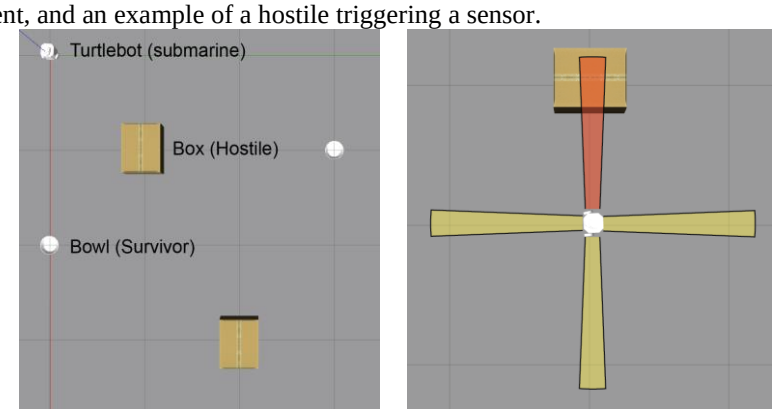

Fig. 2. Example of a simulated environment and sensor range of the submarine. For compatibility reasons, the turtlebot represents the submarine/AUV, the box a hostile entity, and the bowl a survivor to be rescued.

3.2 Control System Architecture

The submarine control system logic is structured hierarchically, with three high-level behaviours: surveying the environment, retrieving a detected survivor, and returning home. The action sets required to execute these behaviours are generated using PAT when requested by the ROS control system logic. Surveying the environment serves as the foundational behaviour, allowing the robot to gather information about the grid. Once survivors are detected, the robot behaviour extends to the higher-level behaviours of retrieving them, and safely returning home. This hierarchical approach enables the robot to perform more complex tasks as needed, while ensuring a default behaviour of continuous environmental surveying when no specific tasks are being executed. Figure 3 shows this process in further detail.

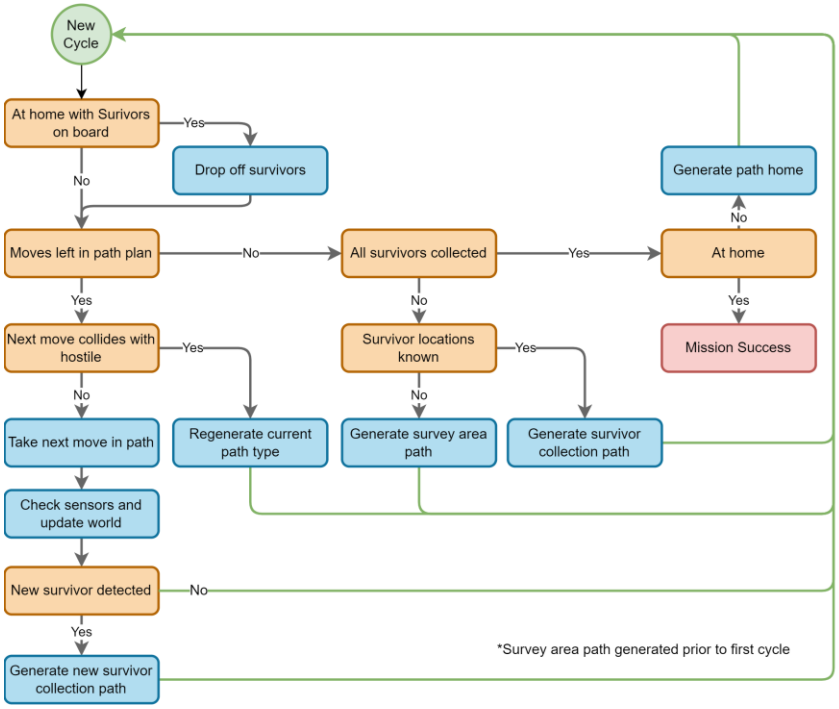

Fig. 3. Submarine control system logic as modelled in ROS

3.3 Integration of ROS and PAT

The proposed approach utilises PAT to generate action plans by providing it with the submarine's internal representation of the map and the desired behaviour. This will allow for a separation of tasks in which ROS is responsible for executing the submarines control logic, monitoring environment variables and state, while delegating path planning to PAT in situations which require behaviour alterations. This integration ensures a more efficient and flexible control system for the autonomous submarine.

By integrating ROS and PAT, this approach aims to achieve the objectives of autonomous navigation, survivor retrieval, and obstacle avoidance in a simulated disaster scenario. The following sections will provide further details on the implementation and evaluation of this proposed approach.

4 Implementation and Experiments

The overall system utilises three tools to implement the project: ROS, PAT, and Gazebo (see Figure 4). ROS provides the main framework for the system functions and code, PAT handles the path planning, and Gazebo simulates the environment. These three tools complement each other to facilitate the effective execution and evaluation of the project.

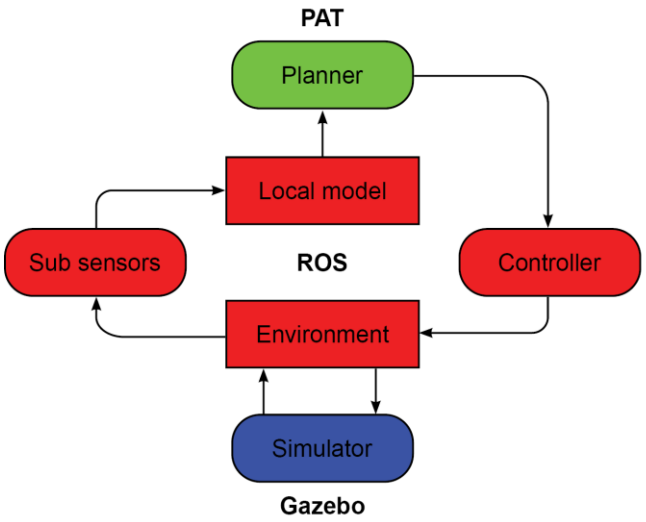

Fig. 4. Overall system flow, inspired by Bride et al.'s work [4].

4.1 Robot Operating System

ROS is programmed using C++, with three files containing the important code:

- search_rescue.cpp
- update_grid.cpp
- communal_defines.cpp

ROS generates a random world on each execution of the program and places a specified number of survivors (represented by bowls) and hostiles (represented by boxes) into a 2D grid array. The submarine (represented by the turtlebot) is spawned and ROS initialises its model of the world. The sensors in ROS simulate a real sonar or infrared sensor and retrieve the data from Gazebo based on the submarine's location.

4.2 Process Analysis Toolkit

PAT reads .csp files to model the world and determine the next path to take. ROS sends PAT the data via internal service calls specifying the type of search and supplying 'world.csp' which contains the submarines internal representation of the world. One of three options of navigation are selected based on the submarines goal: survey the environment, collect survivors, or return home. Survey the environment occurs when the submarine has no known locations of survivors and is not currently at carrying capacity. Collect survivors happens when the submarine knows the location of a survivor. Return home occurs when the submarine is at capacity, or all survivors have been collected.

Goal Specifications. The three goals defined to discover action sets which achieve the three behaviours are:

1. Goal: goalAreaChecked
   - Assertion: All positions in the grid are visited or hostile and the submarine has returned home.
   - Definition: goalAreaChecked is defined as (&& i:{0..Rows-1}@(&& j:{0..Cols-1}@(world[i][j] == Visited || world[i][j] == Bomb)) && home)
   - The behaviour UnvisitedMove() which prioritizes moves to unvisited locations is used.
2. Goal: noSurvivors
   - Assertion: Known survivors are cleared from the map if possible under the constraint of the maximum carrying capacity. In all cases, once the carrying capacity is reached, the submarine returns home.
   - Definition: noSurvivors is defined as ((&& i:{0..Rows-1}@(&& j:{0..Cols-1}@(world[i][j] != Survivor)) && (onBoard != maxCapacity || home)) || (onBoard == maxCapacity && home))
   - The behaviour used is UnvisitedMove().
3. Goal: home
   - Assertion: The submarine is in the home position at (0,0).
   - Definition: home is defined as (xpos == SUB_HOME_X && ypos == SUB_HOME_Y).
   - The behaviour VisitedMove() which prioritizes moves to visited locations is used.

4.3 Gazebo

ROS utilises an updateGrid service that interacts with Gazebo to display the locations of the submarine, hostiles, and survivors as represented in the initial map. As the submarine moves throughout the world, Gazebo continues to display the changes in real time by calling this service with current environment states.

4.4 Execution

To run the simulation from terminal, first the command `catkin_make` is needed to compile the ROS packages. In a terminal within the catkin_ws directory call `roslaunch assignment_3 launch_world.launch` to open the Gazebo simulated environment. In a new terminal window call `rosrun assignment_3 update_grid` to start the updateGrid service. Finally, in a third terminal window call `rosrun assignment_3 search_rescue` to begin the simulation. Figure 5 shows an example of a simulation. Detailed instructions can be found in the repository readme file in section 9.

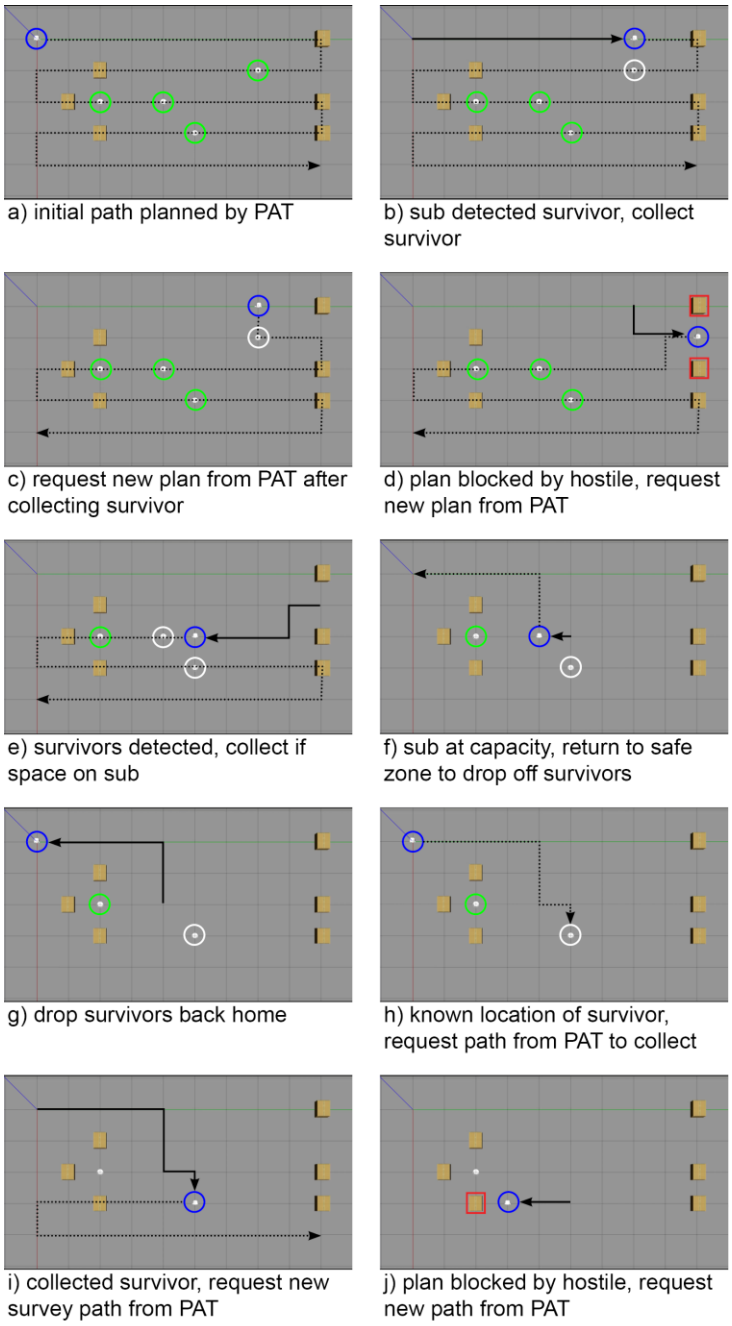

Fig. 5. The beginning of a sample run showing the combinations of the three path options provided by PAT to complete the survey.

5 Experiments and Testing

Testing was carried out on a Windows 10 Machine running Oracle VM VirtualBox. The Windows machine configuration can be seen in Table 1.

Table 1. Computer Hardware and software

OS version	Windows 10 Pro
CPU	AMD Ryzen 5 3600
GPU	GeForce GTX 1660 Super
RAM	32GB

5.1 Test case 1

(capacity= 2, survivor range = 1)

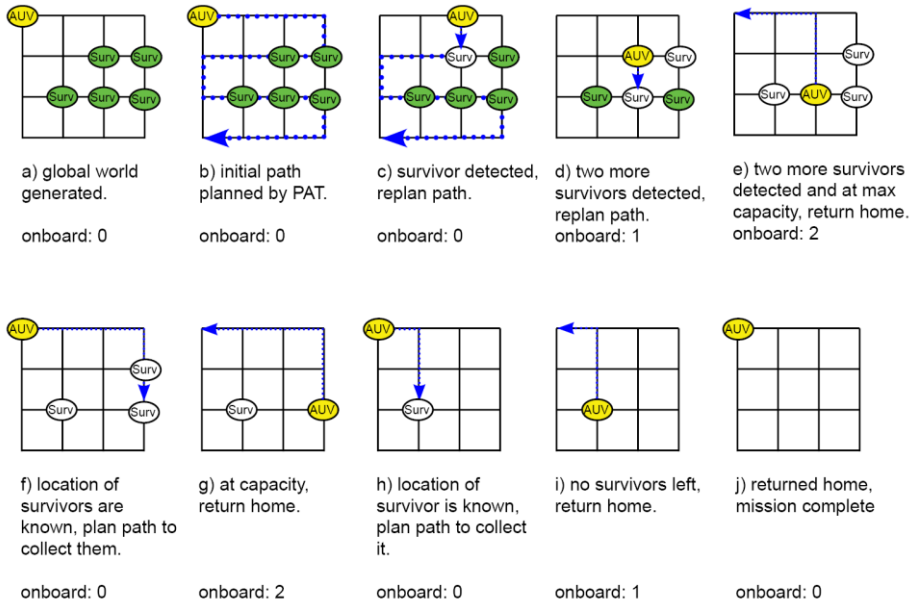

Fig. 6. Test case 1.

Results This test ensures known survivors are prioritised and collected as soon as possible. Once the submarine has dropped the onboard survivors back home, the path back to know survivors is via BFS.

5.2 Test case 2

(capacity= 2, hostile range = 2)

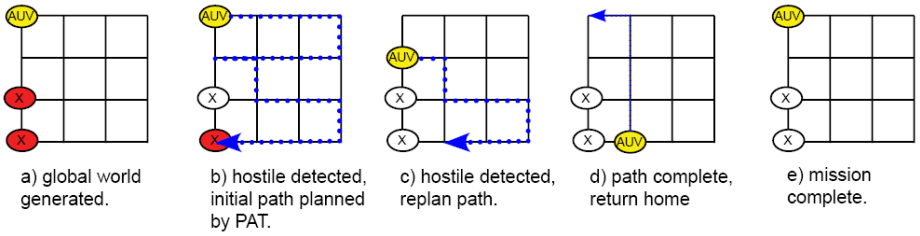

Fig. 7. Test case 2.

Results This test ensures the submarine returns home after the entire search space has been covered. As the number of survivors in the simulation is specified before program compilation, this test case is somewhat redundant. However, it is important to note that this situation may arise in a real-life situation, therefore is important to ensure the submarine can finish the survey even if no survivors are found.

5.3 Test case 3

(capacity= 2, hostile range = 2, survivor range = 1)

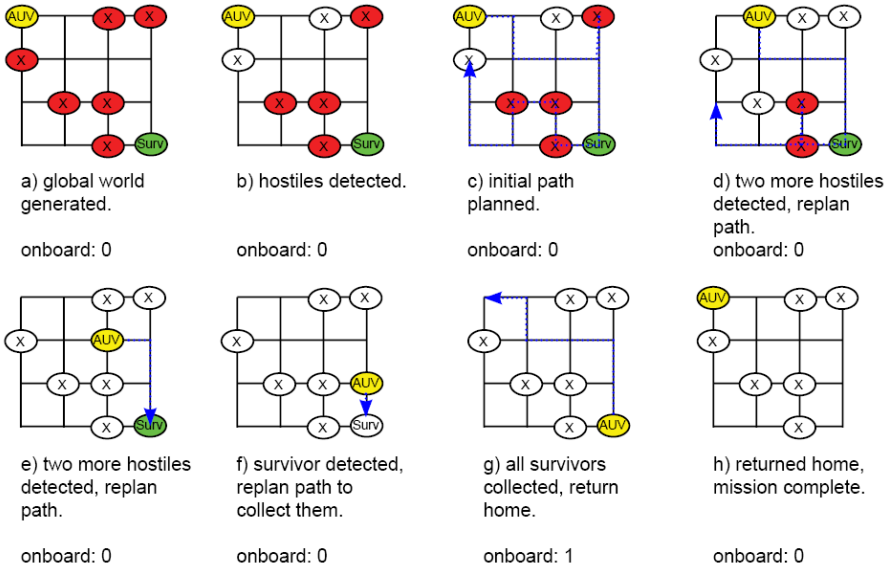

Fig. 8. Test case 3.

Results This test ensures the submarine successfully navigates a congested environment without collisions. There is only one path to the survivor and back home, which the submarine successfully follows.

5.4 Test case 4

(capacity= 2, hostile range = 2, survivor range = 1)

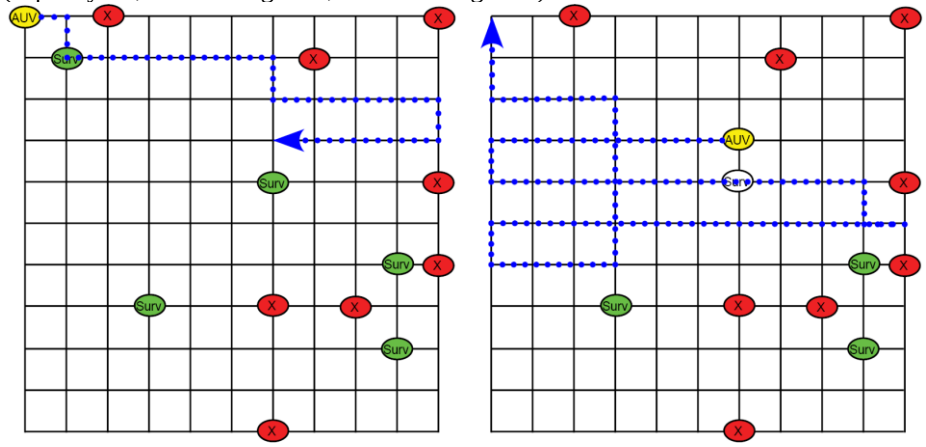

Fig. 9. Test case 4.

Results. The objective of this test case is to validate the functionality of the timeout mechanism when a BFS search takes too long to determine the optimal path. As the state space expands, the computational time required to find a BFS path increases significantly. To address this, a time limit of 10 seconds is set for the BFS search to find a path. If the search exceeds this time limit without finding a path, a timeout function is triggered, and the search algorithm switches to Depth-First Search (DFS). This can be observed in figure 9b, where the submarine has detected a second survivor, but the time taken to plan a path home exceeds 10 seconds. A DFS path is then provided as the solution. It is important to note that the timeout mechanism and the subsequent switch to DFS in this simulation are not suitable for real-life scenarios. DFS does not guarantee optimality and can result in meandering paths to the goal state. Therefore, the observed trade-off between path generation time and efficiency is not realistic in situations where finding an optimal path is necessary. In the simulation, the generated world should ideally be limited to a 10 x 10 grid to avoid BFS timeouts. However, even with the timeout feature, DFS becomes increasingly inefficient as the world size grows, emphasizing its limited applicability in complex real-world scenarios.

6 Discussion

While the project aims to create a robot capable of exploring an area and rescuing survivors, there are limitations which must be acknowledged. These limitations impact both the computational aspects of planning the robot's route and its feasibility for real-world applications.

6.1 Computationally Expensive Search Algorithms

One significant limitation is the computational complexity associated with the search algorithms used for path planning in PAT. PAT utilises BFS to determine the path for the submarine to return home. However, when the state space becomes excessively large, the BFS algorithm expands exponentially. This can result in two major problems: exceeding memory capacity or prolonged search times. Consequently, the efficiency of the exploration process is compromised. For state spaces that exceed the bounds set by the search algorithm and the existing hardware, a new approach would be needed.

To overcome the computational limitations associated with large state spaces, alternative search algorithms can be explored. For example, instead of using BFS, more efficient algorithms such as A* or Dijkstra's algorithm can be considered. These algorithms use heuristics or weighted edges to guide the search and reduce the computational complexity. In this case, a heuristic that prioritises heading towards the home coordinates could drastically reduce computation times compared to the BFS's brute force approach. However, due to the constraints of using PAT for path planning and the limited knowledge and resources available for its implementation, incorporating a heuristic algorithm was not feasible in this study.

6.2 Control System for Realistic Movement

In the current version of the model, the submarine navigates the environment in a grid-like fashion, effectively teleporting a predetermined distance in the desired direction. For the model to be successfully applied in real-world scenarios, a more sophisticated movement control system is necessary. This control system should regulate the submarine's movement and speed within the environment, considering factors such as distance from obstacles (survivors, hostiles, etc.) and terrain conditions, where it can manoeuvre accordingly. Additionally, the incorporation of sensors that operate in the simulated environment is necessary to emulate practical applications more accurately. Although the model is sufficient for the simulated environment, it is important to recognise these limitations for achieving a model which is deployable for real-world applications.

To address this limitation, it is necessary to introduce a control system to regulate the movement of the submarine. Implementing a Proportional-Integral-Derivative (PID) controller to adjust the submarine's velocity based on environmental factors would be an initial step towards resolving the issue. Although the possibility of incorporating such a control system was considered, it was deemed beyond the scope of the current project.

6.3 Handling Energy Consumption

Robots exploring large areas or disaster-stricken zones often operate with limited energy resources. Consideration of fuel constraints is crucial to optimise the robot's exploration strategy and ensure that it can effectively complete its mission within the available energy budget. Neglecting this aspect would undermine the realism and practicality of the project.

By incorporating energy constraints into the robot's exploration strategy, its operations can be optimised to make efficient use of available energy resources. This can involve tracking fuel level as the robot explores the environment, then returning home when either there is not enough fuel to complete the calculated path, or only enough to complete the backtrack home. By integrating energy constraints, the robot must make informed decisions regarding exploration paths and behaviours, balancing different needs to achieve successful missions.

6.4 Isolated survivors

In the current version of the model, there is a limitation where survivors may be inaccessible due to hostiles bocking all paths to them. This issue can lead to program crashes as the model does not currently handle such situations. ROS expects all survivors to be accessible by at least one path.

To address this limitation, the model can be enhanced to incorporate mechanisms for adaptive decision-making. When survivors are found to be inaccessible, the system could adapt its behaviour to simply flag the position or take some other action. Although this limitation was discovered during the testing phase, it was simply noted as an area for improvement in future works due to time constraints.

6.5 Evaluation

These limitations highlight the need for further development and research in several key areas. Addressing these challenges will enhance the effectiveness and viability of the submarine for successful exploration and SAR missions. Future work should focus on improving the computational efficiency of search algorithms, designing an advanced control system for realistic movement, and integrating real-world constraints into the planning process. By addressing these limitations, we can strive to create a more robust and efficient submarine capable of effectively handling the challenges encountered in real-world scenarios.

7 Conclusion

In conclusion, this paper presents a simple approach to modeling a search and rescue task using an autonomous robotic vehicle in a 2D grid environment with ROS and PAT. The integration of ROS and PAT successfully generated and modeled grid environments in Gazebo, allowing the robot to perform its mission of collecting survivors and returning them to the home location. Limitations were encountered, particularly when the environmental complexity increased, or survivors were located far away from desired goal locations. The optimal path calculation time using BFS became prohibitively long, highlighting the need for alternative search methods such as A*.

Further improvements are necessary to better approximate real-world scenarios and handle complex tasks. These include porting the environment into Gazebo, implementing sensors to detect objects, and enabling the robot's movement in the simulated world.

Additional extensions, such as considering fuel constraints, periodic survivor movements, and timed survivor expiry, would enhance the model's realism.

Despite its simplicity and abstractions, this implementation demonstrates the potential of integrating ROS and PAT for controlling autonomous vehicles in underwater search and rescue tasks. By refining the search algorithms and addressing the suggested future work, a more robust system can be achieved, offering promising results in real-world environments. This research provides a preliminary exploration of the integration of ROS and PAT for controlling autonomous vehicles in underwater search and rescue tasks, offering potential directions for future work in the field.

8 References

1. News.com.au: https://www.news.com.au/travel/travel-updates/incidents/new-ocean-infinity-search-for-mh370-encountering-big-problems/news-story/89efc71d393585bb3efc1fcd10b765aa (2018).
2. Niroui, F., Zhang, K., Kashino, Z., Nejat, G.: Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments. IEEE Robotics and Automation Letters 4(2), 610-617 (2019).
3. Venkatesan, S.: AUV for Search & Rescue at sea - an innovative approach. 2016 IEEE/OES Autonomous Underwater Vehicles (AUV), 1-9, (2016).
4. Bride, H., Dong, J. S., Green, R., Hou, Z., Mahony, B., Oxenham, M.: GRAVITAS: A Model Checking Based Planning and Goal Reasoning Framework for Autonomous Systems. Engineering Applications of Artificial Intelligence 97 (2019).

9 Execution Details

The following link provides a GitHub repository[1] where all the necessary files for running the simulations covered in this document can be found. The repository contains a readme.md which gives an in-depth explanation for executing the files, along with any necessary prerequisites. Additionally, a brief summary of the programs overall objective is attached.

1. https://github.com/sal-the-dev/3806ict_assignment_3/#readme