Mission Assurance for Autonomous Undersea Vehicles

Karl Siil
JHU Applied Physics Laboratory
Laurel, Maryland
Karl.Siil@jhuapl.edu

Aviel Rubin
Johns Hopkins University
Baltimore, Maryland
rubin@jhu.edu

Matthew Elder
JHU Applied Physics Laboratory
Laurel, Maryland
Matthew.Elder@jhuapl.edu

Anton Dahbura
Johns Hopkins University
Baltimore, Maryland
antondahbura@jhu.edu

Matthew Green
Johns Hopkins University
Baltimore, Maryland
mgreen47@jhu.edu

Lanier Watkins
Johns Hopkins University
Baltimore, Maryland
Lanier.Watkins@jhuapl.edu

***Abstract*—Autonomous vehicles are all but inevitable, and assurance that they will behave safely with respect to passengers, as well as bystanders incidentally exposed to them, is moving forward, albeit slowly. The state of the art often involves stopping the vehicle, perhaps after diverting it to a nearby safe place. While this is good news, it does not fully realize the benefits of autonomy. Autonomous vehicles are built for a purpose; call it a mission. Being able to perform the mission, or part of it, while experiencing faults (or cyber-attack) should be a factor in determining the vehicle's suitability for the mission. This paper explores the state of the art in achieving autonomous mission assurance in the context of autonomous undersea vehicles (AUVs). It identifies gaps in the literature and proposes a novel plan to address certain gaps.**

Keywords—assurance, autonomy, mission, resilience, safety, vehicle

I. Introduction

Autonomous vehicles are all but inevitable, and assurance that they will behave safely with respect to passengers, as well as bystanders incidentally exposed to them, is moving forward, albeit slowly. For example, safety systems are being architected into unmanned aerial vehicles (UAVs) such that before the vehicle can endanger people or property, the safety system will engage and disable the UAV [3], eliminating such risks.

While this is good news, it does not fully realize the benefits of autonomy. Autonomous vehicles are built for a purpose; call it a *mission*. Being able to perform the mission, or part of it, while experiencing faults (or cyber-attack) should be a factor in determining the vehicle's suitability for the mission.

This paper explores the state of the art in achieving autonomous mission assurance in the context of autonomous undersea vehicles (AUVs). It identifies gaps in the literature and proposes a novel plan to address certain gaps, i.e.,

- Protect passengers onboard an AUV.
- Protect vessels in the AUV's area of operation, along with the fixed obstacles prior works address.
- Complete the mission, or part of it.

Section II provides an autonomous vehicle overview. Section III defines the mission assurance problem this paper addresses. Section IV describes the threat model. Section V presents related works in improving autonomous vehicle safety. Section VI proposes to improve mission assurance by adding mission-essential functionality in a separate (simpler) layer that lends itself more readily to being formally verified and therefore more trusted. Section VII identifies known limitations and suggests future research to address those limitations.

II. Autonomous Vehicle Overview

In layman's terms, an autonomous vehicle can be viewed as a set of computers connected to sensors and actuators (motors, servos, etc.) with which to interact with the physical world. An autonomous vehicle learns and adapts to dynamic environments and evolves as the environment around it changes. Compare this to vehicle automation that typically runs within a well-defined set of parameters and is very restricted in what tasks it can perform [6]. A car's (non-adaptive) cruise control and basic collision-avoidance system are examples of automation. A self-driving car, on the other hand, is autonomous.

Consider an abstract autonomous vehicle that consists of an autonomy engine (AE), which is some combination of processors, software and possibly external communications mechanisms, e.g., to a cloud-based compute capability. The AE is connected to a variety of sensors and actuators, which inform it about its environment and allow it to take actions in that environment, respectively. This paper focuses on a fictitious tourist AUV similar to the crewed T-SUB by Silvercrest Submarines [8].

An AUV is built with a mission in mind. The mission might be surveying undersea cables or pipelines, or it may be something for the military. The mission of this paper's AUV is tourism, i.e., taking tourists to see sights like shipwrecks and geological formations. Each sight is called a point of interest (POI) and Section III describes the tourism scenario in detail.

While the AUV is performing its mission, software flaws in the AE or cyber-attacks attempting to disrupt the mission may

DOI 10.1109/SPW50608.2020.00056

take place. Manufacturers of autonomous vehicles that transport passengers must do more to convince users and the public at large that their products, which are heavily reliant on complex autonomy software, are safe in these conditions.

III. Problem Statement

Consider a company that operates tours of underwater POIs (see Fig. 1 for a notional example). The company would like to replace its crewed submersibles with AUVs. An operational scenario, including constraints and risks, is described below.

Submersibles are dispatched to visit as many POIs as possible within a given time period, henceforth called maximum and minimum tour-time limits. The vessels operate near their design limits against local currents, and do not venture into deep water due to the risk of becoming incapacitated and sinking below their maximum operational depth. The submersibles must also avoid each other and surface traffic while giving the tours. All tourist submersibles partner in an acoustic range finding service that uses trusted fixed stations and provides accurate absolute and relative positions. Other, non-partner, submersibles and surface vessels can only be detected by sonar. Surface vessels must monitor marine radio for broadcasts to clear a given area to allow a submersible to surface.

The AUV being considered is similar in capacity, endurance, and performance to crewed submersibles currently in use [8]. Therefore, each AUV would be a one-for-one replacement for a crewed submersible until, if the plan is successful, all the crewed vessels are retired. Moreover, because of the similarities between the crewed and autonomous vessels, the AUV is subject to the same limitations, e.g., no operations in deep water.

Risks for both crewed vessels and AUVs are running aground, straying into deep water, or coming too close to (or colliding with) a POI, pier or other vessel. Replacing the crewed submersibles with AUVs creates additional risks that would not exist with, or that would be handled by, a human operator. These additional risks stem from software failure of or cyber-attack against the AE, and include not visiting some or all of the POIs, staying out too long, or returning too quickly. Some of these add danger, e.g., the AUV could navigate away from the pier and hover submerged indefinitely. Others are just bad for business.

Given the risks, the tour company would only consider using AUVs if it can be assured that a level of safety and customer satisfaction can be achieved.

IV. Threat Model

Assume an adversary has unfettered access to modify the AE in any way they desire. This could be done via insiders, supply chain attacks, or exploitation of vulnerabilities in deployed systems. Whatever the case, this paper is not about defeating cyber-attack, but how to handle one that has already occurred. Worst case assumptions are made for all potential actions of a compromised AE.

In contrast, assume adversaries cannot affect development facilities, supply chains, etc., used to manufacture the safety systems proposed below. Furthermore, fielded safety systems and the vehicles they are on are protected sufficiently that an adversary cannot get physical access without being detected.

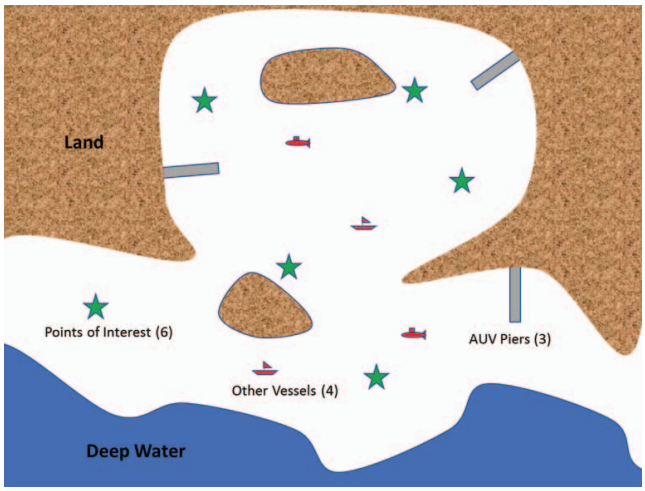

Fig. 1. Notional AUV tourism scenario.

An adversary cannot affect their operation, but sensors, motors, and other components are subject to wear and tear, accidental breakages, and other natural phenomena that could cause them to fail. Such failure cases are out of scope in the initial research, and will be addressed when the technology being developed is more mature (see Section VII).

V. Related Works

Developing high-quality complex software continues to be a significant challenge. To address the problem differently, the concept of incorporating simpler more reliable safety systems has arisen over the years and shows promise. However, as discussed below, current safety systems are limited. Some are implemented with limited assurance, particularly against malicious actors. Very few protect the mission that the vehicle needs to perform.

Xiao, Li, and Zhang [10] developed a rule-based safety kernel for unmanned systems. However, the safety kernel is a user-level process implemented on a main control processor that contains all the other application software and connects via Wi-Fi to an external PC. Malicious software introduced into this processor presumably could circumvent the safety functions.

Safeguard [3] is a totally independent onboard UAV geofencing system (including independent sensors) with two discrete outputs – a warning that a geofence violation is imminent, and a kill signal if the warning does not result in corrective action. Safeguard only protects against damage by the vehicle, not damage to the vehicle, which is understandable because Safeguard was built under the assumption that hull loss is acceptable for small inexpensive UAVs.

ICAROUS [2] augments Safeguard's geofencing by adding detect and avoid capabilities against fixed obstacles and other vehicles, as well as the ability to compute a conflict-free "return to mission" path. ICAROUS is being integrated with Safe2Ditch [5], a computer vision-based landing site selection system, which should reduce risk to the UAV. However, in the event of a geofence violation, Safeguard still takes drastic action.

VI. Proposal

Mission assurance for autonomous undersea vehicles (MA-AUV) proposes to extend what the above related works have done in the following ways:

- Protect passengers onboard an AUV.
- Protect vessels in the AUV's area of operation, along with the fixed obstacles prior works address.
- Complete the mission, or part of it.

MA-AUV focuses on replicating as much mission essential functionality as possible in less-complicated safety systems that are realizable with high-quality high-assurance software. MA-AUV trades reduced autonomous functionality for predictable responses to on-mission events that lead to deviations, and completion of the mission, or part of it, despite these deviations.

Some related works [10] address assurance of system functionality minimally or not at all. Others provide higher assurance of the system, but limited focus on assured mission success. Safeguard [3] in its current form kills the UAV motors during a safety violation, ending the mission, and possibly damaging the vehicle. The modified Safeguard-like system proposed as a starting point below could safely stop and surface an AUV, which minimizes the risks of harm to passengers and vehicle damage, but still ends the mission outright and probably requires passenger rescue by some other vessel.

ICAROUS [2] improves mission assurance somewhat, but its ultimate recourse at present is still Safeguard. When Safe2Ditch [5] is incorporated the vehicle will be safer, but ICAROUS cannot take over the mission if other onboard processors become erratic or unresponsive. The iterative set of solutions below strive to do that.

A. Resilience Layer

MA-AUV proposes to place a resilience layer (RL) between a potentially faulty or compromised (collectively termed, "malfunctioning") AE and the sensors/actuators (see Fig. 2). The RL seeks to limit the effects of any potentially dangerous actions commanded by the AE via the simplest implementation and highest software assurance possible. In addition, as the research progresses, the RL's goal is to maximize protection of the mission (or a subset of it), as well as the vehicle, its occupants and bystanders.

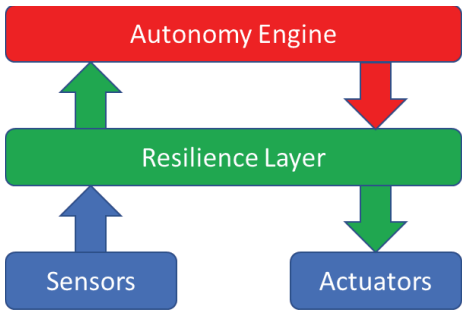

Fig. 2. Resilience layer between autonomy engine and physical components.

The proposed implementation has the RL fed by the same AUV sensors that feed the AE. If the need arises, however, adding independent sensors would be straightforward.

The RL can generate commands to the AUV's motors and other actuators. Under normal conditions, the RL allows the AE's commands to pass to the actuators. If the AE is exhibiting erratic or potentially dangerous behavior, however, the RL can disable the AE interfaces and send its own commands. These commands could be modifications of what the AE sent, or completely different ones, e.g., if a sharp turn in the opposite direction is required to avoid an obstacle.

The RL functionality described in Section F is based on the five functions of the NIST Cybersecurity Framework: Identify, Protect, Detect, Respond, and Recover (IPDRR) [7]. Only the Recover functions are described in detail in this paper.

B. Research Process

MA-AUV is developing a simulated testbed and AUVs complete with RLs. RL development is iterative, with each iteration more capable than the prior one in improving mission assurance. Testing and scoring of each RL iteration is conducted in a variety of manually configured and randomly generated test environments (See Section G).

Our RL development provides assurance beyond just testing via static source code analysis (SSCA) and formal verification (FV). SSCA reduces errors and vulnerabilities in code, but does not guarantee that the code implements the intended functionality correctly. This is the purpose of FV. None of the RL software is expected to be too complex for SSCA, but later iterations may be for FV. In that case, FV constraints could be relaxed regarding residual errors or unproven functions left in the code. Alternatively, simplifications could be made to advanced-iteration RL software to make successful FV possible, while still retaining functionality that is beyond simpler iterations. These two approaches will yield either fully functional but less assured RL software, or high-assurance software with limited functionality. Either way, they will inform the state of the art for high assurance in production software development.

C. Testbed Functionality

MA-AUV implements a simulated AUV testbed where test environments can be configured and test events executed against an AUV test article. The simulated testbed is implemented with the Gazebo robot simulation environment [9] and its Unmanned Underwater Vehicle Simulator plugins [4]. Each test environment has seafloor bathymetry, currents, POIs, AUV piers, surface vessels and other submersibles. See Section E for how each element is represented in a test environment.

D. Autonomous Undersea Vehicle Mission Functionality

The following tour (i.e., mission) functions are implemented in the AUV AE (and eventually RL), and tested in the simulated testbed. Section F describes how the RL monitors the AE and performs Recover functions when safety is at risk.

A tour starts with the AUV generating a *departure tour plan*, which describes the *tour legs* to be traversed from the pier to the first POI, between the POIs, and returning to the pier. Estimated times of arrival (ETAs) at the POIs and the return to the pier are

also in the tour plan. Once activated, the departure tour plan becomes the *active tour plan* and the AUV executes the tour.

Throughout the tour, which is conducted submerged, the AUV uses acoustic range finding and sonar to maintain separation from fixed obstacles and other vessels. If a deviation is required to maintain separation, the AUV determines whether the active tour plan is still achievable, i.e., the next POI can be reached by the ETA. If the active tour plan is no longer achievable, the AUV generates an *enroute tour plan* based on its current position and the remaining unvisited POIs.

The enroute tour plan can re-order or eliminate POI visits. If no modified tour can be completed within the maximum tour-time limit, even after all remaining POI visits are eliminated, the only valid enroute tour plan is to return the AUV directly to a pier. No further POI visits are permitted.

As with the departure tour plan, once activated the enroute tour plan becomes the active tour plan and the AUV begins executing it. This cycle of deviating, determining achievability and (if necessary) generating/activating additional enroute tour plans continues until all POIs, if any, on the latest active tour plan are reached and the AUV has returned to the pier. At that point, the tour is ended and scored (see Section G). If there is a safety violation that results in the invocation of Recover functions, the tour may end before a pier is reached. Scoring still occurs in that case.

E. Separation

The RL bases its safety-related decisions on maintaining separation between the AUV, fixed obstacles, and other vessels. The use of separation in MA-AUV is very similar to its use in the air-traffic control system [1], where it is intended to keep aircraft far enough apart such that unexpected maneuvers or loss of situational awareness by one aircraft doesn't immediately endanger other aircraft and gives everyone time to react.

The testbed implements separation by giving each obstacle a well-defined topology. Land masses, deep water, and the sea floor look the same as their real-world counterparts. Fixed obstacles and vessels have simple topologies to reduce testbed complexity. Tri-axial ellipsoids represent POIs and vessels. Rectangles represent piers.

Everything in the testbed has a set of three or more nested boundaries. The boundaries are like those used in Safeguard [3], and delimit where the state of the AUV's relationship with a given object changes. Crossing a boundary causes state changes in the AUV, which in turn may precipitate actions by the AUV or other vessels or affect scoring. Only the AUV is affected by or can react to these boundaries (see Section VII).

The various boundaries and the effects of crossing them are defined as follows and depicted by the examples in Fig. 3:

- **Physical Boundary**: This represents the physical obstacle. Crossing this boundary is considered a collision, though it's a bit of a misnomer for deep water.
- **Separation Boundary**: This represents the minimum separation (distance) that must be maintained from the obstacle within the boundary. Crossing this boundary enters an obstacle's reduced-separation zone (RSZ).

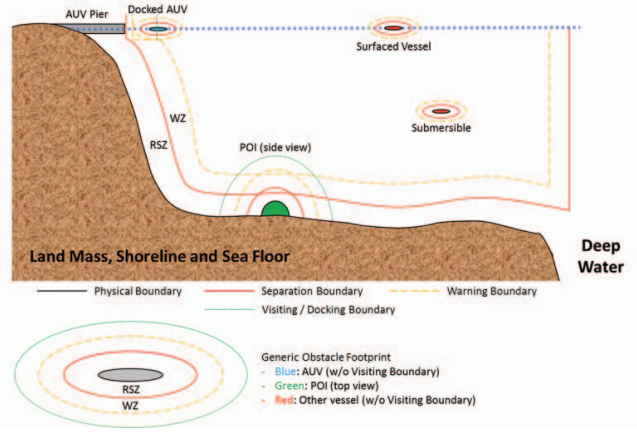

Fig. 3. Example obstacle and vessel separation boundaries (not to scale).

- **Warning Boundary**: This represents the range from an obstacle at which the RL attempts to prevent a safety issue. Crossing this boundary enters an obstacle's warning zone (WZ).
- **Visiting Boundary**: This only exists for POIs and represents the maximum distance that is considered visiting a given POI. Crossing this boundary causes the POI to be considered visited.
- **Docking Boundary**: This only exists for AUV piers and represents the maximum distance that is considered docking at a pier. Crossing this boundary and reducing speed over ground to zero causes the AUV to become docked and ends the test event.

F. Resilience Layer Functionality

The RL Protect functions must grant permission to activate a tour plan before the AUV can start or modify a tour. Without an active tour plan, the RL prevents AE-issued commands from reaching the AUV's motors or other actuators.

During a tour, the RL Detect functions monitor the AUV's position, depth, heading, and speed, as well as the distances to fixed obstacles and other vessels. These data are used to compare the AUV's current situation with what is expected from the active tour plan. The Detect functions also determine whether the AUV is overdue at a POI or pier.

Respond functions are invoked when the AUV is in danger of losing separation. If the AUV enters a WZ, the RL alerts the AE that it must take corrective action to exit the zone, but the RL takes no further action of its own. If the AUV enters a RSZ or collides with an obstacle, Recover functions are invoked.

Recover functions are invoked when the AUV enters a state where the RL takes permanent control from the AE for the remainder of the tour. These are mostly cases where the AE has lost separation. The other case where Recover functions are invoked is if a new enroute tour plan is required and the AE has not taken the steps to get one activated. This is not necessarily a safety risk, but is at least undesirable for the passengers.

The ideal Recover goal is to complete the tour in the AE's place with no impact on tour safety or quality. MA-AUV

attempts to come as close to this goal as possible by iterating on the RL implementation. The first Recover iterations implement Safeguard-like [3] functionality. The RL shuts off propulsion, drops anchor and surfaces the AUV. This cuts the tour short (i.e., fails the mission), but achieves the safety goals.

Successive Recover iterations add the ability for the RL to return to a pier, activate enroute tours from a pre-generated tour-plan library, and ultimately incorporate the AE tour-planning functions. This final iteration replicates a great deal of the AE autonomy into the RL, and may eliminate the need for an AE, at least for tour-plan generation and execution. Replicating this functionality, or a limited version of it, is probably possible. The challenge is to provide the required assurance.

The pre-generated tour-plan library mentioned above is the first step in significantly improving mission assurance. The library is created after the test environment is configured, before the first test event. Creating the pre-generated tour-plan library is a trusted function. The tour plans are assumed the best possible for the algorithms used, unaltered by malicious entities, and delivered in a trusted manner to the AUV.

The tour-plan library contains a small number of departure tour plans, and as few as one. Multiple departure tour plans add variety for the tourists. In terms of MA-AUV, multiple departure tour plans are generated to measure the effects of the differences on scoring (see Section G). In addition to departure tour plans, a set of enroute tour plans is generated for the library. Each pre-generated enroute tour plan addresses a specific set of initial conditions and provides a path for completing the given tour. The initial conditions for each enroute tour plan are the set of remaining POIs to visit and the current position of the AUV.

The more potential initial conditions covered by pre-generated enroute tour plans, the higher the expected score. However, the tour-plan count explodes as the number of POIs increases. While the numbers become daunting quickly, for a given test environment many tour plans may be similar and combinable. Also, in some (perhaps many) cases the visit order may not matter.

The tour-plan library must factor in the possible current position of the AUV, which is anywhere, but many positions are probably alike. For example, all positions within some radius around a given point might have the same optimal tour plan to join. Adding *feeder legs* from such points to join existing tour legs gives the AUV more options to safely navigate from a post-deviation position to a valid tour plan path.

Tour-plan library size is dependent on test environment geography, POI and pier placement. And, there may be commonalities that can be applied generally, e.g., an algorithm for reducing tour permutations, or the number of feeder legs and how to place them, which can lead to manageably sized libraries. MA-AUV will explore such possibilities. At worst, the approach shows promise for small numbers of POIs.

G. Testing and Scoring

MA-AUV's success will be measured by running test events in simulated test environments representing the scenario described in Section III. Multiple test environments will be created, both manually to test the RL's ability to assure the mission in specific situations, and randomly to test the RL across a broad range of situations and uncover any deficiencies caused by those not foreseen in the manually configured environments.

After each test event, a *tour score* based on the departure tour plan requirements defined in Section D is computed to represent how well the AUV did on the tour. The tour score represents the tourists' satisfaction and has the following properties:

- Visiting all the POIs within the tour-time limits with no loss of separation (LOS) yields a perfect score of 100.
- A collision yields a score of 0, as does passengers requiring rescue or the AUV being towed back to a pier.
- Given no LOS and the tour runs within the time limits, the score is based on how many POIs were visited.

While tour satisfaction is an important metric, MA-AUV's focus is on improving mission assurance. The question, therefore, is not just how good the tour was, but how well the RL did in limiting any negative impacts to the tour of a malfunctioning AE. To determine this answer, the tour score for each test event is compared to the *best tour score* possible for the given environment, assuming a properly functioning AE, and factoring in runtime variations in currents and other vessels encountered. This reduces the effects on the tour score of variables the RL cannot control. For example, if on a given tour the AUV encounters so many other vessels that very few POIs are visited, the RL should not be penalized.

To compute the best tour score, the same departure tour plan used in the test event is executed, minus any variations in currents or encounters with other vessels. This establishes an initial best tour time, T_B, along with a POI visit coverage list that includes all the POIs.

For each deviation in the original tour, a mini-simulation is executed with the same initial conditions as when the deviation started. A fully functional AE performs the deviation until the obstacle has been avoided. The time to perform the deviation is added to T_B and any loss of separation is captured. From the new AUV position, achievability is computed and an enroute tour plan generated, if necessary, using only the eligible unvisited POIs. Any POIs not visited on this enroute tour plan are removed from the visit coverage list and are ineligible for inclusion in enroute tour plans required in later deviations.

The formula used to compute the actual tour score is also used to compute the best one, substituting the best tour time and LOS values. The actual tour score is divided by the best one and the *RL score* is the quotient, ranging from 0 to 100 (after multiplication by 100 and application of a floor function).

The best tour score has limited realism, because it implicitly assumes the way a test event played out and the best way it could have played out encountered the same deviations. It does not consider fully, for example, that how the AUV reacts to a deviation depends on how it reacted to previous ones. In fact, if the AUV had reacted differently to a previous deviation, a subsequent deviation may not even have occurred. While limited, the best tour score provides a means to normalize RL performance against specific test event challenges. The need to improve the best tour score algorithm is noted in Section VII.

H. Anticipated Results

Testbed and AUV development have just started, limiting any concrete results. It is expected, however, that early RL iterations will do little for mission success but will greatly improve passenger safety. The simplest Recover function requires rescuing passengers with another vessel, or towing the AUV to a pier. Still, these are improvements over, e.g., the AE piloting the AUV into an obstacle or deep water.

The Recover functions that return the AUV to a pier will make partially successful tours possible, as long as one or more POIs are visited before the Recover function is invoked. Pre-generated tour-plan libraries will enable the RL to execute partial and possibly complete tours, further improving mission assurance. Finally, full tour-planning functionality would make the RL capable enough that the AE may not be required for the planning, execution, and deviation handing portions of a tour, leading to maximal mission assurance.

The best tour-plan generation algorithms, i.e., the ones that lead to the best tour scores, will be ones that leave as much spare time as possible to allow for potential deviations. This will require optimally balancing the use of slower speeds and loitering at POIs with staying as close to the minimum tour-time limit as possible, which are conflicting goals. Also, the RL iterations that implement tour-plan libraries are expected to require quite a bit of experimentation to determine what enroute tour plans (and associated feeder legs) produce the best scores.

VII. Known Limitations

If the tour-plan library functionality is impractical, more work will be required to determine how the RL can force the AE to generate better enroute tour plans. Without at least this functionality, a malfunctioning AE can submit extremely sub-optimal, but valid, enroute tour plans. The plans could have the AUV meander all over the test environment and return to the pier without visiting any more POIs. For now, the only comfort is that sub-optimal planning is reflected in the scoring.

The best tour score algorithm described in Section VI.G is limited, because it does not fully account for the actual places and times of deviations encountered in a test event. As the research progresses and experimental results are generated, improvements to this algorithm will be investigated.

The testbed does not capture the consequences of the AUV forcing other vessels to collide with fixed obstacles or each other. For example, if separation between the AUV and another vessel is at risk, both vessels react to prevent LOS. The AUV is scored on its ability to respond, while the other vessel may silently lose separation or even collide with one or more other obstacles. The AUV causing another vessel to suffer LOS or collision might be considered a safety concern, though it could be argued that the other vessel shares the blame in getting into this situation. Still, future versions of the testbed should capture such indirect safety risks.

The failure of sensors, motors and other components is not considered, nor are cases of an adversary attempting to deny or spoof sensors. As the base technology of the RL matures, additional failure and attack scenarios will be considered.

This paper ignores cost. A real-world engineering solution would factor in cost along with functionality and assurance. An organization's financial resources are limited, and high assurance may not be economical.

VIII. Conclusion

This paper explored the state of the art in autonomous mission assurance. It identified gaps in the literature and proposed a novel plan to address certain gaps. It also presented a plan to implement a resilience layer (RL) to address mission assurance for autonomous undersea vehicles (AUVs), a.k.a., MA-AUV. An iterative development approach was described to allow functionality to be developed, verified, tested, and to have its value assessed by objective quantitative scoring. By developing RL functionality iteratively, MA-AUV provides "off ramps" such that, if a given level of functionality with the required amount of assurance is unachievable, useful open-source artifacts will still have been built during earlier iterations and can be provided to the autonomous vehicle community.

Other proposed work includes improved scoring algorithms for better measurement of RL performance, capturing the second-order effects of a malfunctioning AUV on other vessels, and factoring in the cost of developing a high-assurance RL.

Acknowledgment

The author is thankful for the contributions of Pete Dinsmore, Dave Sames, Meghan Warner and Cindy Widick in making this research possible.

References

[1] "Aeronautical Information Manual - AIM - ATC clearances and aircraft separation," Federal Aviation Administration. [Online]. Available: https://www.faa.gov/air_traffic/publications/atpubs/aim_html/chap4_section_4.html.

[2] M. Consiglio, C. Munoz, G. Hagen, A. Narkawicz, and S. Balachandran, "ICAROUS: Integrated configurable algorithms for reliable operations of unmanned systems," 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), 2016.

[3] E. T. Dill, S. D. Young, and K. J. Hayhurst, "SAFEGUARD: An assured safety net technology for UAS," 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC), 2016.

[4] "Home," Unmanned Underwater Vehicle Simulator Documentation. [Online]. Available: https://uuvsimulator.github.io/.

[5] P. C. Lusk, P. C. Glaab, L. J. Glaab, and R. W. Beard, "Safe2Ditch: Emergency landing for small unmanned aircraft systems," Journal of Aerospace Information Systems, vol. 16, no. 8, pp. 327–339, 2019.

[6] S. Matteson, "Autonomous versus automated: What each means and why it matters," TechRepublic, 07-Jun-2019. [Online]. Available: https://www.techrepublic.com/article/autonomous-versus-automated-what-each-means-and-why-it-matters/.

[7] Nicole.keller@nist.gov, "The five functions," NIST, 10-Aug-2018. [Online]. Available: https://www.nist.gov/cyberframework/online-learning/five-functions. [Accessed: 12-Jan-2020].

[8] Submarines and ROVs for sale and hire by Silvercrest Submarines. [Online]. Available: https://www.silvercrestsubmarines.co.uk/mediumtouristsubinfo.html.

[9] "Why Gazebo?," gazebo. [Online]. Available: http://gazebosim.org/.

[10] Y. Xiao, G. Li, and Y. Zhang, "A rule-based safety kernel for unmanned system," INFONA. [Online]. Available: http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.ieee-000006321097. [Accessed: 12-Jan-2020].