CHALLENGES AND PROGRESS IN BEHAVIOR-BASED ADAPTIVE AUTONOMY

Arjuna Balasuriya,[*] Tyler Mayer,[†] and Gerald Fry[‡]

Today's uncrewed platforms are typically operated by humans using remote control to guide every detailed aspect of a mission. However, as missions become more complex, there are many scenarios (particularly in the marine, ground, and space domains) in which operators are unable to communicate with these uncrewed platforms in real time (due to adverse environmental conditions, regulatory restrictions on communications in ecologically sensitive areas, active interference by adversaries, or the desire to remain covert), making it challenging to recalibrate and update mission and control parameters on the fly. Fortunately, significant technical advances in onboard computing power and enhanced sensors offer a pathway to a level of autonomy that can overcome such communications limitations. Behavior-based autonomy architectures enable onboard mission autonomy software to select optimal behaviors for the mission at hand, assess the evolving state of the mission, and adapt to changing environmental conditions. Developers and operators of autonomy systems for challenging environments will gain insights into the importance of onboard behavior and parameter adaptation to the success of long-duration uncrewed platform missions in unpredictable situations and unknown environments. Designers will benefit from a discussion of how multi-objective optimization and reinforcement learning techniques are currently being used to enable assured autonomy onboard these uncrewed platforms.

INTRODUCTION

Uncrewed platforms have become increasingly popular in recent years and have been employed for various commercial (e.g. household appliances, self-driving cars, search and rescue operations, manufacturing, agriculture, and uncrewed space travel) and defense applications (e.g., uncrewed platoons, intelligence, surveillance and reconnaissance (ISR), and missile defense) across space, air, land, underground, sea surface, and underwater domains.[1] Currently, most of the uncrewed platform operations are carried out in structured or known environments with the human in the loop. For example, uncrewed underwater vehicles (UUVs) are widely used for surveillance and mapping applications. UUV operators plan missions and calibrate the sensors, by studying the bathymetry and environmental properties, such as conductivity, temperature, and depth in the operating environment. In most of the current operations, UUV operators track the UUV's position

[*] Senior Scientist, Sensing Perception and Applied Robotics Division, Charles River Analytics, 625 Mt. Auburn St. Cambridge, MA 02138.
[†] Scientist, Human-Centered AI Division, Charles River Analytics, 625 Mt. Auburn St. Cambridge, MA 02138.
[‡] Scientist, Human-Centered AI Division, Charles River Analytics, 625 Mt. Auburn St. Cambridge, MA 02138.

while it is executing its mission underwater and have emergency abort devices to recover the vehicle in case of any unexpected changes to the environment or mission objectives. The real benefits of uncrewed platforms can be realized only if these platforms are able to adapt to changes in the environment and mission objectives with minimum operator intervention. These platforms can operate in unknown, highly complex, and dynamic environments only by improving their onboard autonomy algorithms. Operators must be able to trust these autonomy algorithms.[2] This becomes a challenge due to unreliable sensors and harsh environmental conditions. For example, acoustics sensors are the most effective option for underwater use; however, their performance is degraded due to continuous changes in acoustic properties in the operating environment. Similarly, continuous changes in water density due to mixing and underwater currents make it hard for these platforms to operate effectively in these highly dynamic environments. As a result, there is a high demand for autonomy algorithms to include resource management, maintenance, and forecasting algorithms for these platforms to operate reliably and interact with human operators only when they need assistance. This paper looks at some of the autonomy frameworks commonly used for uncrewed platforms in highly dynamic environments and discusses how onboard autonomy can be made adaptable/trustable to uncertain sensors, unexpected faults to the platform (e.g., hardware failure or software resource limitations such as battery, memory, or processing power), environmental changes, and changes to mission objectives.

We will present some of the commonly used behavior-based autonomy frameworks and then discuss in the following sections how these frameworks are been implemented in uncrewed platforms to make them adaptive to any unexpected internal and external changes.

REVIEW OF BEHAVIOR-BASED AUTONOMY FRAMEWORKS

An autonomous system should be able to select among multiple possible action sequences to successfully achieve its goals. Behavior-based autonomous system research has been inspired by biological organisms, with the aim of understanding and implementing basic, survival-related behaviors in uncrewed platforms, and advanced behaviors involving, for example, high-level reasoning. Most of the behavior-based autonomy architectures are layered in such a way that the capabilities of the uncrewed platform are broken down into individual behaviors, as illustrated in Figure 1.

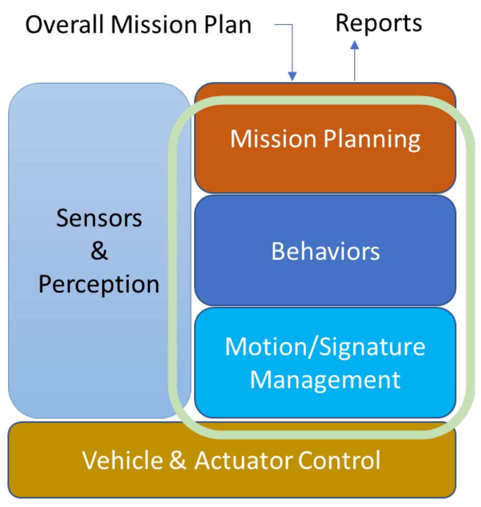


Figure 1: Behavior-Based Layered Architecture Onboard an Uncrewed Platform.

As shown in Figure 1, sensors mounted on an uncrewed platform and its perception subsystem will create situational awareness (i.e., internal (self-awareness) and external (state of the environment) states). The mission planning subsystem will then analyze the situation with respect to the overall mission at hand to determine the current mission objectives. The mission planning subsystem will activate behaviors to achieve the current mission objectives. Each active behavior will determine the best course of action for the given situation and mission objective. However, all the behaviors will be competing for resources onboard the uncrewed platform and the motion/signature management system needs to identify the best action for each moment and command the vehicle and actuator control systems. An example of a layered autonomy framework used onboard a UUV is shown in Figure 2.

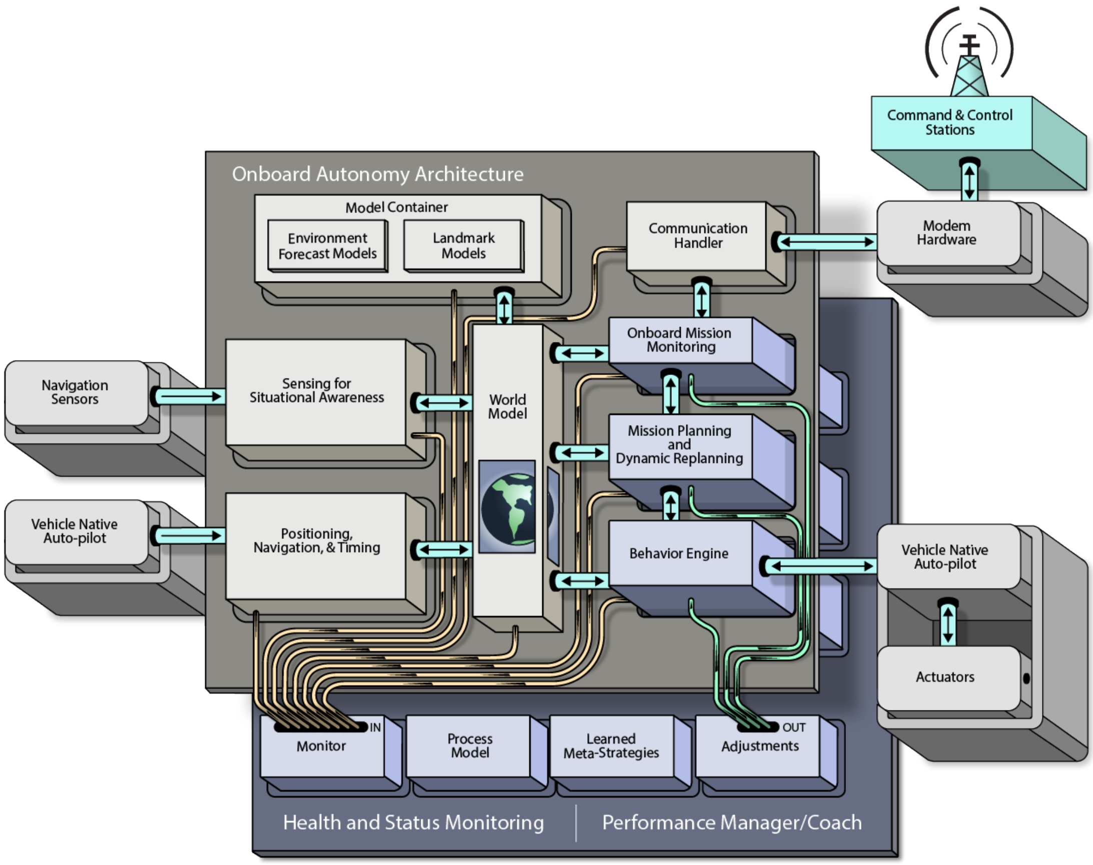

Command & Control Stations
Onboard Autonomy Architecture
Model Container
Environment Forecast Models
Landmark Models
Communication Handler
Modem Hardware
Navigation Sensors
Sensing for Situational Awareness
World Model
Onboard Mission Monitoring
Mission Planning and Dynamic Replanning
Vehicle Native Auto-pilot
Positioning, Navigation, & Timing
Behavior Engine
Vehicle Native Auto-pilot
Actuators
IN
Monitor
Process Model
Learned Meta-Strategies
OUT
Adjustments
Health and Status Monitoring
Performance Manager/Coach

Figure 2: An Example of a Behavior-Based Autonomy Architecture.

A typical UUV is fitted with internal–external situational awareness sensors and position, navigation, and timing (PNT) sensors such as inertial sensors, GPS, and compass. Commonly used sensors include internal environmental sensors for health and internal status monitoring of the following: pressure, leak, humidity, temperature, pressure, and current/voltage sensors. External sensors monitor a range of factors, conditions, and equipment, such as conductivity, temperature, depth (CTD); Doppler velocity log (DVL); sonar (active–passive); camera; pressure; and water current. As shown in Figure 2, UUVs run numerical environmental forecasting models to keep track of the environmental changes. For example, *reachability front behavior* will keep track of the underwater currents to optimize the path to its destination, as shown in Figure 3.

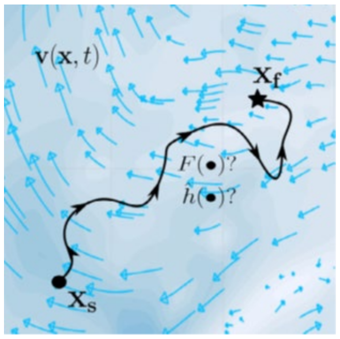
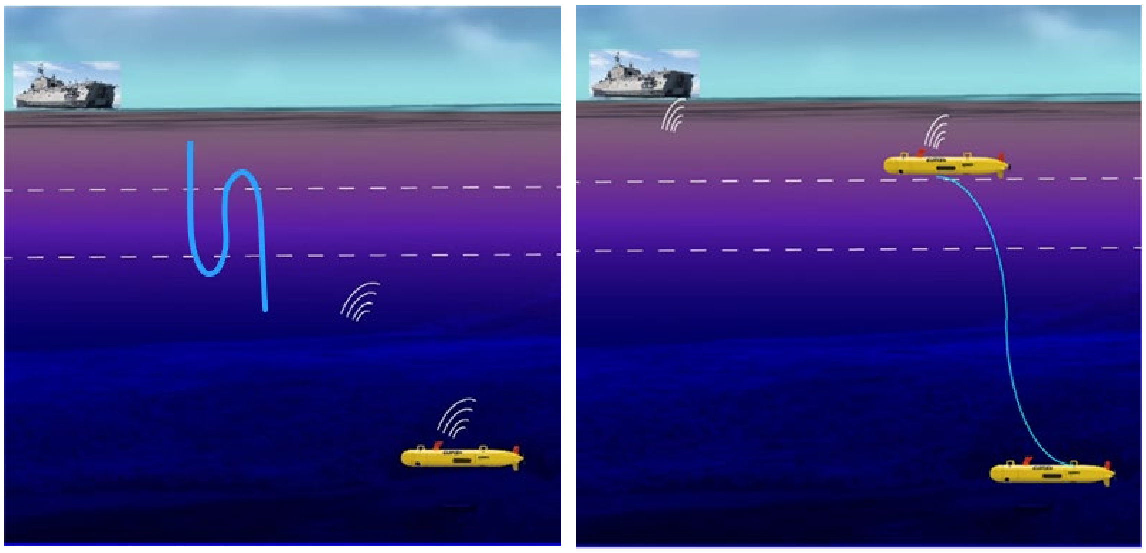


Figure 3: Time-Optimum Path Calculation to Travel from X_s to X_f under Ocean Currents v(x,t).[3]

UUVs need to be conscious about the changes in the environment. For example, due to the range and attenuations, sound waves are the most successful medium used underwater for communication and sensing (lessons learned from the biological systems). However, underwater acoustic properties are constantly changing, which impacts the waveguide properties, and it is important that UUVs are aware of these changes to sense the environment and communicate with the surface or other UUVs, as illustrated in Figure 4.



(a) *ThermoclineDetection* behavior (b) *SurfaceCommunication* behavior

Figure 4: Example of an Adaptive Behavior Used by a UUV to Establish Successful Acoustic Communications with Surface Platforms.[4]

Figure 4 illustrate how a UUV will move in the water column ("yo-yo" motion) to identify the temperature gradient; the region with rapid decrease in temperature over depth is called the thermocline. The speed of sound changes with the temperature and a thermocline may act as a wall blocking any acoustic messages; thus, it is important that the UUV be positioned above the thermocline if it wants to communicate with a surface platform (e.g., ship). Other commonly used UUV behaviors are listed in Table 1.

Table 1: Examples of Common UUV behaviors

Native Behaviors	Description
ConstantDepth	UUV maintains a constant depth
ConstantAltitude	UUV maintains a constant height above the seafloor
WayPoint	UUV travels to its next waypoint
OpRegionBounce	UUV stays inside its operation region
Loiter	UUV loiters around a waypoint
ObstacleAvoidance	UUV avoids obstacles in front of the vehicle
YoYo	UUV changes depth between two levels
DetectThremocline	UUV detects the thermal gradient of the water column
TimeOptimum	UUV follows the optimum path generated by the reachability front

In this paper, we discuss two challenges faced by behavior-based adaptive autonomy systems:

1. Multi-objective Optimization (MOO): As explained above, multiple behaviors active onboard uncrewed platforms will be competing for onboard resources (actuators, processing power, battery power, etc.), and it is important to select the best action (optimum) for a given situation to successfully achieve mission objectives.

2. Performance management: Due to unexpected environmental changes or faulty hardware, the performance of the uncrewed platform can change during mission execution. For example, the power budget of an uncrewed aerial platform can change due to heavy wind conditions.

In the next section, we will present some of the popular MOO and performance algorithms used by uncrewed platforms to identify the best action for unexpected situational changes.

CHALLENGES IN BEHAVIOR-BASED ADAPTIVE AUTONOMY

There are many challenges for uncrewed platforms operating in unknown environments for a long period of time without human intervention. As explained earlier, uncrewed platforms can navigate highly complicated environments for exploration or for executing various tasks, keeping humans away from these unfriendly environments (e.g., deep-ocean environments, space, disaster areas, contested environments, etc.).

Challenge 1: Communication. In many applications, there is no or limited connectivity to the uncrewed platform from the command and control (C2) stations. Therefore, it is necessary that the onboard autonomy can adapt to the unexpected changes in the environment as well as for any changes within the uncrewed platform.

Challenge 2: Sensor uncertainties. The situational awareness (operating environment and the internal state of the platform) of an uncrewed platform is determined by the perception subsystem using various sensors. However, due to complex environmental conditions, sensors can be noisy. For example, sonar sensors used underwater will have clutter due to the changes in the acoustic properties of the environment as well as other factors such as multipath, and signal attenuations, etc. These uncertainties need to be considered when reacting to the determined situation.

Challenge 3: Hardware constraints. Due to harsh environmental conditions, there is a higher likelihood of the hardware's performance deteriorating, and the autonomy system need to constantly monitor the health of the hardware systems.

Challenge 4: Highly dynamic environment. As discuss earlier, uncrewed platforms are often deployed into highly complex environments where severe weather and environmental conditions can create unexpected changes to the operating conditions. The autonomy system should be aware of its own capabilities and be able to adapt its mission plans to unexpected environmental changes. For example, a mission plan might direct the UUV in a certain direction; however, due to heavy underwater currents, even with full forward thrust, the UUV might find itself moving backward. The autonomy system should understand the situation and adapt the plan to change direction and follow a reachability front.

Challenge 5: Limited power budget. Uncrewed platforms are battery powered and missions are planned for a known power budget. However, due to the complex nature of the environment, power consumption may be higher than expected, making it challenging for the platform to complete its tasks. Autonomy systems should keep track of their power availability and adapt their mission plans accordingly.

Challenge 6: GPS denied or unreliable PNT. Knowing the absolute position of the platform is important to navigating in the operating envelope. However, there are many operating domains where GPS is not usable or not available (e.g., underwater). Most of these mobile uncrewed platform navigation decisions will depend on the accuracy of the vehicle's navigation system. The uncertainties associated with the PNT system need to be considered when making mission autonomy decisions.

There are many groups who have worked on developing autonomy frameworks and have implemented them in various types of the uncrewed platforms for various applications. Some of the popular layered autonomy architectures are MIT Mission Oriented Operating Suite – Interval programming (moos-ivp), JPL Control Architecture for Robotic Agent Command and Sensing (CaRACAS), Draper Maritime Open Architecture Autonomy (MOAA), JPL CASPER/ASPEN, CMU UC, etc. These frameworks are effective for static, known environments and for short (on the order of a few days) mission periods. The following sections discuss how MOO and learned meta-strategies can be used to help uncrewed platforms adapt to and handle unexpected situations during mission execution with little or no human interference.

MULTI-OBJECTIVE OPTIMIZATION

In this section we will provide an introduction and motivation for MOO in mission planning, and we will give a brief survey of the state-of-the-art approaches for multi-objective mission planning,

Introduction to MO Mission Planning

In practice there are many competing objectives that describe "success" in a mission. For example, the mission should ideally be completed in a short amount of time, minimize the amount of battery consumption, minimize risk exposure over time, maximize the amount of useful information gained or the value of the kinetic effect deployed, etc. Even in relatively simplistic environments these objectives are often at odds with one another. Consider for example a mission where the goal is to get from point A to B quickly *and* avoid detection by an adversarial vessel. Minimizing the amount of time required to get from A to B may incur great risk (e.g., if the adversary were situated directly between A and B), while minimizing or eliminating risk may require the vessel to get from A to B using a very indirect and therefore suboptimal (with respect to time) path.

In general, there is no single solution (mission plan) that simultaneously optimizes all objectives. A solution is called Pareto efficient if value of any objective cannot be improved without decreasing the value on some other objective (put another way, a solution is Pareto efficient if no other solution achieves a better value on every objective).

The goal of multi-objective optimization is to compute (all or some of) the Pareto-efficient solutions for a given multi-objective optimization problem, and/or the trade-off curves between arbitrary sets of objectives. This requires formalizing the problem as an optimization model and defining a solution space, data (for training, if applicable,) metrics, objectives, constraints, and an algorithm (or policy) for solving that model.

With no other context or subjective preferences across the objectives (e.g., risk tolerance of the mission) all Pareto-efficient solutions are considered equally good. Therefore Pareto-efficient solutions, or trade-off curves, are then fed to the upstream decision-maker, who applies their own instance-specific preferences in determining which solution to pursue. This last step is supported by the broader field of multicriteria decision analytics, which is concerned with studying the positive and negative consequences, risks, and opportunity costs of decision alternatives and helping a decision-maker quantify their subjective preferences when making analytically backed and justified decisions. In our work, and in most of the works surveyed in this section, we assume that we are either providing the Pareto-efficient set to a human operator to make the ultimate decision, or that the human operator has defined some mission-specific preferences over the objectives to facilitate autonomous decision-making.

Survey of Approaches to MO Mission Planning

There is a vast amount of literature in the space of multi-objective mission planning in autonomous systems. We will briefly summarize several popular types of approaches in each of the model-driven and data-driven artificial intelligence (AI) categories.

First, in classic model-based approaches, the goal is to encode relevant aspects of the host platform and the environment, along with their interactions, in a single mathematical model (a mathematical optimization problem) and develop algorithms to cleverly search over the space of feasible solutions to quickly find high-quality solutions. Many works develop novel evolutionary or genetic algorithms[5–7] for single- and multi-UXV mission planning problems. These consider objectives like task allocation, timeliness of mission completion, collision avoidance, etc. Evolutionary and genetic algorithms are a form of search that propose novel solutions by performing some novel kind of mutation to older seed solutions and mimic the process of natural selection.

Wu et al.[8] model the UXV mission planning problem as a path selection problem on a weighted graph. The weights on the graph are objective scores associated with including a vertex or edge in the final solution and consider collision avoidance, corridors, rules of visual navigation, timeliness, and fuel constraints. They apply scalarization techniques to turn the multi-objective problem into a single-objective problem and use single-objective A* algorithm variants for efficiency. This approach is particularly robust as graphs are a fully expressive encoding scheme (i.e., one can encode a wide range of mission planning problems), and single-objective path planning algorithms are extremely efficient. Our approach to Management of Intelligent Navigation for Condition-based Ocean Safe Transit (MIN-COST) was directly inspired by this kind of approach. We take advantage of the efficiency of single-objective path planning problems to build an architecture that automatically searches objective weighting schemes to identify (in the single-objective scalarized space) high-quality solutions to the multi-objective problem. We postprocess the result after the search is completed.

Other approaches consider multitiered architectures that break the overall mission planning problem down into multiple optimal estimation and planning problems to come up with robust solutions in dynamic environments. These include modeling power systems and predicting energy consumption over time to feed in more aureate estimates to the energy optimization loop handled by evolutionary algorithms (among others).[9,10] Ding et al.[11] consider a stochastic hierarchical planning representation of mission planning to forecast motion while trading off mission objectives in cluttered environments with dynamic events. They show that stochastic hierarchical planning operates robustly in uncertain environments and supports contingency management in real time.

In data-driven AI the goal is to train a reinforcement learning (RL) agent to learn a continuous action policy that produces good behavior across a variety of situations. This requires building a reward function (usually within a simulation or labeled data set) that provides a multi-objective reward signal to an agent given an arbitrary vehicle and environmental state. Using reinforcement learning techniques, an agent is programmed to interact with the environment to try and accomplish its task and achieve an appropriately balanced score across objectives. After sufficient training, an agent learns successful sequences of plan fragments as a function of the state. See Tang et al.[12] for a survey of RL approaches and state-of-the-art models.

LEARNED META-STRATEGIES

Various meta-strategies based on machine learning can be effective for enabling autonomous systems to quickly adapt to changes in the environment. In this section, we discuss performance management in general and within the context of an approach to adapting the mission plan of an autonomous UUV to conserve power after an unplanned energy drain (e.g., due to battery failure).

Performance Management for Autonomous Systems

Autonomous systems can be complex with many different components (e.g., path planner, localization, and communication components in a UUV) acting together to achieve mission goals. An approach to performance optimization of the overall system depends upon optimizing the performance of each individual component and their interactions against the mission objectives. To do so, it is helpful to define the *intent* of each system component within the mission context. There are various ways to define intent, as investigated under DARPA.[13] Here, we focus on an approach that defines a set of *intent metrics* and *thresholds* for each system component.

Intent, for a particular component implementation, can be defined either during the development process or after the fact, but requires knowledge about the semantics of the component within a particular mission context. An intent metric for a component is a quantity that can be measured by the monitoring infrastructure of the autonomous platform. These can be continuous measurements or measurements taken after a certain task is completed. For example, an intent metric for an inertial localization component on a UUV platform would be the localization error measured when this information is available (e.g., when the vehicle moves closer to the surface and can access GPS). Another intent metric for the localization component could be an estimated error value (e.g., computed via a Kalman filter) that is computed periodically. An additional example is a waypoint planning component whose intent metric is proportional to the energy consumption resulting from traversing each waypoint. The power usage could be measured periodically during traversal, once reaching each waypoint, or after the vehicle has traversed all waypoints and returned home.

Once intent metrics are defined, intent thresholds can then be applied to them based on mission parameters and available resources. In the localization component example, the mission may require that the estimated and/or the ground truth localization error be maintained under a specified value; otherwise, an *intent violation* occurs. Similarly, an intent violation would be signaled if the sequence of paths specified by the waypoint planner consumes more than a specified amount of energy per unit time. An intent violation signals a system or component-level optimizer that will then take appropriate actions (e.g., adjust navigational/localization parameters, replan waypoints based on different constraints) with the goal of restoring the intent metric values to mission-acceptable levels.

Given intent metrics and their associated thresholds for a collection of system components, an *intent optimizer*, acting autonomously, adjusts the parameters of each component in the system upon each intent violation to bring metric values back within mission requirements. The decision of exactly how to modify component-local and system-wide parameters that modify and control the behaviors of the autonomous system is a complex one, since there could be many interdependencies on the operations and performance of different components within the system. For example, increasing the sampling rate of a sensor filter could increase the power draw due to increased computational load, causing an intent violation for a waypoint planner that attempts to minimize energy use during a mission. These cascading intent violations could be resolved through organizing the system into a component hierarchy defining the *performance* dependencies between settings for component parameters and imposing an ordering on parameter adjustments that will ensure that all intent violations can be resolved without entering an unstable state (e.g., forever "ping-ponging" between intent violations for different components). Therefore, in complex systems with potentially circular performance (i.e., intent) dependencies between component configurations, another approach is needed that can learn these dependencies and enforce higher-level policies on combinations of parameters across all system components.

Machine Learning Approaches to Performance Management

To autonomously and dynamically coordinate adaptation of system component parameters to satisfy intent as described above, we use machine learning techniques based on deep neural networks and reinforcement learning. These techniques can be applied either on a per-component basis (i.e., for adjusting the parameters of a waypoint planner to ensure that the vehicle can return home safely without running out of energy, given the current power profile), or on a system-wide basis if there are nontrivial performance dependencies between component configurations.

Under the Building Resource Adaptive Software Systems (BRASS) program, sponsored by DARPA, we investigated optimizing power usage against objectives for a mission in which an autonomous UUV must search a region of the seafloor for an object. In our experiments, we managed the performance of two different components in a simulation of a UUV system and its environment: (1) a waypoint planner (referred to as a path planner in the reference) that constructs waypoints while executing a "lawnmower" pattern that provides a level of coverage over the search area and (2) a Kalman filter implementation that filters localization sensor readings. These components work together to navigate the vehicle within the search area. We assume the vehicle is equipped with a vision sensor (e.g., camera) that can detect the object if it is within the sensor's limited field of view. The system also includes a capability to monitor the available energy. The primary mission goal is to find the object and return home safely without running out of energy. A secondary goal, if the primary goal cannot be met, is to return home safely despite not finding the object. If the vehicle runs out of energy and cannot return, then the mission fails. To perturb the state of the system, we simulate a battery failure rate that reduces the remaining energy by a random proportion after a random amount of time has passed form the start of the mission. In practice, this immediate energy drain could be a result of environmental factors, such as heat/pressure changes, damage caused by collisions with obstacles/debris, or unexpected power draw from faulty hardware.

To optimize performance to meet the mission objectives, we monitor an intent metric of the waypoint planner that measures the current energy available and compares the results to the expected energy needed to traverse the current waypoint plan. When an intent violation occurs (i.e., the energy needed is greater than the energy available), we adjust the inputs to the waypoint planner that control the shape of the lawnmower pattern that the vehicle will navigate. Note that, in our experiments, we used an automated approach to transform the original code of the waypoint planner to expose new parameters that enabled us to further configure the component, but without semantic knowledge of the direct effects of these new parameters. In many cases, however, components are designed to be highly configurable, and this step is not needed. Similarly, we monitor intent for the Kalman filter component, and perform the same adjustments of parameters to ensure that localization information remains accurate to a given degree.

Upon discovering the intent violation of the waypoint planner, we use a multilayer feed-forward neural network, which we trained on a variety of scenarios within the mission context (e.g., different search regions, search object locations, power perturbations, water currents), to determine the appropriate parameters to the waypoint and Kalman filter components, based on the observed state of the system and the environment (e.g., power available/required, current estimated position error, ground truth position if available at periodic intervals). We then pass the parameters output by the neural network into the waypoint planner, which computes a new route. The new parameters, in our experiments, increased the distance between legs of the lawnmower pattern to reduce the total path length and thus the power necessary to traverse the search area and return home, at the expense of coverage of the vision sensor over the search area. This reduced the probability that the object would be found but would attempt to ensure that the vehicle could return home and not be lost (i.e., mission failure). In our analysis we differentiate between a mission "pass" (finding the object and returning home) and a "degraded" (returning without finding the object) outcome, but we do not distinguish between failure ("failed") cases of finding the object and running out of energy, and not finding the object and running out of energy. Our results show that using our intent-based approach to performance management resulted in about the same number of pass cases (over a variety of mission instances) compared to no adaptation. However, with intent-based performance management enabled, the vehicle was able to return (pass or degraded) in almost 1.5 times more scenario instances. For additional details on our approach and analysis, including results of adapting the Kalman filter component, see Fry et al.[14] and Pfeffer et al.[15]

CONCLUSION

In this paper, we discussed challenges faced by uncrewed platforms for operations in unknown environments with unexpected situations with little or no human operator interference. We reviewed some of the solutions used to make behavior-based autonomy architectures adaptive to overcome these challenges. We discussed how multi-objective optimization algorithms and learned meta-strategies can be used to successfully complete missions by modifying mission and behavior parameters according to the unexpected changes (internal or external to the platform).

REFERENCES

[1] AUVSI. (2023, March). *Unmanned Systems & Robotics Database*. http://robotdirectory.auvsi.org/home

[2] Edmonds, E. (2019, March). Three in four Americans remain afraid of fully self-driving vehicles. *AAA Newsroom*. https://newsroom.aaa.com/2019/03/americans-fear-self-driving-cars-survey/

[3] Subramani, D. N., & Lermusiaux, P. F. (2016). Energy-optimal path planning by stochastic dynamically orthogonal level-set optimization. *Ocean Modelling*, *100*, 57–77.

[4] Petillo, S., Balasuriya, A., & Schmidt, H. (2010). *Autonomous adaptive environmental assessment and feature tracking via autonomous underwater vehicles. OCEANS'10 IEEE Sydney*, IEEE, 1–9.

[5] Lamont, G. B., Slear, J. N., & Melendez, K. (2007). UAV swarm mission planning and routing using multi-objective evolutionary algorithms. *2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, IEEE, 10–20.

[6] Peng, C., & Qiu, S. (2022). A decomposition-based constrained multi-objective evolutionary algorithm with a local infeasibility utilization mechanism for UAV path planning. *Applied Soft Computing*, *118*, 108495. https://doi.org/10.1016/j.asoc.2022.108495

[7] Ramirez-Atencia, C., Bello-Orgaz, G., R-Moreno, M. D., & Camacho, D. (2017). Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms. *Soft Computing*, *21*(17), 4883–4900. https://doi.org/10.1007/s00500-016-2376-7

[8] Wu, P. P.-Y., Campbell, D., & Merz, T. (2009). On-board multi-objective mission planning for unmanned aerial vehicles. *2009 IEEE Aerospace Conference*, IEEE, 1–10.

[9] Liu, J., Wang, W., Li, X., Wang, T., Bai, S., & Yanfeng, W. (2018). Solving a multi-objective mission planning problem for UAV swarms with an improved NSGA-III algorithm. *International Journal of Computational Intelligence Systems*, *11*(1), 1067–1081.

[10] Thompson, F., & Galeazzi, R. (2020). Robust mission planning for autonomous marine vehicle fleets. *Robotics and Autonomous Systems*, *124*, 103404.

[11] Ding, X. D., Englot, B., Pinto, A., Speranzon, A., & Surana, A. (2014). Hierarchical multi-objective planning: From mission specifications to contingency management. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 3735–3742.

[12] Tang, Y., Zhao, C., Wang, J., Zhang, C., Sun, Q., Zheng, W. X., Du, W., Qian, F., & Kurths, J. (2022). Perception and navigation in autonomous systems in the era of learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, https://doi.org/10.1109/TNNLS.2022.3167688

[13] DARPA. (2015). *DARPA BRASS Program*. https://www.darpa.mil/program/building-resource-adaptive-software-systems

[14] Fry, G., Samawi, T., Lu, K., Pfeffer, A., Wu, C., Marotta, S., Reposa, M., & Chong, S. (2019). Machine learning-enabled adaptation of information fusion software systems. *2019 22th International Conference on Information Fusion*, IEEE, 1–7.

[15] Pfeffer, A., Wu, C., Fry, G., Lu, K., Marotta, S., Reposa, M., Shi, Y., Kumar, T. S., Knoblock, C. A., & Parker, D. (2019). Software adaptation for an unmanned undersea vehicle. *IEEE Software*, *36*(2), 91–96.