Search-Based Testing Methods for Evaluating the Resilience of Autonomous Unmanned Underwater Vehicles

Galen E. Mullins, Paul G. Stankiewicz, and Melissa A. Huntley

ABSTRACT

The resilience of an unmanned underwater vehicle (UUV) can be defined as the vehicle's ability to reliably perform its mission across a wide range of changing and uncertain environments. Resilience is critical when operating UUVs where sensor uncertainty, environmental conditions, and stochastic decision-making all contribute to significant variations in performance. A challenge in quantifying the resilience of an autonomous system is the identification of the performance boundaries—critical locations in the testing space where a small change in the environment can cause a large change (i.e., failure) in an autonomous decision-making system. This article outlines a methodology for characterizing the performance boundaries of an autonomous decision-making system in the presence of stochastic effects and uncertain vehicle performance. This approach introduces a method for hierarchically scoring the autonomous decision-making of these systems, allowing the test engineer to quantitatively bound the performance prior to UUV deployment. When using this scoring approach, engineers apply a set of novel subclustering methods, allowing them to identify stable performance boundaries in stochastic systems. The result is a process that effectively measures the resilience of an autonomous decision-making system on UUVs.

INTRODUCTION

Autonomous vehicles are expected to execute complex missions with multiple competing objectives. For example, consider an unmanned underwater vehicle (UUV) acting in an unknown environment with local sensing and limited fuel. Its mission is to reach specific waypoints, maintain a safe distance from obstacles, and keep enough fuel in reserve that it can make the trip to a recovery location. Autonomy software that values fuel consumption over safety may take paths that veer too close to obstacles in order to minimize path length, or another software that prioritizes reaching waypoints may result in the UUV running out of fuel before it returns home. There are multiple ways the system can fail or act in an unsatisfactory manner. Identifying faults where a subsystem throws an error or the system fails a hard constraint, such as exceeding a speed constraint or colliding, are straightforward and well-studied areas of research.[1–3] However, many events are ambiguous and cannot be

easily coded, requiring a human engineer to make a judgment call. An example is assessing when it is appropriate for the UUV return home without completing the mission. Therefore, a methodology is needed to identify and group the many types of behaviors the autonomous vehicle can exhibit and present them to the test engineer.

Because of the large degree of uncertainty in real-world environments, autonomous vehicles must be capable of adapting to a wide variety of situations. Testing the resilience of these decision-making systems requires methods that evaluate system performance and account for stochasticity in planning, perception, and control submodules (i.e., sensor noise, plant disturbances, etc.).[2] Additionally, as the mission and the autonomy software become more complex, it is necessary to explore various emergent behaviors that arise through the combination of all system submodules. However, exploring all possible configurations of the system and the environment is simply impossible, particularly when each scenario must be run multiple times to account for stochastic effects. Thus, the limited computational budget for running simulations demands new methods for intelligently generating test cases.

To address the challenge of finding a diverse set of behavioral modes in a large testing space, the Range Adversarial Planning Tool (RAPT)[4,5] was developed. RAPT is a software framework that allows test engineers to identify safe operating envelopes for decision-making systems. The objective of RAPT is to make field tests more cost effective by finding a small set of salient test scenarios that are known to demonstrate relevant changes in the vehicle's behavior. Since field tests are expensive to run and execute, it does not make sense to run tests when the performance of the vehicle is known to be certain. Instead, RAPT focuses on finding tests that are in regions of high uncertainty that represent the thresholds of the vehicle's performance. While past works have used optimization techniques to search testing spaces for failure scenarios,[3,6] these efforts have all used the discovery of collision cases as their only objective function. The aim of RAPT is to find a variety of different behaviors across multiple mission-relevant scoring criteria without the need for a single unifying objective function that defines the robustness of a scenario. To achieve this, RAPT introduces the concept of performance boundaries, which are a structural feature of any system under test that can be identified via unsupervised learning methods.

Performance boundaries are transitional regions in the testing space where small changes in scenario parameters (e.g., obstacle positions or environmental factors) cause large changes in system performance. References 4 and 5 outline methods for identifying test scenarios in the regions of these performance boundaries. For high-dimensional spaces where exhaustive testing is infeasible, identifying the system's performance boundaries allows a test engineer to characterize the performance landscape with a limited number of simulations. As an example, consider the autonomous UUV mission introduced at the beginning of this section. The vehicle must navigate to a waypoint while conserving fuel and avoiding obstacles. Figure 1 illustrates a performance boundary that occurs when a small change in the position of the pentagonal obstacle closes a narrow channel that results in drastically different performance—the difference between obstacle avoidance and a collision. In this example, when the obstacle is moved the vehicle has more room to maneuver and avoids a collision. The vehicle makes the decision to return to a recovery point because the route around the large obstacle is estimated to be too long given the remaining fuel.

A key assumption of the methodology presented in Refs. 4 and 5, however, is that it used both a deterministic autonomy and a deterministic simulation to measure the performance of the system. As shown in this article, additional research is needed to properly handle the stochastic effects produced by sensor noise, plant disturbances, and other forms of uncertainty that result in probabilistic performance (i.e., instances where repeated tests of the same scenario produce uncertain, and potentially drastically different, outcomes).

This article addresses these shortfalls by demonstrating RAPT's ability to handle both deterministic and stochastic simulations. It begins with a brief overview

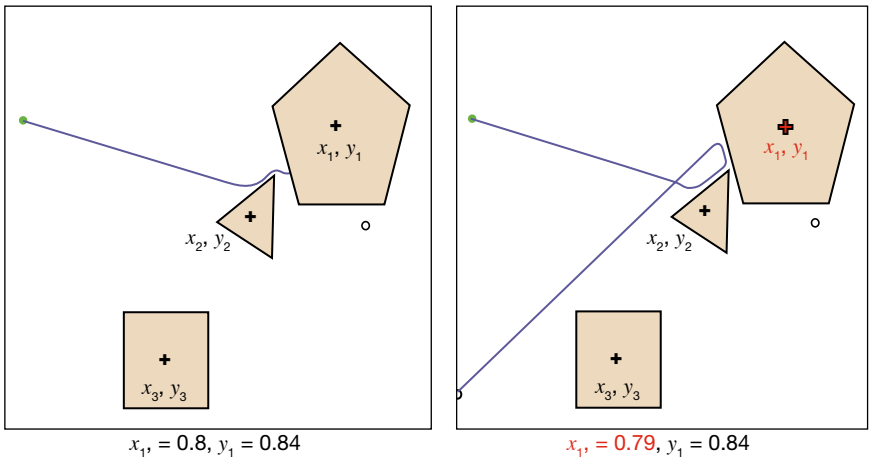


Figure 1. Example of the performance boundary for a simple UUV navigation mission. The green circle is the starting position of the vehicle, and the black circle on the right is the waypoint the UUV is trying to reach. A small change in the x_1 location of the pentagonal obstacle results in the UUV avoiding a collision and then aborting its attempt to reach the waypoint.

of the RAPT framework and our scoring methodology. It then discusses how noise in the sensing and controls of an autonomous vehicle affects RAPT's ability to discover salient test cases. Next it introduces the subclustering algorithms used to identify emergent behaviors and discusses the results of applying these techniques to our UUV simulation. The article concludes with a discussion on how these new techniques impact evaluations of UUV resilience.

RANGE ADVERSARIAL PLANNING TOOL

This section briefly overviews the methodology RAPT used to generate salient test scenarios for autonomous systems. For a full discussion of this framework and the underlying algorithms, refer to Ref. 4. See Fig. 2 for an architectural overview of the tool. The process begins with the test engineer specifying two configurations for the autonomy under test (AUT): the testing state space and the scoring space. The testing state space dictates the elements of the simulation that will be varied between scenarios, including obstacle positions, mission parameters, ocean current magnitudes, or any other factors that could influence the autonomy decision-making. The scoring space then controls how the AUT is evaluated for each completed scenario. Once the testing state space and scoring space are defined, the performance evaluation process is broken into two phases: adaptive sampling followed by boundary identification.

Adaptive Sampling

In the first step, adaptive sampling is used to generate test scenarios that give insight into the performance boundaries of the system. This iterative process builds a surrogate model of the AUT's performance based on the results of completed simulations. Inputs of the surrogate model are defined as parameters in the testing state space, and the outputs are the evaluation criteria defined in the scoring space. This surrogate model is then used to generate subsequent scenarios in regions of the testing state space where performance boundaries are expected to occur (e.g., regions characterized by high variance in the AUT performance). A key feature of the adaptive sampling algorithm is the ability to balance the trade-off between fully exploring the testing state space while also preferentially focusing samples on regions of predicted performance boundaries. The best-performing version of the adaptive sampling algorithm used the nearest-neighbor density and variance (NNDV) meta-model. By searching for areas that predicted high variance but were far away from previous samples, it generated samples in the regions of interest more effectively than any other technique.

Boundary Identification

After all the simulations have been executed, the second step is to use the raw simulation data to characterize the performance boundaries of the system. However, because these performance boundaries are highly nonlinear and exist in high-dimensional spaces, they are not easily characterized analytically. Thus, their structure is inferred based on adjacent scenario pairings that exhibit distinct differences in performance, where groupings of performance classes are identified using unsupervised clustering methods. These scenario pairings along the system's performance boundaries are then returned to the user. Ultimately, scenarios that straddle performance boundaries aid in diagnosing the AUT's decision-making by attributing small changes in scenario parameters to large transitions in performance.

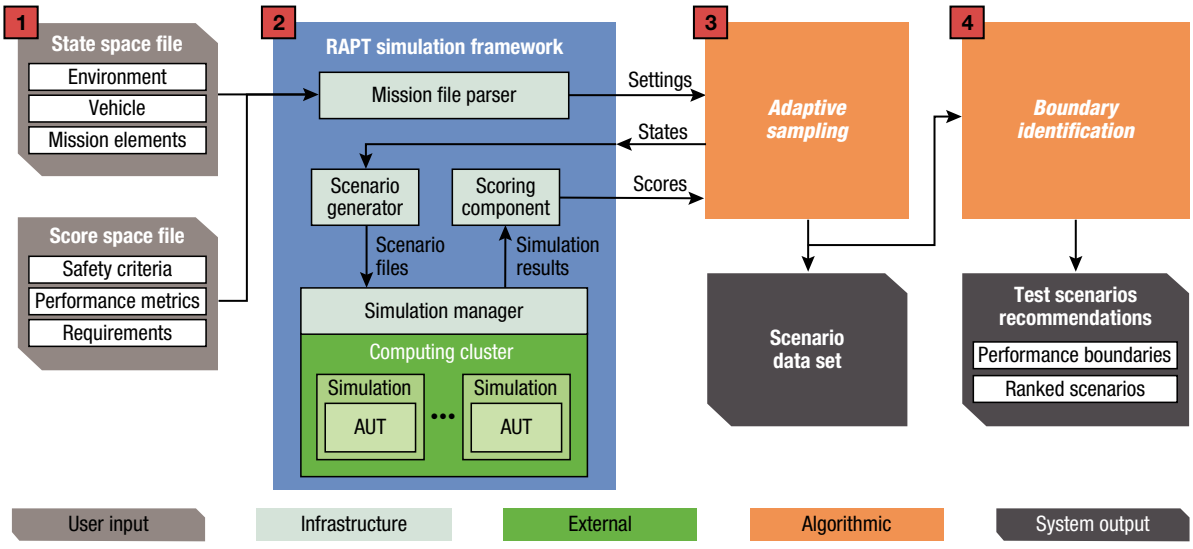


Figure 2. Overview of the RAPT framework.

Scoring Methodology

The RAPT framework is domain agnostic and can accept any parameterized simulation and set of scoring criteria. It gives the user the capability to design studies based on a combination of a testing space and performance metrics that define a mission. The user defines a score tree structure that describes the relationship between continuous metrics extracted from the simulation (e.g., distance from an obstacle) and the resulting scores (e.g., whether a collision occurred). These relationships can be any mathematical expression but are traditionally a combination of logical statements and thresholds. For example, the vehicle must achieve a certain distance from a waypoint to achieve a success at that objective and must achieve success at all objectives to succeed at the mission. Returning to the UUV example described in the introduction, there are two primary criteria to evaluate the vehicle performance: mission success and safety success. Mission success is a binary score representing that the vehicle completed all the objectives of the mission, such as reaching waypoints and completing surveys. It is computed of continuous submetrics such as the distance from each waypoint and the time in which the vehicle completed the mission. Safety success is a binary score representing that the vehicle has not violated any safety constraints—for example, that it avoided obstacles and reached the recovery point successfully. The score-tree structure used for this UUV mission is shown in Fig. 3.

The score tree is a flexible and intuitive way of describing the performance of an AUT. Performance boundaries are derived from the scores used to evaluate the mission. Selection of these metrics will affect the types of boundaries RAPT discovers. Using more metrics increases the possibility of finding all boundaries for a given study, but it also dilutes the search. Therefore, searching over a small number of score criteria results in a more focused search, and test engineers can use the saved subscore and metric information to discover the causes of performance boundaries during post-processing and analysis.

PROBABILISTIC PERFORMANCE BOUNDARIES

Whether the result of error in the sensor inputs, stochasticity in the vehicle dynamics, or random processes inside the autonomy software, there is an inherent level of uncertainty when executing a scenario in the real world. A resilient UUV must be capable of operating within a large range of scenarios and should not be sensitive to small errors or minor changes to the environment.[7] Testing the resilience of an autonomous vehicle requires a simulation environment that incorporates the stochastic properties of both the environment and the platform. This means that the test-generation algorithms must be robust against probabilistic effects. This section discusses the results of applying our test-generation algorithms to a system with noisy output.

Uncertainty in the UUV Simulation

The UUV simulation and autonomy software used in the previous RAPT work shows that a relatively simple mission can still exhibit multiple informative performance boundaries. This simulation models an IVER underwater vehicle equipped with a sonar sensor with a 100-meter range and a 120° field of view and an inertial navigation system (INS). The vehicle must reach one or more waypoints, avoiding obstacles as well as no-go areas, and then reach a recovery point with sufficient fuel remaining. The no-go areas are known in

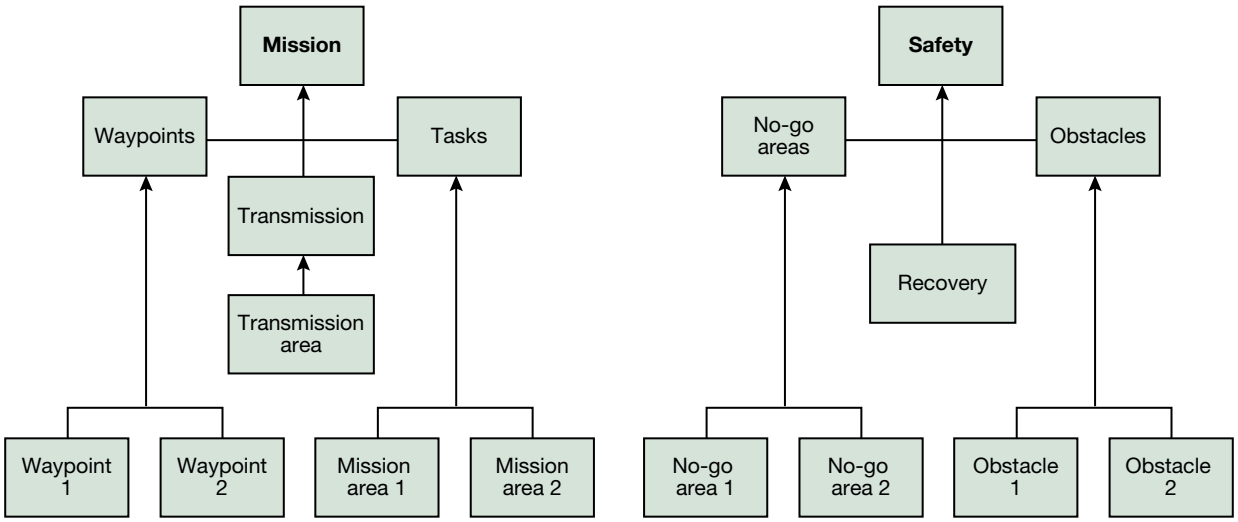


Figure 3. Diagram of the hierarchical score trees for our UUV simulation, illustrating the first three levels of the score tree for both mission success and safety success.

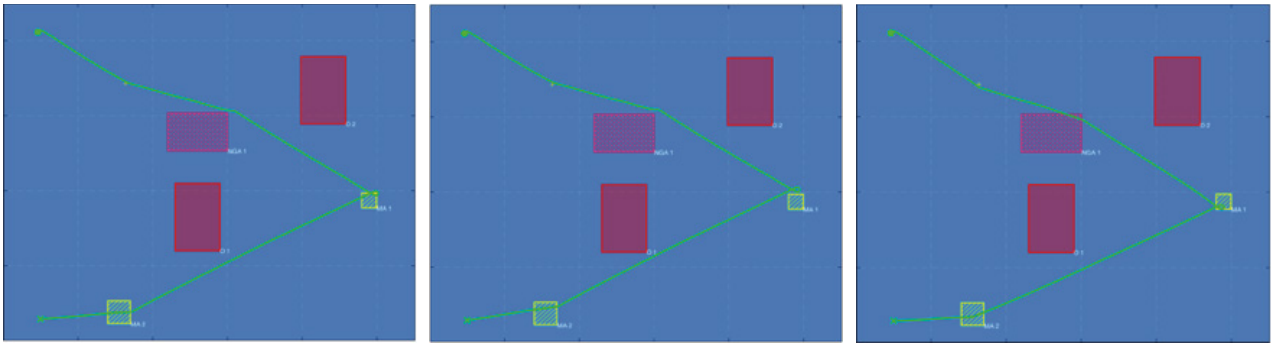


Figure 4. Three runs of an UUV scenario designed to evaluate the effects of noise.

advance but cannot be detected by the sonar; they can be avoided only by using the INS. The obstacles are not known in advance but can be detected by the sonar. The testing space is designed such that the vehicle has just enough fuel to complete the mission if minimal obstacles are detected and must abort if safety thresholds are encountered.

To add realistic uncertainty to the UUV's dynamics, the RAPT simulation incorporates stochastic models that account for difficulties in navigating underwater, including perturbations of the ocean current and drift in the INS.[8] To understand how these changes to the simulation will affect the test scenarios and boundaries RAPT generates, it is necessary to characterize the effect that this type of noise has on the output of the system. This characterization allows the creation of a synthetic function that has the same properties as the system under test.

For noise characterization tests, 40,000 scenarios in a five-dimensional state space were generated. These scenarios used start time, obstacle latitude and longitude, and no-go area latitude and longitude as the parameters to be varied. For this study, each scenario was run 10 times. The mission objective was to avoid obstacles and no-go areas and explore two survey areas before returning to the recovery point. The vehicle had the ability to detect obstacles by using its sonar sensors but had to rely on its state estimate to avoid no-go areas and reach the mission areas. The mission areas were deliberately made small for this test to increase the overall difficulty. An illustration of a representative scenario run three different times is shown in Fig. 4.

It is clear from these examples that the amount of error in the vehicle's state estimate is directly responsible for deviations in the trajectory that cause it to either travel through the shaded no-go area or miss the mission area. The vehicle's autonomy software fails to account for this error when performing its path planning and thus fails under low levels of estimation inaccuracy. The effect of the error is most strongly evidenced when plotting the relationship between the latitude of the no-go area with the distance of the closest point of approach; see Fig. 5.

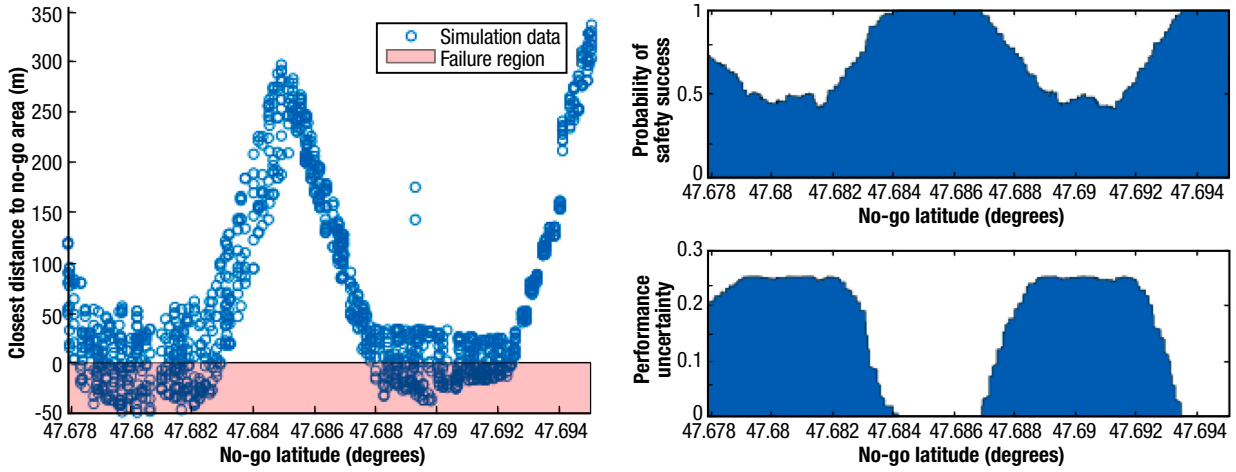


Figure 5. Left, Simulation results for the distance to no-go area metric are distributed with Gaussian noise around an unknown mean function. Top right, Plotting the binary success/failure metric leads to varying probabilities of success. Bottom right, The locations of the performance boundaries can be seen as regions where there is high uncertainty in the performance criteria.

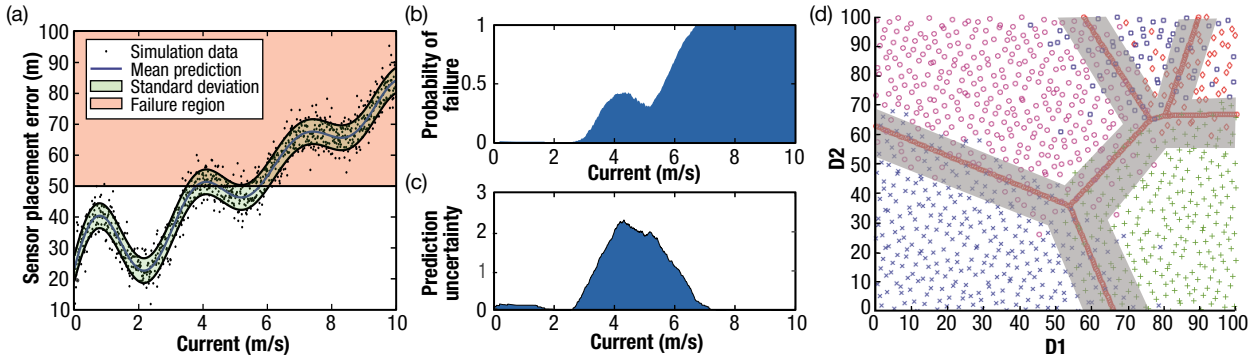


Figure 6. (a) Data points from the simulation for this continuous metric are distributed with Gaussian noise around a modeled mean function. (b) Plotting the binary success/failure metric leads to an increasing probability of failure as current increases as well as a region of high uncertainty in the binary metric (c) in the boundary region. (d) A scatterplot of the noisy synthetic function with boundary width $\sigma = 0.1$. The colored markers indicate the different classes, while the red lines indicate the true performance boundaries. The gray-shaded region indicates the area within 1 standard deviation of the boundary.

Effects of Noise on Adaptive Sampling

To perform a more rigorous analysis of the effects of noise, two synthetic systems were created by modeling different outputs of the stochastic UUV simulation. Each of these systems were parameterized to accommodate varying levels of noise. The first of these was a system with continuous output modeled after the sensor placement accuracy of the simulation. The second of these systems was a binary output model of the safety success metric. In each case, the noise variable was represented with a Gaussian distribution. The first system is a continuous function with a standard deviation of σ_m illustrated in Fig. 6a. The second results from applying a binary threshold to the first system, resulting a probabilistic performance boundary with a width of σ_c. This generated both one-dimensional and two-dimensional synthetic systems that shared the same properties as the UUV simulation. Instead of a sharp boundary between categorical outputs, the boundary was a region where two Gaussian distributions overlapped. These systems are illustrated in Fig. 6.

The probability of obtaining a sample of an incorrect performance mode is given by the equation

$$P(\sim C \mid x) = 0.5 * e^{-\frac{d(x)^2}{2*\sigma_c}}, \quad (1)$$

where $d(x)$ is the distance of the sample x from the nearest boundary. An illustration of this function is shown in Fig. 6d, with the true boundaries shown as red lines and the standard deviation from the boundary shown as a gray-shaded region. The width of this region was described via the standard deviation σ_c of these distributions and was varied between 0 and 0.4.

We used these systems to compare the adaptive sampling algorithm NNDV[5] with a commonly used uniform sampling method known as a Sobol design, using two metrics of performance boundary quality. The first is boundary precision, the percentage of the total samples taken that lie within 0.01 units of the true boundary location. The second is boundary coverage, the percentage of the boundary region that has been sampled. The results of this experiment are shown in Figs. 7 and 8.

While the resulting boundary regions are wider than in the original deterministic function, the NNDV search successfully samples the correct regions and returns tighter boundaries than the Sobol set. As boundary width increases, the NNDV search begins to degrade in performance. Once the standard deviation of the boundary region reaches 0.35, the adaptive search begins sampling in the same space-filling manner as the Sobol design. At this point the entire search space becomes probabilistic as all the performance modes

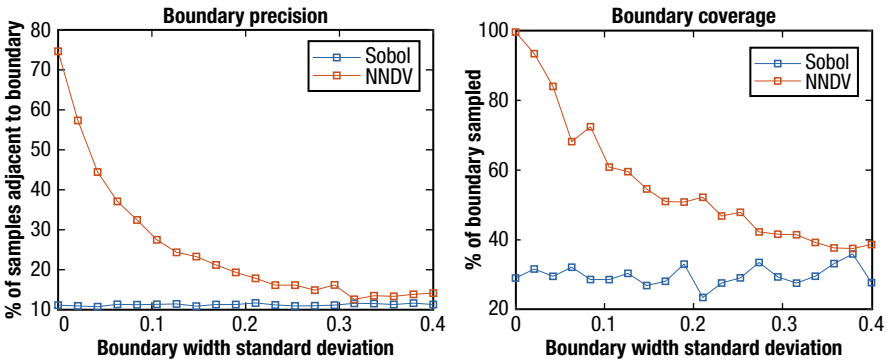


Figure 7. Plots for the boundary precision and coverage for each of our search approaches as the boundary width increases.

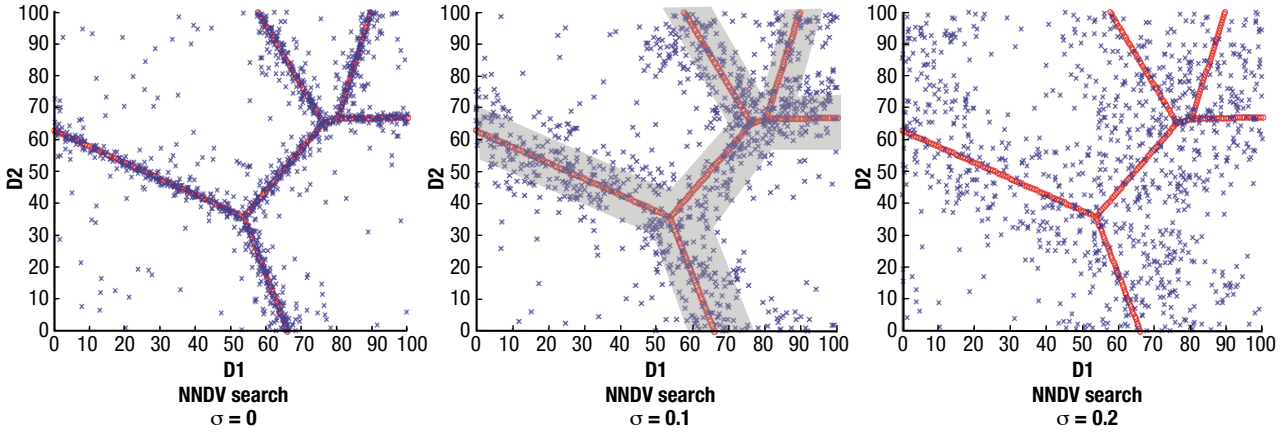


Figure 8. Scatterplots showing the effect of increasing boundary width on our adaptive search.

overlap, causing the entire system to consist primarily of noise. As such, there is insufficient information for the adaptive search to exploit, and defaulting to a global search approach is appropriate.

Effects of Noise on Boundary Identification

For the UUV system, boundary identification becomes difficult in some cases because of the effects of noise. Consider a system where there are two simplified performance modes: success and failure. Significant noise could cause the entire region that contains this performance mode to become probabilistic. This effect can clearly be seen in Fig. 9, a scatter plot of both the performance modes and the boundaries between them. Each of the points in the plot represents a single scenario. In the top plots, each scenario is given a color based on one of four possible score modes accounting for the permutations of mission success and safety success. In this example, the red points indicate the areas where the vehicle collides with an obstacle or no-go area, while

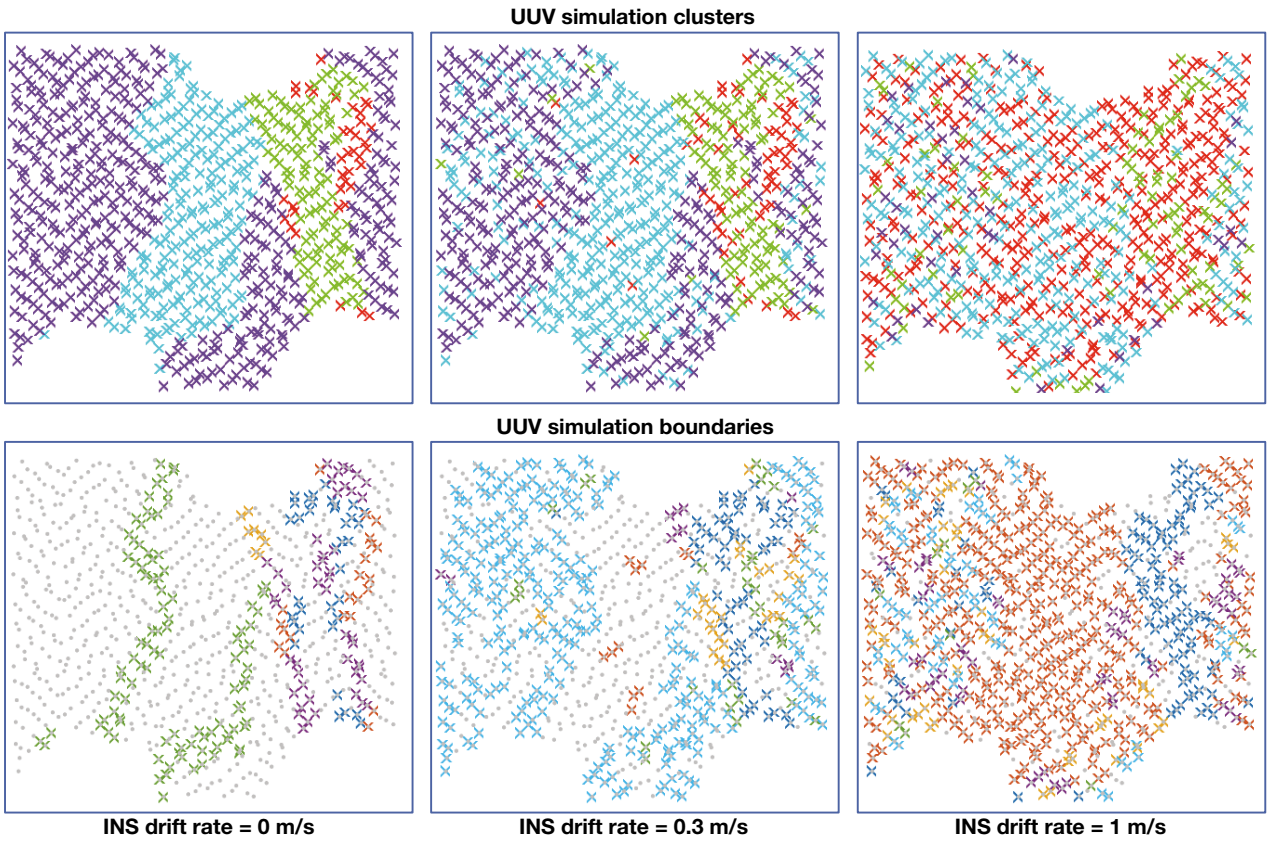


Figure 9. Plot showing the effect of stochasticity in the form of increasing INS drift rate on the performance modes and boundaries.

the purple points indicate scenarios where the vehicle completes the mission with no violations. As INS drift increases, the vehicle becomes increasingly likely to fail: first the mission criteria, as the vehicle fails to correctly reach the waypoint, and then the safety, as it collides with obstacles and no-go regions more frequently. The bottom plots show the locations of each scenario on a performance boundary as a colored *x*, with other scenarios shown as gray dots. In the far-left plot, the boundaries show up as easily distinguished lines of colored points, but as the INS drift rate increases these become broader and more scattered. This continues until the entire testing space is categorized as a performance boundary.

While the RAPT algorithms are robust to noise and will accurately find and label boundary regions as they exist, when stochastic effects become too large these boundaries cease being accurate indicators of changes in the autonomous vehicle's behavior. Therefore, it is necessary to look at another way to measure the system's performance. One method is to use the continuous valued metrics rather than the binary criteria that result from applying thresholds.

SUBCLUSTERING

When the system under test exhibits probabilistic effects, binary success criteria are not the best indicators of a change in the vehicle's behavior. Sometimes it is better to use continuous metrics, such as closest distance to a waypoint or fuel consumed, because they are affected more by large changes in the vehicle's trajectory. Therefore, a set of analytical tools that can explore the boundaries for many different possible scoring metrics is required. This section covers the hierarchical boundary identification method, which allows faster computation and retrieval boundary information for any scoring metric in the simulation.

Background

For decades hierarchical clustering has been one of the most popular methods for organizing data into sets and classifying data.[9] There are two broad categories of hierarchical clustering techniques: agglomerative and divisive. With agglomerative methods, each data point starts as its own cluster and then forms small local clusters, which then merge together to create larger clusters. This includes pairwise comparison methods, which have been used successfully for test-case prioritization in the past.[10] With divisive methods, all data points start in one cluster and then iteratively break up the into smaller and smaller sets. Divisive clustering has been used to quickly filter for test cases likely to cause software failures.[11] The subclustering algorithm described in this section is a combination of both a divisive clustering algorithm and an agglomerative algorithm. It first operates in a divisive manner, breaking clusters into smaller subclusters, and then merges those subclusters back together to identify individual score clusters.

Clustering test cases has been a been a popular method for prioritizing testing suites[10,12–14] that find a diverse set of failures. Our approach differs from algorithms described in other works in its use of multiple scoring criteria. While most hierarchical algorithms split or merge clusters based on a single metric space, our algorithm applies multiple distinct metric spaces as defined by the score tree. This means that rather than trying to detect a hierarchical structure in the test data, the algorithm instead imposes the structure defined by the expert knowledge inherent in the relationships between the performance metrics.

Definition of Subclusters

Clustering over continuous metrics rather than binary score criteria can mitigate the effects of noise in the search for and identification of performance boundaries. The challenge is identifying the continuous metrics to cluster over. Including all possible metrics leads to a diluted search, and the points become too far apart to cluster. Selecting only a few metrics tends to improves clustering. However, discovering a diverse set of behaviors requires a diverse set of metrics, as each metric could potentially reveal a different type of performance boundary. Ideally, test metrics are selected that have the greatest impact on the primary performance metrics of interest. In the case of the UUV mission, the performance metrics of interest are safety success and mission success. Subclustering using hierarchical relationships is one way to balance the needs of including many metrics for diversity, while not diluting the results of the search or clustering algorithms.

The key to the new approach is the subscore tree introduced in the section on RAPT. It is the structure that defines the relationships between a binary success criterion such as mission success to subscore criteria such as waypoint success or transmission success. Each element of the subscore tree H has two properties, a set of indices $H.K$ and a set of child nodes $H.children$. The indices $H.K$ indicate the position of that score element in the score vector Y, as a flattened representation of the score tree.

Using the subscore tree, it is possible to develop both subclusters and subboundaries for our system. A cluster $V^C \subset V$ is defined as the subset of samples in V that all have the same class $c \in C$. Each cluster is composed of a number of disjointed subclusters such that $V^C = \{V_1^C, V_2^C, \ldots V_n^C\}$. The cluster V^C is created by applying our algorithms to the score elements at the root of the score tree H, and its subclusters are created by applying our algorithms to the children of H. For example, if we cluster based on the mis-

sion and safety scores to create V^C, we create its subclusters $\{V_1^C, V_2^C, \ldots V_n^C\}$ by applying the clustering algorithm to the next level of scores: waypoint, transmission, etc. Finally, a subboundary is defined as a set of paired samples that lie between two subclusters, $B^{(C_1,C_2)} = \{[v_1, v_2], \ldots, [v_{n_1}, v_{n_2}]\}$, where all members of the boundary are members of V^C and each pair is made up of a member of V_1^C and a member of V_2^C.

The second category of clusters is the subscore cluster V^k, which is created by applying clustering to the kth element of the score vector. For example, once the score tree is flattened, the waypoint distance metric may be the 12th element in the score vector. These, in turn, have a subscore boundary designated as B^k. These subscore clusters are our true objective, and later in this section we discuss how these two types of clusters are related and how each is used for our analysis of the UUV simulation.

Figure 10 shows the different subclusters and subscore clusters for a simple three-element tree. In this system, the primary clusters are defined by the root element A that is computed via an AND operator on the leaf elements B and C. The subclusters provide different ways of subdividing the system using the values of leaf elements. If both B and C are used (Fig. 10b), the original parent cluster [A=0] is split into two clusters: [B=0,C=1] and [B=1,C=0]. Isolating either score element B or C would create subscore clusters based on their values alone, as shown in Figs. 10c and 10d.

As illustrated in Fig. 11, as more subscore elements are used for clustering, more subclusters are created. The synthetic system has four classes that define the primary clusters. Each of these can be broken into smaller subclusters by applying clustering algorithms to the second level of the tree. These subclusters can then, in turn, be broken down further by applying a clustering algorithm to the third level of the tree. Figure 11 illustrates this process.

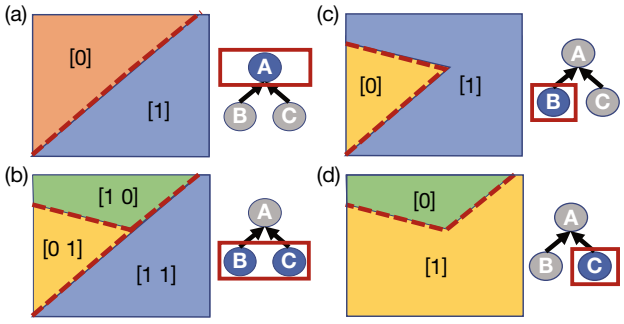


Figure 10. Illustration of the various subclusters and subboundaries for a simple three-element score tree. (a) The clusters and boundaries for the root element A. (b) Subclusters formed by applying our clustering algorithm to both B and C simultaneously. (c) The subscore clusters for element B. (d) The subscore clusters for element C.

Hand in hand with the concept of subclusters is the concept of subboundaries. Subboundaries are the divisions between subclusters inside of a larger cluster. The synthetic system includes six primary boundary types based on the transitions between different classes. The subclusters of the intermediate scores are separated by subboundaries between the primary boundaries, thereby dividing each cluster and subcluster into smaller and smaller sets. An example of this process is shown in Fig. 11.

Hierarchical Subclustering

Our goal is to obtain boundary information for any score element in the score tree without having *a priori* knowledge of which elements will be important. However, it is not possible to compute all score elements simultaneously using our previous clustering technique. This section introduces an approach for creating a subboundary structure more quickly than the method

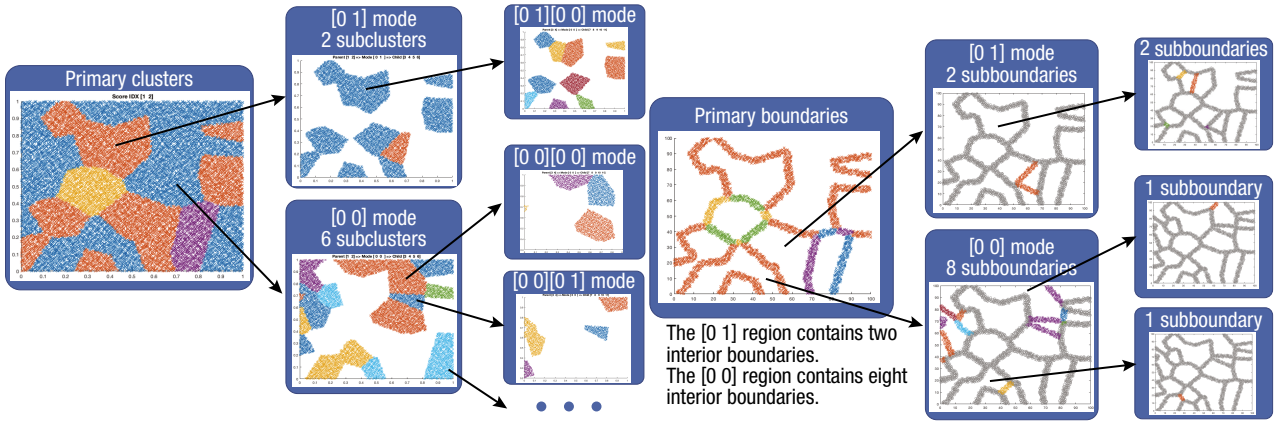


Figure 11. Example illustrating how each cluster of the system is split into smaller subclusters and subboundaries. The [0 1] cluster can be broken up into two subclusters, while the [0 0] cluster can be broken up to six subclusters using the level 2 score elements. These can be broken up further using the level 3 score elements.

described in Ref. 2. It also allows for efficient retrieval of any score boundary.

The new subboundary algorithm uses the hierarchical nature of the subscore tree to break the problem into smaller pieces. This turns an $O(m^2n^2)$ operation and into an $O(mn^2)$ operation, allowing us to more quickly create clusters and their boundaries for every metric in our score tree. This process is illustrated in Fig. 11, where we broke each cluster into smaller and smaller subclusters by iteratively applying our clustering algorithms. The algorithm works as follows:

- The primary clusters of the system are identified by applying mean-shift clustering[15] to the root elements of the score tree.
- Each of these clusters is then subjected to clustering using the metrics at the next level down on the score tree.
- These new clusters are then added as children to the parent cluster.

This process is applied recursively until it has reached the bottom of the score tree or a cluster cannot be subdivided. At each step of the subclustering process, we use pairwise comparison between members of different clusters to identify the boundary pairs. Since the sizes of these clusters become progressively smaller as the process continues, each child in the subboundary tree takes less time to compute than its parent. This process is described more formally in Algorithm 1.

The output of this algorithm is a boundary tree structure. Each level in the boundary tree relates directly to

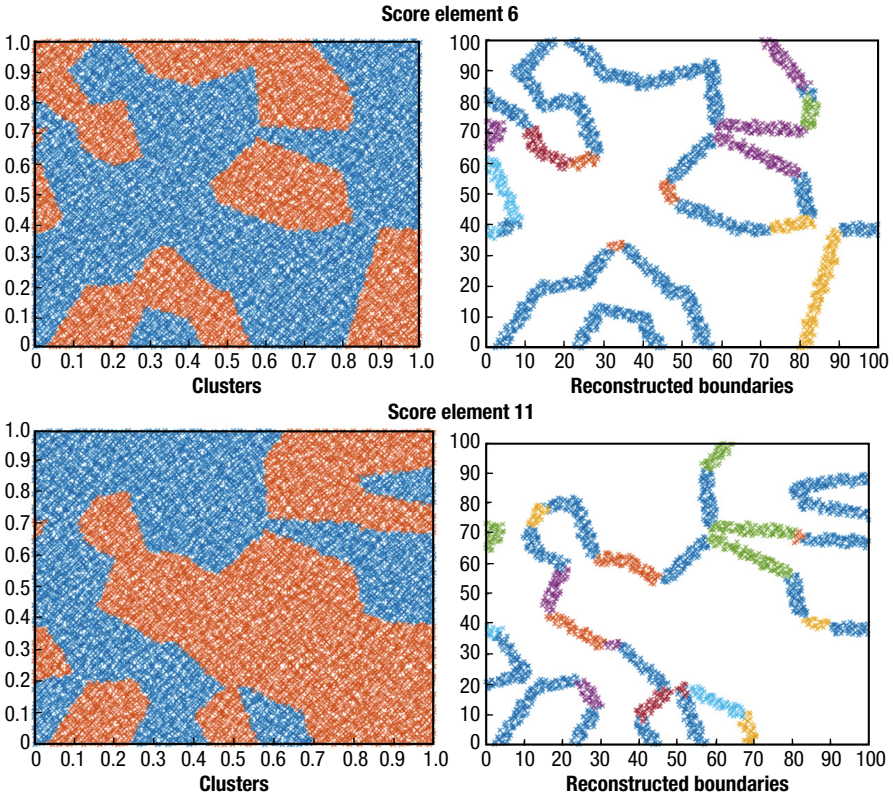


Figure 12. Examples of reconstructed subscore clusters and boundaries for the synthetic system.

ALGORITHM 1

Subboundaries(X,Y,H)

1. Set the subscore indices $K = H.K$
2. Cluster using the selected subscores $[L, C] = MeanShift(Y(K))$
3. Create labeled sample set $V = [X, Y, L]$
4. Set the subclusters for V as $\{V^{C_1}, \ldots, V^{C_n}\} \forall\, C_i \in C$
5. **Foreach** $C_i \in C$
   a. **Foreach** $C_k \in C, k > i$
      i. $I_{\{ik\}} = knnsearch\,(X^{C_i}, X^{C_k}, 1)$
      ii. $I_{\{ki\}} = knnsearch\,(X^{C_k}, X^{C_i}, 1)$
      iii. Create boundary pairs $b_k = [x_j, x_l] \in B^{\{C_i, C_k\}}$ if $I_{\{ik\}}(x_j) = x_l$ and $I_{\{ki\}}(x_l) = x_j$
      iv. $B.append(B^{C_i, C_k})$
   b. $T^{C_i} = Subboundaries(X^{C_i}, Y^{C_i}, H.children)$
   c. $S.append(T^{C_i})$
6. **Return** $T = [V, B, C, S, K]$

a level in the score tree. Each node in the boundary tree contains information about the clusters at that level, the boundaries for those clusters, the score indices used for clustering, and the cluster's child subboundary nodes.

Subscore Clusters and Boundaries

The objective is to identify the boundaries for every score element in the tree. However, it is too computationally intensive to apply our original boundary identification algorithm to every score element simultaneously. Instead of computing all boundaries individually, the subboundary tree T can be used instead to reconstruct the boundaries associated with any scoring element. This means the clusters and boundaries can be retrieved for any given score element with minimal additional computation. This process involves searching the subboundary tree structure for all subclusters that have the same value for the specified score element, then merging all the identified subclusters and subboundaries into a single set. The method is detailed in Algorithm 2.

An example of reconstructed subscore boundaries for score elements 6 and 11 is shown in Fig. 12. The clusters

ALGORITHM 2

Reconstruct (T, y_S, K_s)

1. $[V, B, C, S, K] = T$
2. If $K = K_s$
   a. Find $y_k = nearestNeighbor(y_s, Y)$
   b. Set c_k to the label of y_k
   c. Set $V_s = V^{C_k}$
   d. Set $B_s \subset B$ s.t $\forall\, b_i = [v_j, v_l] \in B_s$, either $v_j \in V_s$ or $v_l \in V_s$

Else

   e. **Foreach** $T_i \in S$
      i. $[V_i, B_i] = Reconstruct(T_i, y_s, K)$
      ii. $V_s.append(V_i)$
      iii. $B_s.append(B_i)$
3. **Return** $[V_s, B_s]$

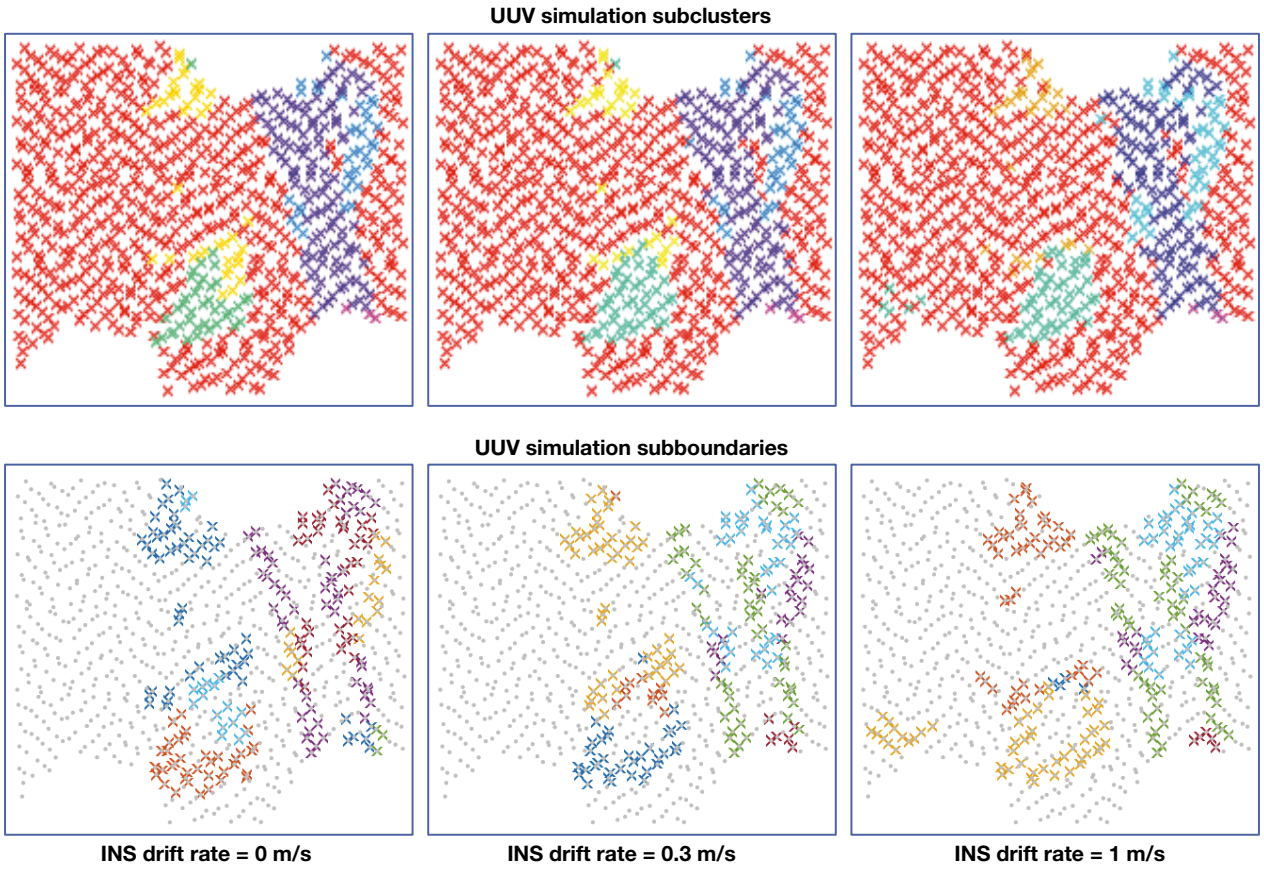


Figure 13. Scatterplots showing the effect of increased stochasticity in the form of INS drift on the boundaries identified by the subclustering algorithms. The plots show the reconstructed subclusters and subboundaries for the waypoint distance and recovery point distance metrics.

are shown in red and blue for when the score is 0 or 1, respectively. The different colors in the reconstructed boundary indicate the different subboundaries that were merged in the reconstruction process.

APPLICATION TO THE UUV MISSION

The ability to find subclusters and then use them to reconstruct any subscore boundary makes it possible to efficiently identify performance boundaries for any of the continuous metrics in the subscore tree. Different metrics will reveal different behavioral modes of the system under test, and some will be less impacted by stochastic effects than others. This provides a way to differentiate resilience of the autonomy's subsystems from resilience in its decision-making process. For example, as shown in Fig. 4, there is a great deal of variance in the binary success criteria of the vehicle without much change in the vehicle's trajectory. The vehicle's navigation is unreliable, but the decision-making process is stable.

Applying these new subclustering and subboundary identification algorithms to the UUV simulation immediately reveals their robustness to increasing stochasticity in the simulation. Figure 13 illustrates the effect of increasing INS drift on the subboundaries defined by the waypoint distance and recovery point distance metrics. These metrics were selected because they are primary contributors to the safety success and mission success criteria. Comparing Fig. 13 to Fig. 9 shows the dramatic difference in stability between the original binary success clusters and the subclusters created by the continuous metrics. Even when the INS drift rate is increased to the point where the vehicle is unable to complete the mission because it consistently misses the target, the identified subboundaries remain the same.

One of the desirable qualities of the system under test is a sharp performance boundary. It is both an indicator of resilience and can help a test engineer isolate the cause in the behavior of the system. A method

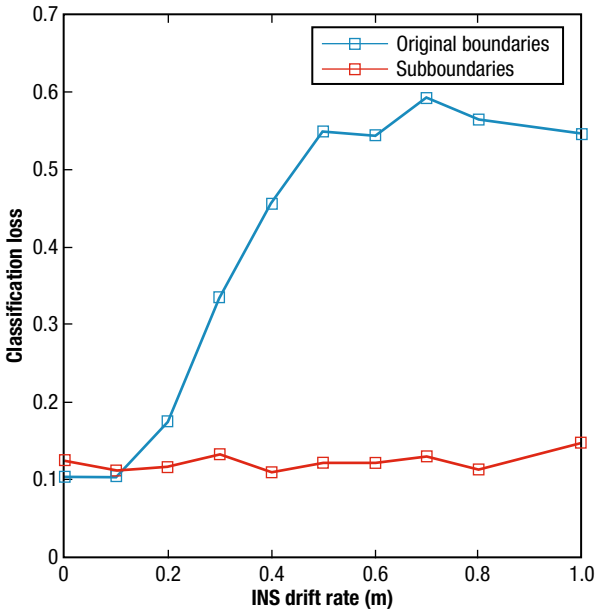


Figure 14. Plot showing the effect of increasing simulator stochasticity in the form of INS drift on ability of the boundary and subboundaries to separate the performance modes.

for quantifying the sharpness of the performance boundary is measuring how well it separates the two classes. These classes can be calculated by training a classifier on both the original performance boundaries and the subboundaries comparing the cross-validation loss of each. A more accurate classifier means the data are easier to separate, and thus identified boundaries are "sharper." A classification tree model[16] was trained for both data sets for varying levels of INS drift rate, and the results of this comparison are shown in Fig. 14. Classification loss for the subboundaries remains constant despite the level of the INS drift rate, whereas the loss for the original boundaries grows significantly until the classifier is no better at separating the two classes than random chance.

The subboundaries derived from the continuous metrics also tend to be more representative of changes

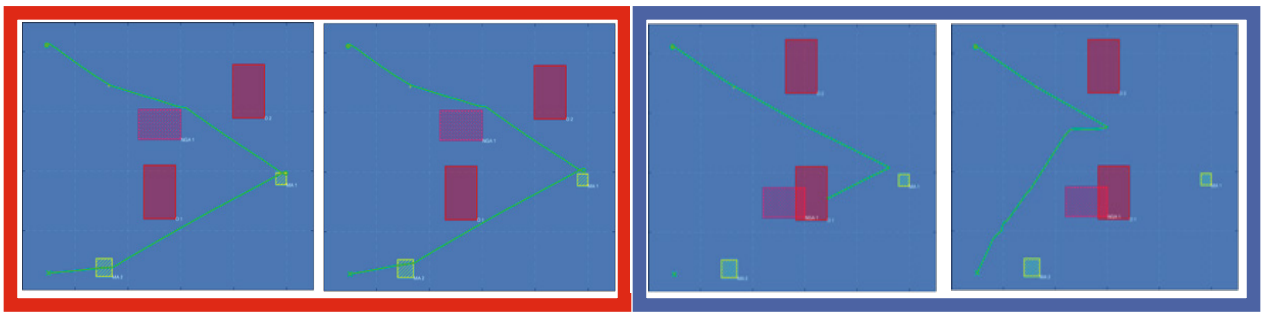


Figure 15. Comparison of a boundary pair found for the original boundaries using the binary scores versus the subboundary pairs using the continuous metrics.

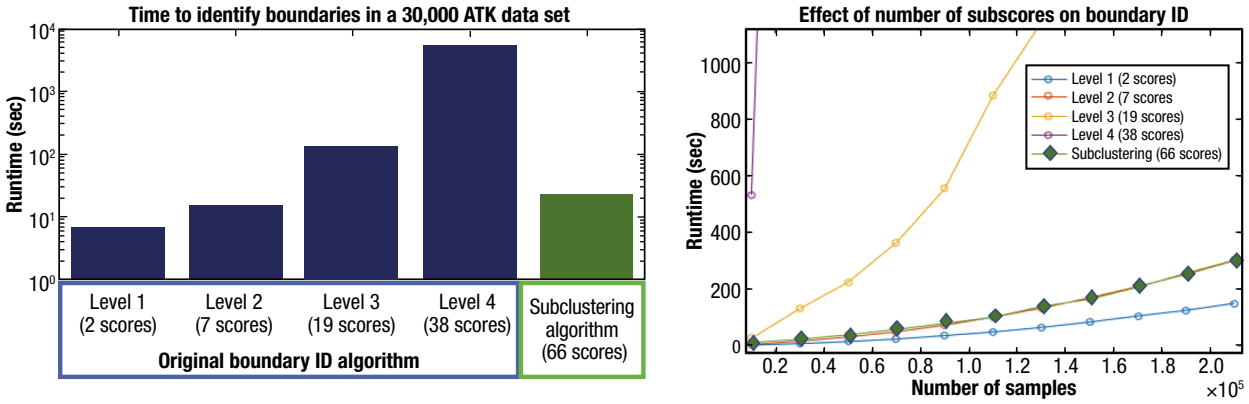


Figure 16. Timing comparison of the subboundary algorithm and previous algorithm. Left, Bar chart showing the time to cluster a 30,000 data set for increasing numbers of subscores, with the new technique on the far right. Right, Line plot showing the time to cluster a data set for increasing numbers of samples. The new subclustering process is collinear with the results for clustering seven scores simultaneously.

in the decision-making than the those identified by the binary score metrics. An example of this is shown in Fig. 15, which compares boundary pairs found for each method. The boundary pair found for the binary score exhibits no change in behavior, just a violation of the mission constraints by touching a no-go area. The boundary pair for the continuous metrics exhibits a different decision on when to attempt to return to the recovery point.

The final key feature of the subboundary identification process is that it discovers all the subboundaries of the system without significantly increasing the computational time. The subboundary algorithm was evaluated on 1 million data points generated by the Sobol design technique. Some of the 1 million data points were invalid for reasons such as spawning an obstacle over a waypoint. This left a total of 863,000 valid UUV simulation runs. The resulting data set was then run against both the new and old boundary identification algorithms against progressively deeper levels of the score tree. The results of this comparison are shown in Fig. 16. The new algorithm takes approximately 20 s to process a subscore tree with four levels and 66 leaf metrics that is comparable to applying the original boundary identification algorithm to nine metrics simultaneously. Applying the original boundary identification technique to all 66 leaf metrics simultaneously takes approximately 2 h. Thus, this new approach for analyzing all the score elements simultaneously takes 200-fold less time than our prior approach.

CONCLUSIONS AND FUTURE WORK

This article introduces a hierarchical approach for evaluating an AUT's performance and how uncertainty in that performance affects the RAPT framework. By focusing on continuous metrics rather than binary success criteria, the search and boundary identification methods become more robust against the effects of noise. Hierarchical scoring combined with subclustering algorithms generates salient boundaries for a large number of scoring metrics in a computationally efficient manner. These techniques enable application of the RAPT framework to stochastic simulations that are key for testing the resilience of unmanned underwater systems.

The use of a score tree is an intuitive extension of how autonomy designers are already evaluating autonomous systems, where requirements are typically written as binary success criteria based on a continuous metric reaching a threshold. While RAPT provides a flexible method for defining these criteria, the problem of how to design effective resilient scoring metrics remains. It is still the responsibility of subject-matter experts to identify the AUT's requirements and how they should be measured. The purpose of RAPT is to enable test engineers to quickly study the effects of a large number of metrics and inform them of the performance modes that result from that evaluation. The tool allows engineers to iterate on their metric designs to find the best way to measure the resilience of the system.

Subclustering as a method for finding the different emergent behaviors of an autonomous system remains an open area of investigation. Black-box testing is limited by what can be externally observed, and as such it can be difficult to differentiate when a new behavior has been triggered or when two different performance modes are actually the result of the different behaviors. Unsupervised methods for identifying these behavioral modes are a powerful tool that can discover phenomenon the test engineer did not expect.

Research into methods of measuring AUT resilience and automatically identifying these performance modes from externally observable data continues.

REFERENCES

[1]Chen, J., and Patton, R. J., *Robust Model-Based Fault Diagnosis for Dynamic Systems*, Springer Science & Business Media, Berlin (2012).

[2]Koopman, P., and Wagner, M., "Challenges in Autonomous Vehicle Testing and Validation," *SAE Int. J. Transport. Safety* **4**(1), 15–24 (2016).

[3]Tuncali, C. E., Fainekos, G., Ito, H., and Kapinski, J., "Sim-ATAV: Simulation-Based Adversarial Testing Framework for Autonomous Vehicles," in *Proc. 21st International Conf. on Hybrid Systems: Computation and Control* (Part of CPS Week), New York, pp. 283–284 (2018).

[4]Mullins, G. E., Stankiewicz, P. G., Hawthorne, R. C., and Gupta, S. K., "Adaptive Generation of Challenging Scenarios for Testing and Evaluation of Autonomous Vehicles," *J. Syst. Softw.* **137**, 197–215 (2018).

[5]Mullins, G. E., Stankiewicz, P. G., and Gupta, S. K., "Automated Generation of Diverse and Challenging Scenarios for Test and Evaluation of Autonomous Vehicles," in *Proc. 2017 IEEE International Conf. on Robotics and Automation (ICRA)*, Singapore, pp. 1443–1450 (2017).

[6]Zou, X., Alexander, R., and McDermid, J., "Safety Validation of Sense and Avoid Algorithms Using Simulation and Evolutionary Search," in A. Bondavalli and F. Di Giandomenico (eds.), *Computer Safety, Reliability, and Security*, Springer, Cham, pp. 33–48 (2014).

[7]Steinberg, M., Stack, J., and Paluszkiewicz, T., "Long Duration Autonomy for Maritime Systems: Challenges and Opportunities," *Auton. Robots* **40**(7), 1119–1122 (2016).

[8]Tan, H.-P., Diamant, R., Seah, W. K., and Waldmeyer, M., "A Survey of Techniques and Challenges in Underwater Localization," *Ocean Eng.* **38**(14–15), 1663–1676 (2011).

[9]Murtagh, F., and Contreras, P., "Algorithms for Hierarchical Clustering: An Overview," *WIRES Data Min. Knowl.* **2**(1), 86–97 (2012).

[10]Yoo, S., Harman, M., Tonella, P., and Susi, A., "Clustering Test Cases to Achieve Effective And Scalable Prioritisation Incorporating Expert Knowledge," in *Proc. 18th International Symp. on Software Testing and Analysis*, Chicago, pp. 201–212 (2009).

[11]Dickinson, W., Leon, D., and Podgurski, A., "Finding Failures by Cluster Analysis of Execution Profiles," in *Proc. 23rd international Conf. on Software Engineering*, Toronto, pp. 339–348 (2001).

[12]Sapna, P. G., and Mohanty, H., "Clustering Test Cases to Achieve Effective Test Selection," in *Proc. 1st Amrita ACM-W Celebration on Women in Computing in India*, Coimbatore, India, p. 15 (2010).

[13]Zalmanovici, M., Raz, O., and Tzoref-Brill, R., "Cluster-Based Test Suite Functional Analysis," in *Proc. 2016 24th ACM SIGSOFT International Symp. on Foundations of Software Engineering*, New York, pp. 962–967 (2016).

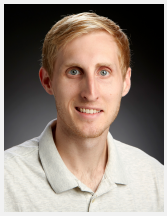
[14]Nurmuradov, D., Bryce, R., Piparia, S., and Bryant, B., "Clustering and Combinatorial Methods for Test Suite Prioritization of GUI and Web Applications," in *Proc. 14th International Conf. on Information Technology*, Las Vegas, pp. 459–466 (2018).

[15]Comaniciu, D., and Meer, P., "Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(5), 603–619 (2002).

[16]Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A., *Classification and Regression Trees*, 1st Ed., Routledge, New York (1984).

Galen E. Mullins, Research and Exploratory Development Department, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Galen E. Mullins is a roboticist in APL's Research and Development Department. He received bachelor's degrees in mechanical engineering and mathematics from Carnegie Mellon University, a master's degree in applied physics from Johns Hopkins University, and a doctorate in mechanical engineering from University of Maryland. He is the algorithm lead for the Range Adversarial Planning Tool (RAPT) program and has research interests in robotics, autonomy, machine learning, and numerical optimization. His e-mail address is galen.mullins@jhuapl.edu.



Paul G. Stankiewicz, Force Projection Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Paul G. Stankiewicz is an engineer in the Ocean Systems and Engineering Group of APL's Force Projection Sector. He received an M.S. and a B.S. in mechanical engineering from Penn State in 2015 and 2013, respectively. His research background is in dynamic systems and control, with a focus on autonomous systems. Paul supported the Range Adversarial Planning Tool (RAPT) algorithm development for intelligent search and boundary identification. His e-mail address is paul.stankiewicz@jhuapl.edu.

Melissa A. Huntley, Force Projection Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Melissa Huntley is the assistant section supervisor of the Autonomous Systems Section in the Ocean Systems and Engineering Group in APL's Force Projection Sector. She led the test range software development effort for the Range Adversarial Planning Tool (RAPT). She received a B.S. in computer engineering from University of Maryland, Baltimore County, in 2013 and an M.S. in systems engineering from Johns Hopkins University in 2016. Her e-mail address is melissa.huntley@jhuapl.edu.