Delivering Test and Evaluation Tools for Autonomous Unmanned Vehicles to the Fleet

Galen E. Mullins, Paul G. Stankiewicz, R. Chad Hawthorne, Jordan D. Appler, Michael H. Biggins, Kevin Chiou, Melissa A. Huntley, Johan D. Stewart, and Adam S. Watkins

ABSTRACT

The Johns Hopkins University Applied Physics Laboratory (APL) is working to develop the next generation of test and evaluation (T&E) tools for maritime, air, and ground autonomous systems. Advancement in autonomy on unmanned vehicles is outpacing test ranges' capability for effective T&E of these systems. DoD test ranges face the challenge of being able to execute a very limited number of live tests to validate increasingly complex systems. APL is performing research and development to help solve the cost and reliability issues associated with on-range T&E of autonomous systems. Using advanced optimization techniques to intelligently explore the highly complex state space in which autonomous systems operate, the Range Adversarial Planning Tool (RAPT) team is developing tools for test ranges to identify the most relevant tests for the full scope of maritime, airborne, and ground-based autonomous systems. The principal challenge with testing and evaluating autonomy is addressing the complex, NP-Hard (non-deterministic polynomial-time hard) interactions between autonomy and the environment. Decomposing the problem only exacerbates the situation by producing an intractable set of options in various mission conditions and internal states of an autonomous system. Therefore, autonomy can be evaluated only with precise understanding of the interactions between the autonomous vehicle and the environment, enabling delineation of which situations are effective from a T&E perspective and which are not.

INTRODUCTION

Designing test scenarios for validation and verification of autonomous vehicles is currently an expensive and involved process. Testing requires the input of subject-matter experts who are thoroughly versed in the behaviors of both the platform and the autonomous decision engine under test. Performing live tests is also very time consuming, which severely limits the number of tests that can be performed. In this article, we introduce a new method for intelligently generating test scenarios that inform testers on the expected performance of the autonomous system.

Our approach to designing informative test cases differs from recent work in validating autonomous systems. We are not focusing on fault detection[1] or model checking[2] of the underlying decision engine. Instead of modeling the underlying behavior of a black-box

autonomous system,[3] we try to model the relationship between scenario configuration and the resulting performance metrics for the autonomy. This approach has the advantage of greatly reducing the input and output state-spaces from those that are used for traditional fault detection. The result is a model that can detect and classify anomalous behavior based on the performance metrics it exhibits.

A candidate test scenario is deemed interesting and informative if it is close in the configuration space to a performance boundary. In this article, we define a performance boundary as a location in the configuration space where a discontinuity or large change in performance occurs. For example, a performance boundary may occur when a slight change in obstacle position causes an autonomous agent to transition from succeeding in its mission to failing. By identifying test cases near performance boundaries, we can assist test designers in selecting scenarios that are most likely to be informative so they can design test plans tailored to the specific autonomy in question. The method described in this article is divided into two primary approaches: the search approach and the boundary identification (ID) approach. During the search approach, we use active learning to select new test cases that are run by the autonomy simulation. Gaussian process regression (GPR)[4] is used to form a model of the autonomy performance and preferentially select regions that might indicate performance boundaries. The high dimensionality of the configuration space for an autonomy under test makes it intractable to simply perform an exhaustive spread of simulations. Thus, intelligent searching of the configuration space is necessary to obtain adequate coverage of the boundary regions while minimizing the number of simulations. In the boundary ID approach, samples generated during the search approach are used to identify the performance modes in the resulting data by using unsupervised clustering algorithms. Once test cases have been classified by their performance mode, the boundaries between performance modes are identified, and the tested scenarios adjacent to boundaries can be used to aid in live test design.

TEST AND EVALUATION NEED

The validation and verification of autonomous systems has been an active area of research in the past few years, particularly in the use of active learning for test case generation. Here we discuss recent work in the fields of verification of autonomous systems and automatic test case generation and how it relates to the work discussed in this article.

Validation of autonomous systems for critical missions has been increasing in importance as robots have become more prevalent in ordnance disposal, search and rescue, and other systems that have very low tolerance for failure.[5] There has been a lot of work to develop methods that provide performance guarantees based on model-checking and formal methods.[2,6] These methods require that a model—which must fully describe the autonomy performance—be generated and exhaustively tested for exceptions that break the specifications. Models that have been used in the past include finite state machines[1,7] and process algebras.[6] A drawback of this approach is that a model must first fully describe the autonomy, and test engineers must have full access to the model. Given the increasing complexity of autonomous systems and the black-box nature of proprietary software, these limitations prevent these methods from being applied to many systems.

Learning-based methods for developing models of black-box systems for robustness testing have been the topic of many recent papers.[8] These techniques often combine a model-checking strategy with an incremental model inference or learning algorithm. Typically, a finite-state machine built using Angluin's algorithm L* has been used.[9,10] Our method differs from these techniques by not attempting to model the actual behavior of the autonomy with respect to its inputs. Rather, we try to model its overall performance with respect to the stimuli it receives from its mission and environment. One method similar to our approach of boundary identification is the production of a set models used for fault detection.[11] Our focus on modeling autonomy performance based on mission and environmental inputs allows us to perform black-box testing of the autonomous system.

RAPT ARCHITECTURE AND SIMULATION

The Range Adversarial Planning Tool (RAPT) generates test-planning products by using data from simulations of the autonomy software. Our approach relies on GPR techniques to identify the boundary conditions of the autonomy in simulation; autonomy performance metrics intelligently guide our search algorithm to identify performance boundary scenarios. Our GPR search generates scenarios that vary environmental, mission, and vehicle states in an open-architecture simulation environment.

As shown in Fig. 1, the RAPT system consists of three primary components: (1) an offline state description, (2) optimization algorithms, and (3) a simulation environment. The offline state description allows test designers to define the state space for air, ground, and maritime systems against which the autonomy will be tested. The entirety of the state space cannot be executed in simulation due to the sheer size, so RAPT explores this state space using a stochastic optimization algorithm to identify test scenarios for execution in simulation. Results from these simulations inform the optimization algorithms as to which subsequent simulations will be informative and help determine the performance bounds of the autonomy.

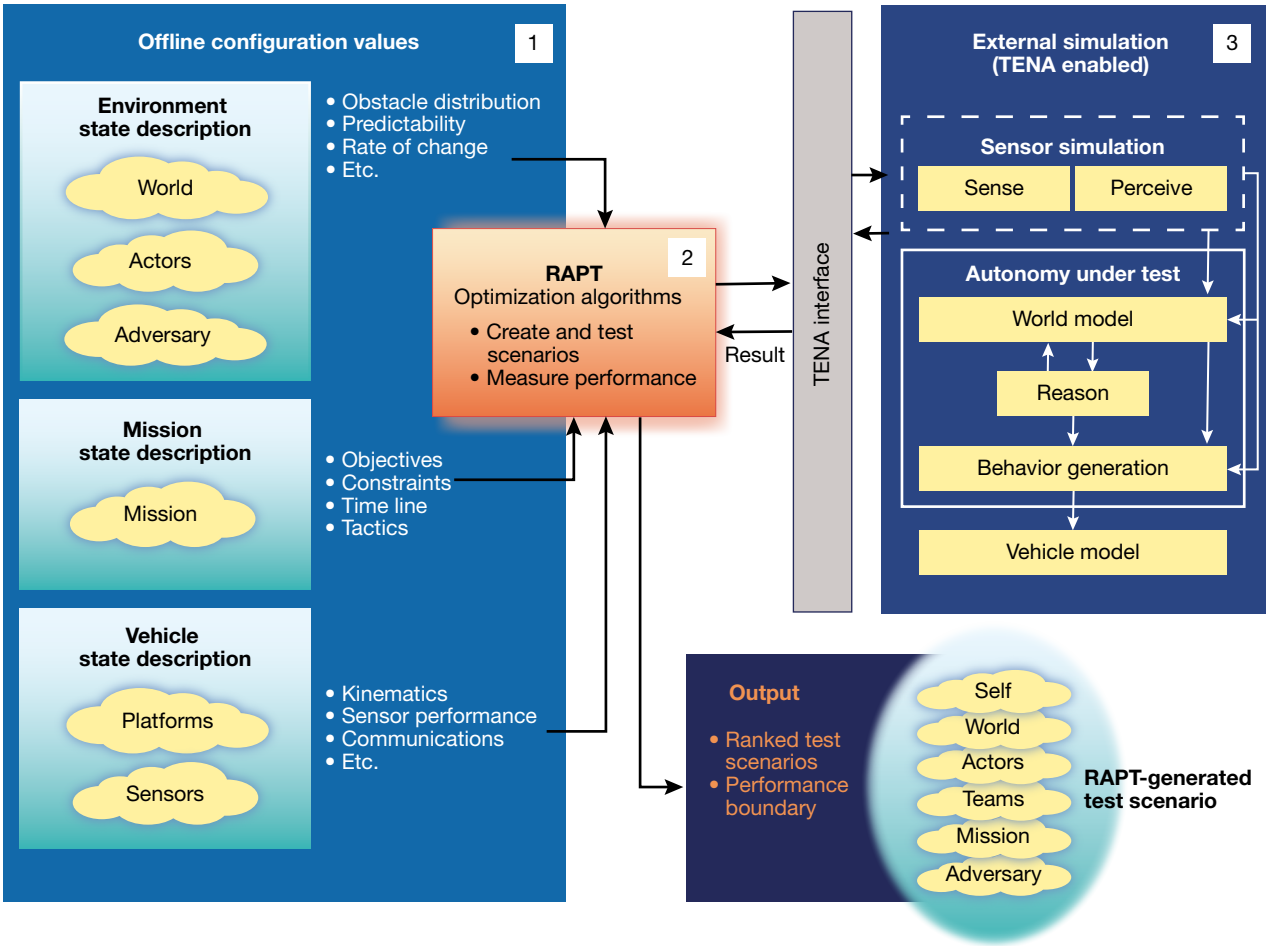


Figure 1. High-level architecture diagram of the RAPT system. A human-derived description of (1) the state space provides input to (2) an optimization algorithm, which creates tests performed in (3) simulation and assesses efficacy in defining the performance boundary. TENA, Test and Training Enabling Architecture.

PROBLEM DEFINITION

We define the performance boundary identification problem as one of discovering test cases in the scenario configuration space that are close to discontinuities or large gradients in the performance space (i.e., where small changes in the scenario result in large changes in autonomy performance). Definitions include the following:

(i.) The scenario configuration state space $\mathcal{X}^n$ of n parameters, where each input state is defined as the vector $X = [x_1, x_2, ..x_n]$. A sample set of N state vectors is defined as $X^N = [X_1, .., X_N]$. The normalized state vector where each $\bar{x}_i \in [0, 1]$ is defined as $\bar{X}$.

(ii.) The performance score space $\mathcal{Y}^m$ of m parameters where each output score is defined as the vector $Y = [y_1, y_2, ..y_m]$. A sample set of N score vectors is defined as $Y^N = [Y_1, .., Y_N]$. The normalized score vector where each $\bar{y}_i \in [0, 1]$ is defined as $\bar{Y}$.

(iii.) The System Under Test (SUT) function $\mathcal{F}(X^N) = Y^N$ which accepts a set of N input states $X^N = [X_1, ..., X_N]$ and returns sample set of N score vectors $Y^N == [Y_1, ..., Y_N]$.

(iv.) The search algorithm $\mathcal{S}(\mathcal{F}, \mathcal{X}^n, \mathcal{Y}^m, N) = \mathcal{L}$ which accepts the SUT function $\mathcal{F}$, the scenario configuration space $\mathcal{X}^n$, the performance score space $\mathcal{Y}^m$, and the desired number of samples N as input and returns a set of labeled samples $\mathcal{L} = [X^N, Y^N]$ consisting of the queried states X^N and their respective scores Y^N.

(v.) The identification algorithm $\mathcal{P}(\mathcal{L}) = \mathcal{B}$ which accepts a set of labeled samples $\mathcal{L}$ and returns the set of boundaries $\mathcal{B} = [B_{1,1}, B_{1,2}, ..., B_{L-2,L}, B_{L-1,L}]$ of all identified boundaries where each boundary $B_{a,b}$ is the set of points that borders the performance modes a and b.

SYSTEM UNDER TEST

Several candidate systems were developed to evaluate our search and boundary ID algorithms. The first category of candidate systems consisted of mathematical test functions that accept either a two- or three-dimensional input state and output one-dimensional scores. The second category of candidate systems comprised a simple unmanned undersea vehicle (UUV) scenario with a

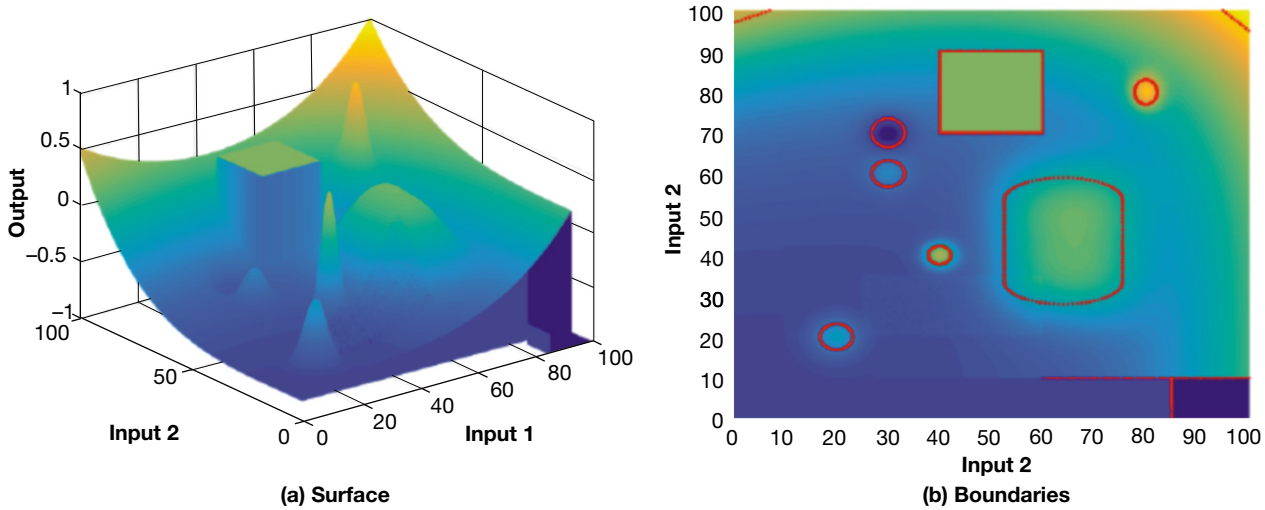


Figure 2. Custom2d: a two-dimensional input, one-dimensional output test function showing the (a) surface and (b) self-identified boundaries.

six-dimensional input space and a two-dimensional output space.

Three custom test functions were developed in order to evaluate the algorithms for performance boundary identification. These test functions have the advantage that they are easy to visualize and have ground-truth boundaries that were known *a priori*. In each function, we defined the performance boundaries as the local maxima of slope based on the features of the function. Two of the test functions accepted a two-dimensional input state and returned a one-dimensional score, while the third accepted a three-dimensional input state. These test functions are named the Custom2d (Fig. 2), Plates2d (Fig. 3), and Plates3d test functions. The Custom2d function outputs a continuous score and contains 11 artificially introduced features including peaks, valleys, plateaus, and cliffs in score. The surface can be seen in Fig. 2a, and the truth boundaries are drawn in Fig. 2b. This function was chosen and developed over standard test functions for optimization in an attempt to represent the many different types of performance boundaries an autonomy may experience. Thus, the features included in the function are a mix of discontinuities, noise, and varying degrees of gradients. Standard test functions were also tested but are not discussed in this article. The Custom2d function tests the algorithms' ability to discover performance boundaries of different magnitudes and different shapes. The Plates2d function is a representation of a nearest neighbor function and outputs a discrete score based on the label of the seed point closest to the sampled state. The surface can be seen in Fig. 3a, and the truth boundaries are drawn in Fig. 3b. The purpose

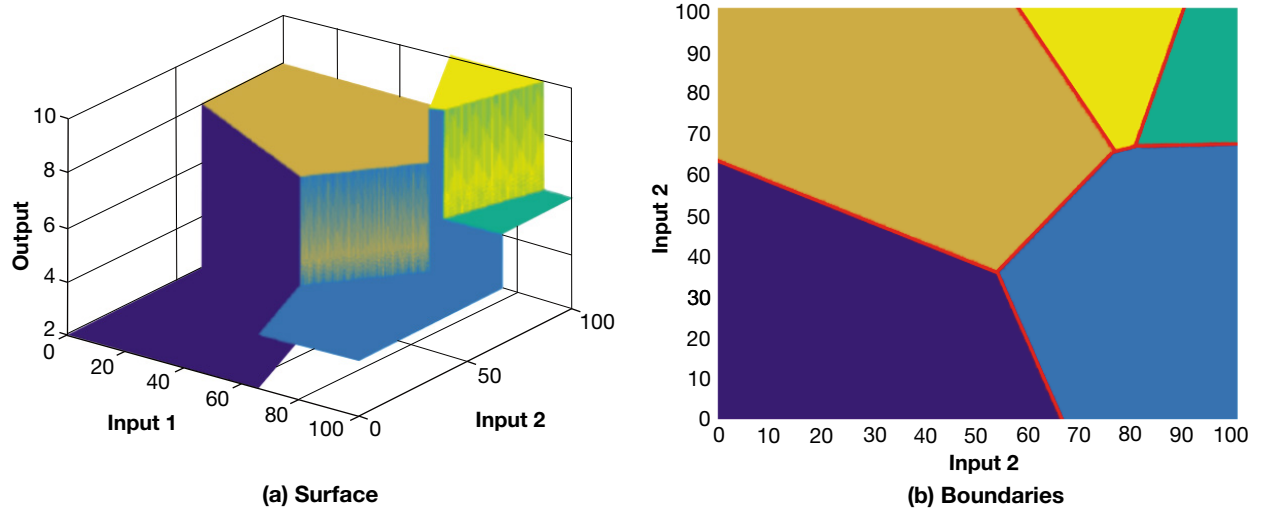


Figure 3. Plates2d: a two-dimensional input, one-dimensional output test function showing the (a) surface and (b) self-identified boundaries.

of this function was to provide a system in which the performance boundaries were clearly defined and to test the algorithms against a discrete output. The Plates3d function is identical to the Plates2d function; however, the seed points and nearest neighbor search are performed in a three-dimensional input space. The purpose of this function was to provide a higher-dimensional case that still had clearly defined boundaries.

In addition to test functions, the search and selection algorithms were also evaluated on an autonomous vehicle simulation. For our autonomous vehicle, we chose a simulated UUV operating under a multiobjective navigation scenario. As shown in Fig. 4, the objective of the mission was to travel from a starting point to a goal waypoint and then return to a separate recovery point (all of which are fixed) before its battery runs out of charge. The goal waypoint was placed on the opposite side of a 2-km operational area from the start and recovery points. Inside the operational area, we have placed three square obstacles that are 400 m on a side. These obstacles can vary in the east and north directions. Thus, the centers for the three obstacles form our six input state dimensions, $X = [E_1, N_1, E_2, N_2, E_3, N_3]$.

The autonomy was designed to evaluate its current power limitations and determine whether it should continue to the goal point or return to the recovery point. The autonomy was scored based on whether it reaches the goal waypoint (deemed a mission success) and whether it returns to the recover point successfully (deemed a safety success). These two mission criteria give the output score dimensions, $Y = [M_S, S_S]$, both of which are binary values based on success or failure to meet the requirements.

To experiment further, we also created an alternative scoring method based on continuous metrics including the closest distance of approach to the goal point and the recovery point, as well as the remaining fuel at closest approach, $Y_{alt} = [D_{recovery}, F_{recovery}, D_{goal}, F_{goal}]$. This alternative score was used to determine whether our algorithms could detect any additional performance modes that were not captured by the broader generalizations of mission success and safety success. The vehicle was also given a forward-looking sonar with a range of 1000 m and a 60° field of view for detecting obstacles.

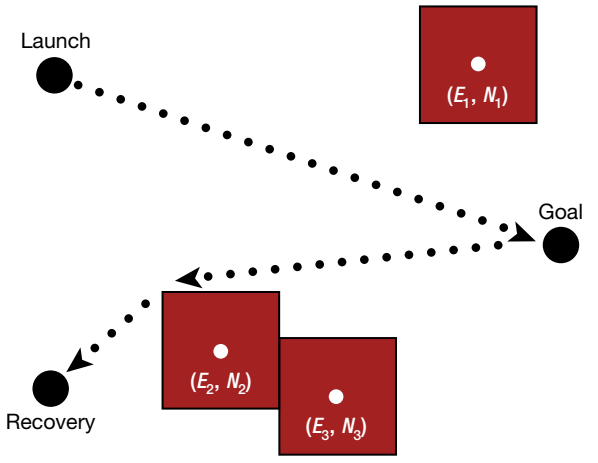


Figure 4. An autonomous UUV scenario. The simulated UUV's navigation mission is to travel to a fixed goal waypoint and return to a separate fixed recovery point—avoiding three obstacles (red boxes)—before its battery loses charge.

The autonomy under test is an implementation of the Tangent Bug algorithm, which estimates the total distance remaining in its mission. If it estimates that the total distance exceeds its remaining battery charge, it immediately begins navigating back to its recovery point. Note that the autonomy intelligence and decision-making was not meant to be optimal. Rather, the naive autonomy described above was implemented so as to create performance boundaries that could be identified.

OPTIMIZATION APPROACH

In this section, we present the first of a two-part approach for generating test scenarios. A search algorithm first preferentially samples from regions near performance boundaries to create a sample set of states and scores. The next section describes a clustering algorithm used to identify performance modes and boundaries from the results and classify the sampled states.

While software-in-the-loop testing of autonomy using a simulated environment allows for a large number of test cases to be run, there are still limitations in the number of samples that can be collected. The simulated mission duration could be several hours, and if the autonomy under test cannot be run faster than real time, the number of samples that can be run will be similarly restricted. Because of these constraints, we require a search strategy that preferentially samples from the regions of interest. To achieve this, we follow an active learning strategy for generating the sample set.

An iterative process is used consisting of model generation followed by the submission of additional queries to the SUT. Our objective is to find areas with high gradients that might indicate the presence of a performance boundary (i.e., where small changes to the state space lead to large changes in performance). However, we also need to continue exploring the space rather than oversampling regions of interest that have already been covered. We achieve this by using GPR to provide both a model of the mean value, the gradient of the mean, and an estimated variance for each prediction point. The covariance of the Gaussian process adjusts a point's distance from a previously sampled point. Thus, a metric based entirely on the covariance would eventually lead to a uniformly sampled state space. To balance exploration of uncertain areas of the space versus refined searching of previously found boundaries, we developed the following information metric to guide the search:

$I(X) = \frac{dY}{dX}(X)^P \sigma(X)^D$, where D and P are turning parameters for the algorithm. Figure 5 illustrates the information metric for a search of the Custom2d function. The final method, shown in Fig. 5, for generating the sample set is given in Algorithm 1, Gaussian Process Search.

Algorithm 1 GAUSSIANPROCESSSEARCH$(SUT, \mathcal{X}^n, N)$

Input: A function representing the system under test $\mathcal{F}$,a scenario state space $\mathcal{X}^n$, and a desired number of samples N

Output: A set of labeled samples $\mathcal{L}$ Select a query batch size of L and an initial batch of randomly selected query states X_0^L. In addition choose a number of predictions p to perform per iteration.

for all $i \in [0, N/L]$ **do**
  $\mathcal{F}(X_i^L) = Y_i^L$
  $\mathcal{L} \frown [X_i^L, Y_i^L]$
  Generate GPR based on latest labeled sample set $\mathcal{L}$
  Randomly select a new set of predictions X^p
  $X_{i+1}^L = argmax_{X \in X^p}[\frac{dY}{dX}(X)^P \sigma(X)^D]$
end for
return $\mathcal{L}$

BOUNDARY IDENTIFICATION APPROACH

The process of boundary detection is a classification problem. While we can potentially quantify the number of performance modes an SUT might exhibit, we do not know where they will occur. We consider a performance boundary as a contiguous region in the state space that exists between two regions exhibiting differing performance modes. Therefore, an SUT that has only two performance modes could still have an unlimited number of boundaries if it exhibits a periodic behavior. In addition, there is no way of knowing the shapes of the performance boundaries for a given SUT. Therefore, a technique is needed that is capable of identifying an unknown number of boundaries that can have an unknown shape in potentially sparse data.

To solve this, unsupervised clustering techniques were used to group samples that are likely on the same performance boundaries. We chose the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm[12] because it does not require a user to define the number of boundaries *a priori* and because of its ability to cluster contiguous regions. In cases in which the SUT outputs discrete values (i.e., the binary criteria for mission success and safety success described earlier), it is trivial to identify distinct performance modes from the resulting scores. To apply our techniques to systems that provide continuous outputs, we have implemented two methods for identifying the performance modes. The first method filters out all cases whose gradients are below the 90th percentile in magnitude of all the sampled states. This results in isolated sets of test cases that can then be clustered in state space using the DBSCAN algorithm and directly identified as performance boundaries. One drawback of this approach is that it does not identify the performance mode associated with each case in the boundary, only that each case exists in a distinct region of interest. The second method is to use mean-shift[13] clustering in the score space to identify the performance modes and classify the samples. Once the samples have been classified with respect to their performance mode, they are then subjected to DBSCAN clustering in state space to identify distinct regions of interest. To obtain the boundaries from these clusters, we perform a pairwise comparison between every cluster with a differing performance mode. We use a k-nearest neighbor detection algorithm to determine the closest neighbor in the adjacent cluster for each sample. Any samples that are within ϵ_B distance of their nearest neighbor in the opposite cluster are added to the final boundary set. The pair distances in the boundary set are then used to determine how close the samples are to the performance boundary. This approach is defined further in Algorithm 2, Boundary Identification.

Algorithm 2 BOUNDARYIDENTIFICATION$(\mathcal{L})$

Input: A set N of labeled samples $\mathcal{L}$ containing theinput states X^N and output scores Y^N

Output: A set of identified boundaries $\mathcal{B}$

Let $_P$ be the threshold distance for the flat kernal MeanShift function, $_C$ and n_{min} be the radius and minimum member parameters for the DBSCAN function. Let ϵ_B be the maximum distance between two points to be considered part of a boundary.
$\mathcal{Y} = MeanShift(Y^N, \lambda_P)$, identify the performance modes
for all $Y \in \mathcal{Y}$ **do**
  Create the set of all states belonging to that performance mode
  $X_Y = X_i | Y_i \subset Y$
  Append the new cluster of states $C_Y = [X_Y, Y]$ to the list of existing clusters
  $C \frown [C_Y]$
end for
for all $C_Y \in C$ **do**
  Create a set of subclusters for the regions of interest using the DBSCAN algorithm
  $\hat{\mathbf{C}}_\mathbf{Y} = DBSCAN(X_Y, \epsilon_C, n_{min})$
  Append the subclusters to the complete set of clusters
  $\hat{\mathbf{C}} \frown [\hat{\mathbf{C}}_\mathbf{Y}]$
end for
for all $\hat{\mathbf{C}}_{\mathbf{Y}i}$ and $\hat{\mathbf{C}}_{\mathbf{Y}j} \in \hat{\mathbf{C}} | Y_i \neq Y_j$ **do**
  $D_{ij} = knnsearch(X_{Yi}, X_{Yj})$
  $\mathcal{B}_{ij} = [X_{Yi}, X_{Yj}, Y_i, Y_j] \forall X_{Yi}, X_{Yj} | D_{ij} < \epsilon_B$
end for
return $\mathcal{B}$

RESULTS

Using the three mathematical test functions defined in the "System Under Test" section, we evaluated our search and selection algorithms' ability to discover and identify the known boundaries. We then applied these algorithms to the generation of test scenarios for the UUV simulation and compared the ability of our system to discover interesting test cases compared to a standard Latin hypercube sample set. We evaluated the performance of our iterative GPR search based on its ability to identify features in known data sets and compared it to other sampling methods.

We chose two commonly used statistical sampling techniques as our baseline: Latin hypercube and metropolis sampling. We compared these against the iterative GPR search algorithm by using the following metrics for each of the mathematical test functions: precision, coverage, and convergence. Precision is measured as percentage of samples that are within 5% of a known boundary after a set number of samples. Coverage is the percentage of known boundary with a sample within 1% after a set number of samples. Convergence is the number of samples required to reach 90% coverage of all known boundaries.

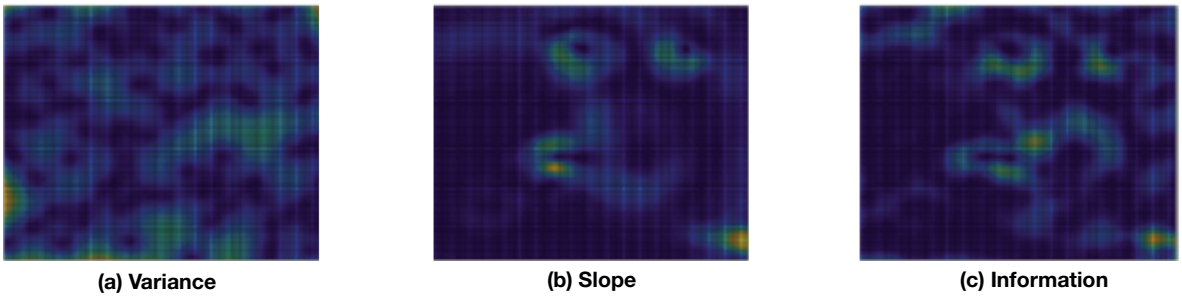


Figure 5. Heatmap of the variance (a), slope (b), and the resulting information metric (c) for a GPR model of the 2d test function after 200 samples.

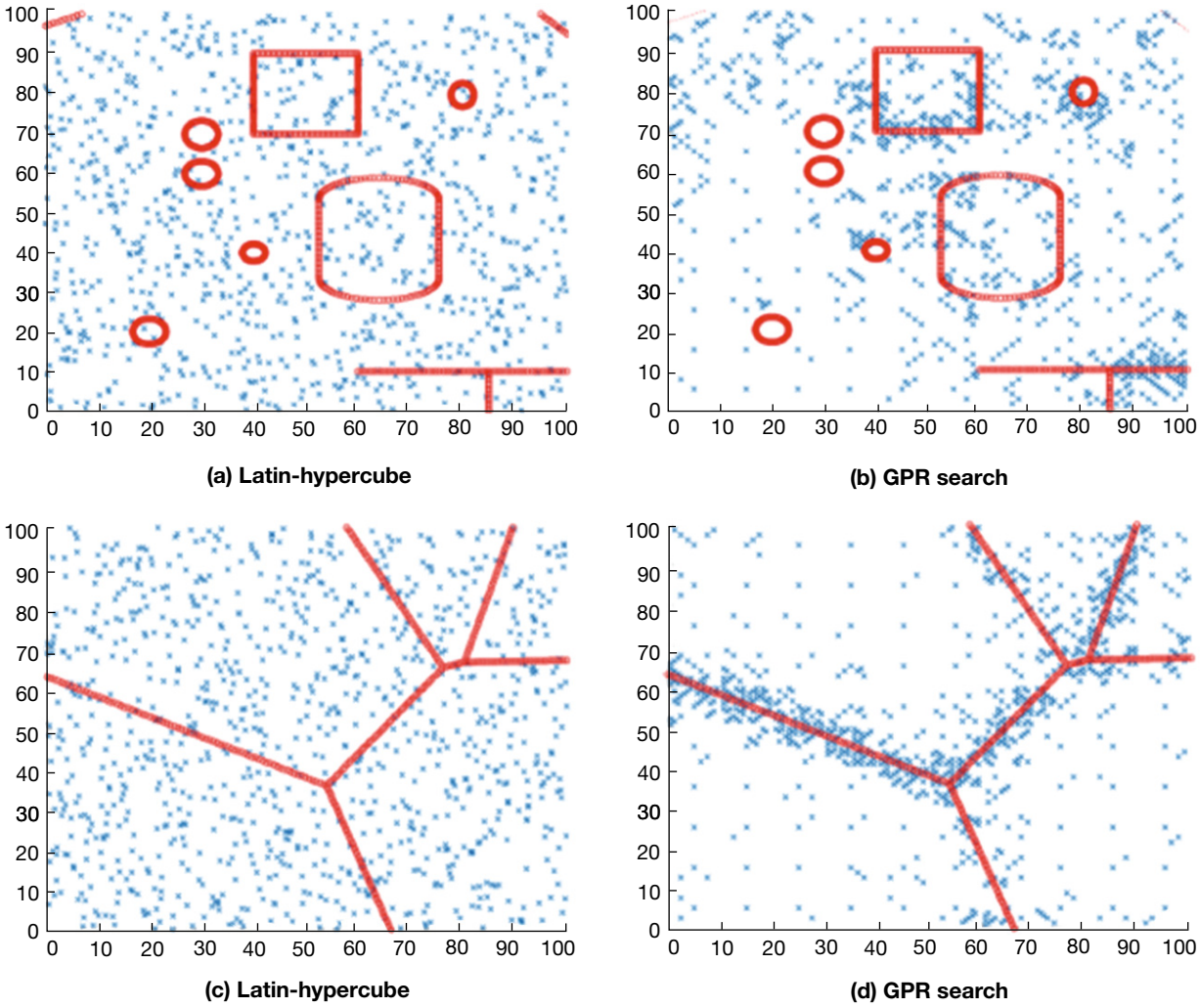


Figure 6. Scatter plot of a GPR search vs. Latin hypercube sampling of the Custom2d (top) and Plates2d test function (bottom). Scenario samples taken are in blue and the true locations of the boundaries are in red.

The GPR-based search outperformed the Latin hypercube sampling method in all of our chosen metrics, particularly in cases in which the boundaries are sharply defined as in our Plates2d test function. As shown in Fig. 5, the GPR-based search concentrated nearly all of its samples in the regions near the boundaries, with minimal cases selected in the uninteresting regions. More importantly, it also managed to obtain near full coverage of the boundary in under half the cases of the uniform sampling of the Latin hypercube method. One thing that becomes apparent immediately in the performance comparison between the Plates2d and Plates3d functions is that the added dimension greatly increases the number of cases necessary to obtain coverage of the boundaries.

Evaluation of the two boundary ID methods (DBSCAN and mean-shift clustering) was performed on the three mathematical test functions. Metrics to evaluate the performance of boundary ID must account for the fact that the search and identification steps are highly coupled. Poor performance by the search algorithm in sampling regions near performance boundaries will inevitably lead to poor performance in boundary identification. For example, if none of the samples returned by the search algorithm are near a truth boundary, it is a hopeless task for the identification algorithm to extract successful boundaries from the data. For this reason, the boundary ID methods were evaluated on a data set consisting of a large number of Latin hypercube samples. In this manner, the identification algorithms can be evaluated independently of the search algorithms, where the number of samples for each test function provides sufficient coverage of the boundaries. Additionally, the precision and recall of the identified boundaries was calculated based on the samples from the data set that were near the *a priori* truth boundaries. These precision and recall results are presented in Figs. 6 and 7 and Table 1.

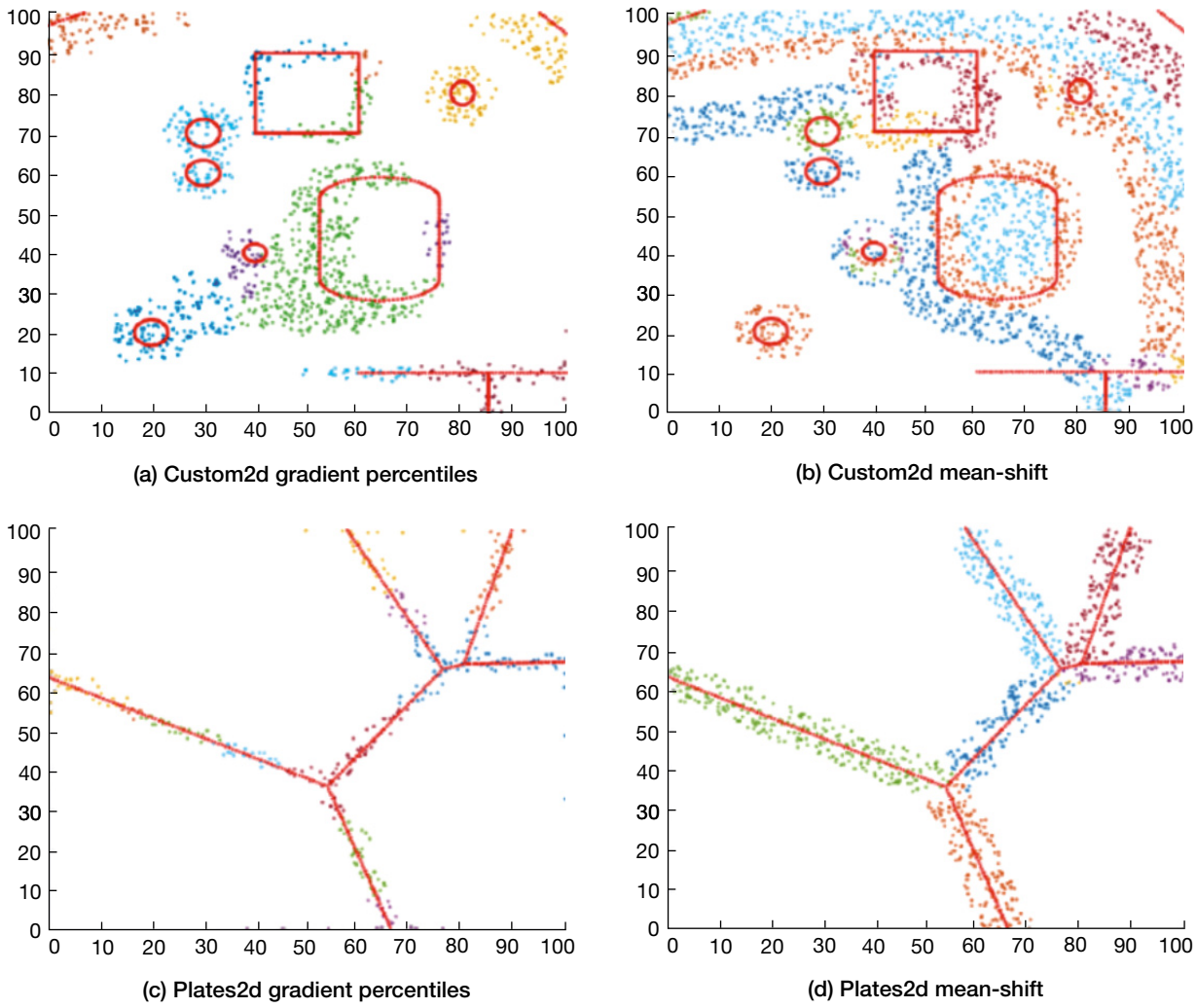


Figure 7. Boundary ID results for a 5000-sample data set. Colors of sample points indicate a unique boundary.

Table 1. Comparison of boundary ID methods

Test System	Gradient Percentiles (%)	Mean-Shift (%)
Custom2d	**Based on 5,000 Latin hypercube samples**	
Precision	75.6	50.2
Recall	57.6	86.1
Plates 2d	**Based on 5,000 Latin hypercube samples**	
Precision	96.6	100
Recall	33.4	87.7
Plates 3d	**Based on 15,000 Latin hypercube samples**	
Precision	83.8	100
Recall	89.2	60.5

CONCLUSIONS

In this work, we introduce a process for discovering and identifying test cases near performance boundaries, in which a small change in the scenario configuration would cause a large change in the system's performance. By using an active learning approach and a GPR model, we can reduce the number of queries required to find the features of interest.

We also introduce two methods for unsupervised clustering of the resulting samples that group all test cases that exhibit the same transition in performance. We tested the ability of each of our techniques against mathematical test functions where the true boundaries in performance are well known. These test functions are designed to test both our ability to preferentially sample in the regions of interest near a boundary and our ability to correctly identify cases that should be added to the boundary set.

In our tests of the GPR-based search approach, we demonstrate that it can significantly outperform a random search of the state space and can fully describe a boundary in considerably fewer queries. Our tests into higher-dimension test functions demonstrate that although the GPR search cannot solve the underlying problems that come as the number of cases necessary becomes exponentially larger, it still performs better than a uniform random search method such as Latin hypercube sampling.

The two boundary ID methods are both capable of obtaining high precision and recall for correctly classifying samples as being near a boundary. Using the gradient percentile filtering method has a significantly higher precision when identifying boundaries in continuous output space compared to the mean-shift classification method, while the mean-shift method has better precision and recall when identifying boundaries in a space with discrete output values. One issue we encountered is that the performance boundary as a feature in a high-dimensional and sparse data set is not one that has been explored before in the literature.

This research is still ongoing, and many avenues have yet to be explored. In particular, we are interested in methods for scaling up our system to handle even larger numbers of dimensions and test cases. One approach is using localized GPR techniques to speed up model generation and prediction time as the number of samples increases. We are also interested in investigating nonstationary covariance functions for the GPR to handle varying resolution of features.

REFERENCES

[1]Meinke, K., and Nycander, P., "Learning-Based Testing of Distributed Microservice Architectures: Correctness and Fault Injection," *Software Engineering and Formal Methods: SEFM 2015 Collocated Workshops: ATSE, HOFM, MoKMaSD, and VERY*SCART*, D. Bianculli, R. Calinescu, and B. Rumpe (eds.), Springer, Berlin, Heidelberg, pp. 3–10 (2015).

[2]Choi, J., "Model Checking for Decision Making Behavior of Heterogeneous Multi-Agent Autonomous System," PhD dissertation, Cranfield University (2012).

[3]Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., et al., "An Orchestrated Survey of Methodologies for Automated Software Test Case Generation," *J. Syst. Software* **86**(8), 1978–2001 (2013).

[4]Snelson, E., *Tutorial: Gaussian Process Models for Machine Learning*, Gatsby Computational Neuroscience Unit, University College London, http://mlg.eng.cam.ac.uk/tutorials/06/es.pdf (2006).

[5]Durst, P. J., Gray, W., Nikitenko, A., Caetano, J., Trentini, M., and King, R., "A Framework for Predicting the Mission-Specific Performance of Autonomous Unmanned Systems," in *Proc. 2014 IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS 2014)*, Chicago, Illinois, pp. 1962–1969 (2014).

[6]O'Brien, M., Arkin, R. C., Harrington, D., Lyons, D., and Jiang, S., "Automatic Verification of Autonomous Robot Missions," in *Simulation, Modeling, and Programming for Autonomous Robots: 4th International Conference, SIMPAR 2014*, D. Brugali, J. F. Broenink, T. Kroeger, and B. A. MacDonald (eds.), Springer, Cham, Switzerland, pp. 462–473 (2014).

[7]Leucker, M., "Learning Meets Verification," *Formal Methods for Components and Objects: First International Symposium, FMCO 2002*, F. S. de Boer, M. Bonsangue, S. Graf, and W.-P. de Roever (eds.), Springer, Berlin, Heidelberg, pp. 127–151 (2007).

[8]Feng, L., Lundmark, S., Meinke, K., Niu, F., Sindhu, M. A., and Wong, P. Y. H., "Case Studies in Learning-Based Testing," in *Testing Software and Systems*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell et al. (eds.), Berlin, Heidelberg, Springer, pp. 164–179 (2013),

[9]Raffelt, H., Merten, M., Steffen, B., and Margaria, T., "Dynamic Testing via Automata Learning," *Int. J. Software Tools Tech. Transf.* **11**(4), 307–324 (2009).

[10]Howar, F., Steffen, B., and Merten, M., "From Zulu to Rers," *Leveraging Applications of Formal Methods, Verification, and Validation: Third International Symposium, ISoLA 2008*, T. Margaria and B. Steffen (eds.), Berlin, Heidelberg, Springer, pp. 687–704 (2010).

[11]Vernaza, P., Guttendorf, D., Wagner, M., and Koopman, P., "Learning Product Set Models of Fault Triggers in High-Dimensional Software Interfaces," in *Proc. 2015 IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS 2015)*, Hamburg, Germany, pp. 3506–3511 (2015).

[12]Ester, M., Kriegel, H.-P., Sander, J., and Xu, X., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proc. Second International Conf. on Knowledge Discovery and Data Mining (KDD-96)*, Portland, OR, pp. 226–231 (1996).

[13]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).

Galen E. Mullins, Research and 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 and a master's degree in applied physics from Johns Hopkins University. 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.

R. Chad Hawthorne, Force Projection Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

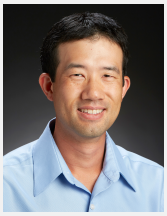
R. Chad Hawthorne is the Assistant Group Supervisor of the Ocean Systems and Engineering Group in APL's Force Projection Sector and the Principal Investigator for the Range Adversarial Planning Tool (RAPT). His e-mail address is chad.hawthorne@jhuapl.edu.

Jordan D. Appler, Force Projection Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Jordan Appler is an electrical engineer in the Ocean Systems and Engineering Group in APL's Force Projection Sector and contributed to the development of the Range Adversarial Planning Tool (RAPT) autonomy and simulator. He received a B.S. in electrical engineering from the University of Maryland, College Park, in 2015. His e-mail address is jordan.appler@jhuapl.edu.

Michael H. Biggins, Force Projection Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Mike Biggins is a software engineer in the Ocean Systems and Engineering Group in APL's Force Projection Sector and led the autonomy software development for the Range Adversarial Planning Tool (RAPT). His background is in multivehicle autonomous control system development, for which he has been recognized twice as an R. W. Hart Prize recipient. He received a B.S. in computer science from the University of Maryland, College Park, in 2012 and an M.S. from Johns Hopkins University in 2016. His e-mail address is michael.biggins@jhuapl.edu.



Kevin Chiou, Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Kevin Chiou is a software engineer in the Enterprise Systems Group in APL's Asymmetric Operations Sector. His e-mail address is kevin.chiou@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.

Johan D. Stewart, Force Projection Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Johan Stewart is a Senior Professional Staff member of the Ocean Systems and Engineering Group in APL's Force Projection Sector. He led the design, installation, and deployment of the Range Adversarial Planning Tool (RAPT) at the Naval Undersea Warfare Center, Keyport. His e-mail address is johan.stewart@jhuapl.edu.

Adam S. Watkins, Force Projection Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Adam Watkins is the Section Supervisor of the Intelligent Sensor Technologies Section in the Ocean Systems and Engineering Group in APL's Force Projection Sector. He provided leadership in the conceptual development of the Range Adversarial Planning Tool (RAPT). Adam holds a Ph.D. in aerospace engineering from the University of Florida. His research is focused on autonomous system control and image processing. His e-mail address is adam.watkins@jhuapl.edu.