Identifying Performance Regression Conditions for Testing & Evaluation of Autonomous Systems

Paul Stankiewicz[1,2] and Marin Kobilarov[2]

***Abstract*— This paper addresses the problem of identifying whether/how a black-box autonomous system has regressed in performance when compared to previous versions. The approach analyzes performance datasets (typically gathered through simulation-based testing) and automatically extracts test parameter clusters of predicted performance regression. First, surrogate modeling with quantile random forests is used to predict regions of performance regression with high confidence. The predicted regression landscape is then clustered in both the output space and input space to produce groupings of test conditions ranked by performance regression severity. This approach is analyzed using randomized test functions as well as through a case study to detect performance regression in autonomous surface vessel software.**

I. Introduction

Given the need to trust autonomous systems in safety-critical domains, it is well understood that rigorous testing and evaluation (T&E) methods are needed to ensure robust performance [1]. This is especially critical due to the "black-box" nature of complex decision-making components. Field experiments in realistic settings are the ideal avenue to perform autonomy T&E; however, they are expensive and time-consuming. Simulation-based testing offers an alternative that can produce large datasets under a wide array of test conditions for statistical performance analysis.

A significant technical gap within autonomy T&E exists, however, when it comes to *comparing* the performance of the current system against previous versions, i.e., performance regression testing. While traditional regression testing is common in software development for code analysis and fault detection [2], autonomous systems introduce a new challenge where it also becomes necessary to ensure that software changes do not adversely affect the holistic performance and behavior of the autonomy in unexpected ways, particularly when changes are constantly made during active development. When using simulation-based testing to study holistic performance, the question becomes how to analytically compare the performance and failure regions of one dataset to another. Figure 1 illustrates this notion of performance regression in an obstacle avoidance example, where performance landscapes for two different autonomy versions (A and B) are shown. These scatter plots represent simulated scenarios for various test parameter combinations of the obstacle position. The color shows the performance score assigned to each scenario based on the autonomy's actions. Two failure regions (red and gray) are highlighted between the datasets. The red failure region exists in version A, but is fixed based on software changes made for version B. The gray region does not exist in version A, but rather appears as a new, unexpected failure mode in version B. In this sense, the gray region represents a region of performance regression characterized by a decrease in performance.

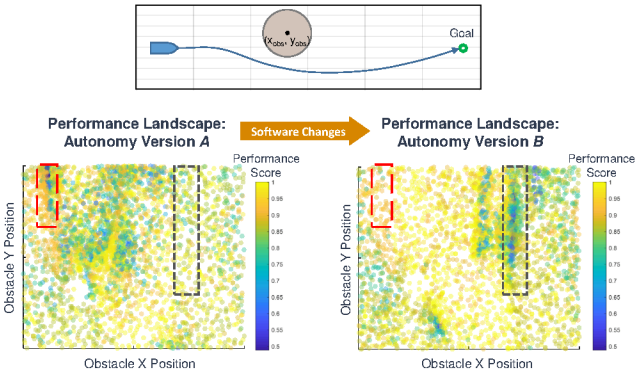


Fig. 1: Simulation performance datasets for two versions of obstacle avoidance software. The red box highlights performance improvement whereas the gray box highlights performance regression.

The underlying research objective highlighted by the example of Fig. 1 is to apply statistical learning to analytically compare two (potentially high-dimensional) performance landscapes. Such an analysis helps ensure that more "mature" releases of autonomy software reduce the failure space and do not produce new, unexpected failure modes. Alternatively, this process could also be used to conduct a principled performance comparison between two different autonomy strategies (or two different autonomy products) tested within the same context. To the best of the authors' knowledge, this paper presents one of the first frameworks specifically designed to extract regression regions based on the holistic performance of the system from non-identical datasets. The remainder of this paper is organized with Section II providing related work, Section III formalizing the problem definition, Section IV detailing the methodology for identifying and analyzing performance regression regions, and Section V evaluating the efficacy of the approach through randomized test functions and an autonomous surface vessel case study.

II. Related Work

Autonomy T&E encompasses a large swath of research summarized in [3]. These areas include fault detection and software robustness [4], [5], formal methods [6], and simula-

[1]P. Stankiewicz is with the Johns Hopkins Applied Physics Laboratory, 11100 Johns Hopkins Road, Laurel MD, 20723, USA. paul.stankiewicz@jhuapl.edu

[2]P. Stankiewicz and M. Kobilarov are with the Department of Mechanical Engineering, Johns Hopkins University, 3400 N Charles Street, Baltimore, MD 21218, USA. pstanki2@jhu.edu, marin@jhu.edu

tion scenario generation [7]. With regards to the latter, several veins of research [8], [9], [3] have shown that adaptive scenario generation is an effective method to identify unknown failure modes within high-dimensional testing spaces (i.e., the space of all possible test conditions under consideration). The work in [10] uses the cross-entropy method to detect rare-event failures in autonomous driving scenarios. Previous work by the authors [11] has also explored adaptive scenario generation with the purpose of discovering performance boundaries in autonomous systems, defined as regions with a large performance gradient. The datasets of Fig. 1 serve as an example to this form of adaptive sampling.

While simulation-based testing is an effective way to generate performance datasets, there is more limited literature on performance regression techniques. The work in [12] considered auto-generating scenarios for regression tests in the context of self-driving car maneuvering. The problem posed here, however, is the more general comparison of performance datasets, one which shares similarities with the change detection field. Applications within this field usually fall into detecting changes within time-series information [13] or between images [14], [15]. Generalized methods that detect differences in the underlying probability distributions between datasets also exist [16], [17].

III. Problem Setup

Let the autonomous system under test (SUT) be treated as a function $\mathcal{F} : \mathcal{X}^D \to \mathcal{Y}$ that maps a D-dimensional testing space $\mathcal{X}^D = \mathcal{X}_1 \times \ldots \times \mathcal{X}_d$ to a scalar performance space $\mathcal{Y} \subset \mathbb{R}$. This research tackles the problem of comparing two different SUTs, $\mathcal{F}_A$ and $\mathcal{F}_B$, where $\mathcal{F}_A \neq \mathcal{F}_B$. Assuming both SUTs are evaluated on the same testing space $\mathcal{X}^D$, we define the change in performance over the entire testing space as $\Delta\mathcal{Y} = \mathcal{Y}_B - \mathcal{Y}_A = \mathcal{F}_B(\mathcal{X}^D) - \mathcal{F}_A(\mathcal{X}^D)$. Given these definitions, we aim to identify the subset of regression regions characterized by a decrease in performance: $\mathcal{X}^- = \{\mathcal{X}^D \mid \Delta\mathcal{Y} < 0\}$.

Because an exact characterization of the regression regions is intractable, we aim to estimate them using samples collected from each SUT (in the form of simulation datasets). A scenario $\boldsymbol{x} = [x_1, \ldots, x_D]^T \in \mathcal{X}^D$ is a specific instantiation of the testing space, where $x_k \in \mathcal{X}_k$ is the value of each test parameter. The performance score $y = \mathcal{F}(\boldsymbol{x}) \in \mathcal{Y}$ is the output of each simulated scenario. Further, let sets of N scenarios $X = \{\boldsymbol{x}_i\}_{i=1}^N$ and their resulting performance scores $Y = \{y_i\}_{i=1}^N$ be combined to form a study $S = \{X, Y\}$. These studies could be generated through the adaptive scenario generation techniques described in Section II or simply through randomized Monte Carlo methods. While we assume that both SUTs are evaluated on the same testing space, the resulting studies S_A and S_B (collected from $\mathcal{F}_A$ and $\mathcal{F}_B$ respectively) need not be identical. For the purposes of performance regression analysis, we aim to estimate the regression regions by identifying a set of regression clusters R_* that are uniquely determined by both the severity of the performance decrease between S_A and S_B, and also the location within the testing space.

IV. Approach

The overall approach taken to identify and characterize the regression clusters is a combination of both supervised and unsupervised learning. Fig. 2 shows a graphical flowchart of the process. The algorithms that follow are assumed to operate on the normalized testing space $\bar{\mathcal{X}}^D \in [0,1]^D$ and normalized performance space $\bar{\mathcal{Y}} \in [0,1]$, where the $\bar{(\cdot)}$ operator indicates data that has been normalized to this range based on its maximum and minimum possible values.

A. Regression Modeling

The first step of the performance regression analysis entails fitting surrogate models to each normalized study dataset $\bar{S}_A$ and $\bar{S}_B$. These surrogate models $\mathcal{M} : \bar{\mathcal{X}}^D \to \hat{\mathcal{Y}}$ attempt to approximate the input-output relationship of the SUT, where $\hat{\mathcal{Y}}$ is the predicted normalized performance space. Given that $\bar{S}_A \neq \bar{S}_B$, regression regions are estimated by first fitting surrogate models $\mathcal{M}_A$ and $\mathcal{M}_B$ to each dataset $\bar{S}_A$ and $\bar{S}_B$, respectively. Regression regions are then estimated based the predicted response of each model to the opposite dataset:

$$[\hat{Y}_{AB}^{lb}, \hat{Y}_{AB}, \hat{Y}_{AB}^{ub}] = \mathcal{M}_B(\bar{X}_A), \tag{1}$$

$$[\hat{Y}_{BA}^{lb}, \hat{Y}_{BA}, \hat{Y}_{BA}^{ub}] = \mathcal{M}_A(\bar{X}_B), \tag{2}$$

$$\Delta\hat{Y}_A = \hat{Y}_{AB} - Y_A, \tag{3}$$

$$\Delta\hat{Y}_B = -(\hat{Y}_{BA} - Y_B). \tag{4}$$

Here, $\hat{Y}_{AB}$ and $\hat{Y}_{BA}$ are the predicted scores from studies A and B on models B and A, respectively. The superscripts $(\cdot)^{lb}$ and $(\cdot)^{ub}$ represent the lower and upper prediction bounds under the assumption that the choice of $\mathcal{M}$ can provide these values. These measures are translated to predict the bounds on the performance regression of each dataset as

$$\Delta\hat{Y}_A^{lb} = \hat{Y}_{AB}^{lb} - Y_A, \tag{5}$$

$$\Delta\hat{Y}_A^{ub} = \hat{Y}_{AB}^{ub} - Y_A, \tag{6}$$

$$\Delta\hat{Y}_B^{lb} = -(\hat{Y}_{BA}^{ub} - Y_B), \tag{7}$$

$$\Delta\hat{Y}_B^{ub} = -(\hat{Y}_{BA}^{lb} - Y_B). \tag{8}$$

A combined study set $\bar{S}_* = \{\bar{X}_*, \Delta\hat{Y}_*\}$, where $\bar{X}_* = \bar{X}_A \cup \bar{X}_B$ and $\Delta\hat{Y}_* = \Delta\hat{Y}_A \cup \Delta\hat{Y}_B$, is then created to predict the change from $\mathcal{F}_A$ to $\mathcal{F}_B$. The set of samples representing performance regression can be estimated as $\bar{S}_*^- = \{\bar{S}_* \mid \Delta\hat{Y}_* < 0\}$. This combined set augments the number of samples ($N_* = N_A + N_B$) for improved fidelity in cluster analysis. Alternatively, the predicted regression bounds $\Delta\hat{Y}_*^{lb} = \Delta\hat{Y}_A^{lb} \cup \Delta\hat{Y}_B^{lb}$ and $\Delta\hat{Y}_*^{ub} = \Delta\hat{Y}_A^{ub} \cup \Delta\hat{Y}_B^{ub}$ could also be used to define the combined study set. Using the lower bound offers a conservative approach and prioritizes extracting regions with the greatest possible regression. The upper regression bound, meanwhile, lets the analysis prioritize performance regression that is occurring with high confidence, meaning that even the upper bound is predicted to have regressed.

The choice of surrogate model in this step is important since autonomous systems can exhibit unpredictable

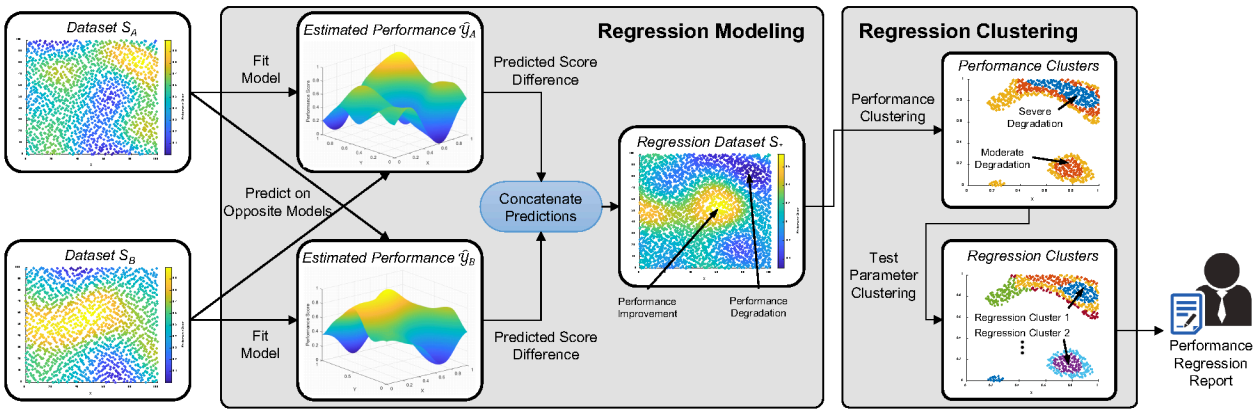


Fig. 2: Overall performance regression analysis approach. Surrogate modeling is used to normalize each study to the other and predict whether samples have regressed. Unsupervised clustering is then used to extract unique regression clusters in both the severity of the performance decrease and its location within the testing space.

emergent behavior that results in highly nonlinear and/or discontinuous performance (as exampled in Fig. 1). Further, the surrogate model must also be able to accommodate high-dimensional testing spaces (i.e., tens of dimensions) and large sample sets. For these reasons, this work adopts quantile random forests (QRFs) [18] as the surrogate modeling technique, although as discussed in Section V, alternative methods such as deep neural networks or Gaussian process models could be easily substituted. QRFs are well-suited to handle the challenges described above through their non-parametric structure, while also providing confidence bounds on predictions. In general, for a response variable Y, predictor variable X, and underlying conditional distribution $\mathbb{P}(Y \leq y|X = \boldsymbol{x})$, quantile regression methods attempt to estimate the conditional α-quantiles, defined as

$$Q_\alpha(\boldsymbol{x}) = \inf\{y : \mathbb{P}(Y \leq y|X = \boldsymbol{x}) \geq \alpha\}. \tag{9}$$

Quantile random forests achieve estimates $\hat{Q}_\alpha(\boldsymbol{x})$ of these α-quantiles using the inherent bagging structure of random forests. In essence, rather than simply storing the weighted mean of all observations for a given leaf node (as is the case with traditional random forests), QRFs extend this by storing the *values* of all observations from each tree to compute an estimate of the full conditional distribution,

$$\hat{\mathbb{P}}(Y \leq y|X = \boldsymbol{x}) = \sum_{i=1}^{N} w_i(\boldsymbol{x})\mathbb{I}_{\{Y_i \leq y\}}, \tag{10}$$

where w_i are the typical weights calculated when training the random forest and $\mathbb{I}$ is the indicator function (see [18] for details). The α-quantile estimates can then be calculated by using Eq. (10) in Eq. (9). Most importantly, $\hat{Q}_\alpha(\boldsymbol{x})$ can be used to provide confidence bounds on the predictions made by the QRF model. Using a 95% prediction interval, the predictions provided by Eq. (1) and (2) become

$$[\hat{Y}^{lb}, \hat{Y}, \hat{Y}^{ub}] \widehat{=} [\hat{Q}_{0.025}(\boldsymbol{x}), \hat{Q}_{0.5}(\boldsymbol{x}), \hat{Q}_{0.975}(\boldsymbol{x})]. \tag{11}$$

B. Regression Clustering

It is possible for $\bar{S}_*^-$ to contain samples from all areas of the testing space with varying degrees of performance regression. Thus, to aid in understanding the reason for the performance change, it is useful to cluster the regression samples in both (i) the degree of performance decrease and (ii) the region of the testing space where regression occurs.

1) Performance Space Clustering: The regression dataset is first clustered on the $\Delta\mathcal{Y}^-$ space such that performance degradation can be separated based on its severity. Similar to the choice of surrogate model used to represent the data, the algorithm used for unsupervised clustering is dependent on the structure of the data and the fact that the number of regression clusters is not known *a priori*. The clustering algorithm implemented for performance space clustering uses Gaussian mixture models (GMMs) fit with the iterative Expectation-Maximization algorithm [19], although alternative algorithms could also be used. The assumption of Gaussian structure to the performance clusters imposes limitations to their shape, but also adds useful analysis properties by giving estimates of the cluster center (mean) and spread (covariance). A set of j performance clusters is defined as $P_j = \{p_i\}_{i=1}^{j} = GMM(\Delta\hat{Y}_*^-, j)$, where each p_i is a unique performance cluster and GMM is the clustering operation of [19]. To overcome the assumption inherent to GMMs that the number of clusters is known, multiple models are fit, where each assumes a different number of clusters $j \in [1, j_{max}]$, and each is evaluated using its averaged silhouette score of the clustered samples [20]. The final performance cluster set P_* is then chosen as the P_j set with the maximum associated silhouette score.

2) Testing Space Clustering: The second clustering step takes samples within each performance cluster and further groups them in the $\mathcal{X}^D$ space. This aids in diagnosing regions of the testing space that cause different forms of performance regression. In other words, as illustrated in Fig. 2, performance regression could occur at multiple different locations within the testing space and it is useful to separate each of these clusters.

Once again, the choice of clustering algorithm is important given the new domain $\mathcal{X}^D$. Imposing structure requirements on the data (such as in GMMs) in the testing space proved to be too limiting. Additionally, there are potentially many

more individual clusters within this space, amplifying the challenge that the number, size, and locations of regression clusters are not known *a priori*. Thus, the testing space clustering makes use of density-based methods (specifically DBSCAN [21]) to alleviate some of these problems, which can accommodate an unknown number of clusters and imposes loose requirements on their shape.

Let $\bar{X}_p$ be the scenarios for which their respective performance scores belong to each $p \in P_*$, i.e., $\bar{X}_p = \{\bar{X}_* \mid \Delta\hat{Y}_* \in p\}$. Then, in a similar fashion to the performance clustering step, a set of regression clusters is calculated as $R = DBSCAN(\bar{X}_p, \sigma)$, where each $r \in R$ is now a unique regression cluster of scenarios, $DBSCAN$ is the clustering operation of [21], and σ denotes the clustering algorithm's hyperparameters. Because these hyperparameters significantly affect the clustering process, several cluster sets are calculated over variations to the hyperparameters and each is again evaluated by its averaged silhouette score. The regression cluster set for a given p, deemed R_p, is the set with the maximum silhouette score. This clustering process is then repeated using scenarios for each $p \in P_*$ such that the final regression cluster set is $R_* = \{R_p\}_{p=1}^{p_{max}}$.

3) Feature Scaling: When analyzing realistic SUTs, the well-known curse of dimensionality dilutes the effectiveness of clustering if applied directly to high-dimensional data; thus, feature importance scaling is applied during testing space clustering to reduce the size of the testing space based on the importance of each test parameter. This is achieved by fitting a random forest surrogate model $\mathcal{M}_*$ to the combined study set $\bar{S}_*$. The importance β_k of each k-th parameter is determined as in [22] by how much the out-of-bag mean-squared error of $\mathcal{M}_*$ changes through different parameter permutations. Normalized parameter importance is given as $\bar{\beta}_k = \beta_k / \sum_{\ell=1}^{D} \beta_\ell$. The normalized importance values are then used as weights to scale the Euclidean distance d between two scenarios $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ as:

$$d(\boldsymbol{x}_m, \boldsymbol{x}_n) = \sqrt{(\boldsymbol{x}_m - \boldsymbol{x}_n)^T B (\boldsymbol{x}_m - \boldsymbol{x}_n)}, \qquad (12)$$

where $B = \text{diag}(\bar{\beta}_1 \dots \bar{\beta}_D)$. This weighted distance function is used during testing space clustering and has the effect of minimizing unimportant test parameters.

V. Analysis

A. Performance on Test Functions

The proposed framework is first evaluated on a Monte Carlo study of auto-generated test functions from which a ground truth regression landscape can be calculated. Each randomized test function is meant to be representative of the types of performance surfaces seen in autonomy testing datasets. Specifically, the test functions generate $\mathcal{Y}$ to consist of several "plateaus" within $\mathcal{X}^D$. The quantity, performance score, and location of each plateau are all randomized. The boundary between adjacent plateaus is linearly smoothed and measurement noise is added to each sample such that $y = y_{truth} + \mathcal{N}(0, \sigma^2)$, where $\sigma = 0.02$. For each study, two test functions are randomly generated and assigned as $\mathcal{F}_A$ and $\mathcal{F}_B$ (examples shown in Fig. 3a).

Because unsupervised cluster evaluation is generally more subjective in nature, this portion of the analysis primarily focuses on the regression modeling aspects of the approach. In order to compare different modeling approaches, the QRF of Section IV was compared against a Gaussian process regression (GPR) model and a five-layer neural network (DNN), all of which used hyperparameters that were optimized to minimize prediction error. The performance of the regression framework using each model was then evaluated across various combinations of D and N to determine scalability for high-dimensional testing spaces and large datasets. Ten studies were run for each combination of D and N (each with new test functions) and the results were averaged.

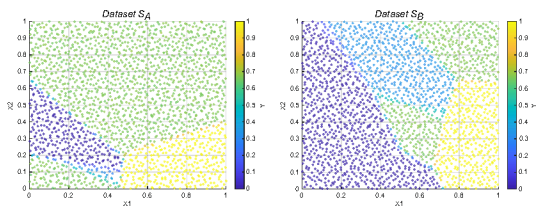
Figure 3 shows the performance of each model in its predictions of the regression dataset S_*. When comparing to the ground truth regression between pairs of randomized test functions, the first evaluation metric shows the five-fold cross validation mean absolute error (MAE) of $\Delta\hat{Y}_*$. Each model exhibits the expected trend that MAE increases for large values of D and small values of N. The QRF model has lower MAE scores than the GPR model over all combinations of D and N, but is slightly outperformed by the DNN model for large sample sizes, indicating that there may be conditions to apply different modeling techniques.

The second evaluation metric shown in Fig. 3 is the percentage of $\Delta\hat{Y}_*$ predictions outside the regression bounds (i.e., the 95% prediction interval $[\Delta\hat{Y}_*^{lb}, \Delta\hat{Y}_*^{ub}]$). For the GPR and DNN models, the 95% prediction intervals of Eq. (1) and (2) are calculated as $[\hat{Y}^{lb}, \hat{Y}^{ub}] = \hat{Y} \pm 1.96\sigma$, where the standard deviation σ for the GPR model is calculated through its inherent covariance structure, and for the DNN model is approximated using the model's five-fold cross validation root-mean-square error. The percentage of outliers for all models is roughly consistent with a 95% confidence interval, however, the QRF exhibits the lowest outlier percentage, where typically only $\sim$1–2% of predictions lie outside the regression bounds. While the prediction interval is centered on the mean for the GPR and DNN models, the prediction interval of the QRF model could be asymmetrical based on the estimated percentiles, which likely results in the increased robustness.

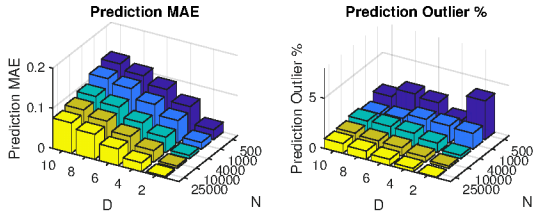
Overall, the QRF surrogate modeling offers a good balance of reducing MAE while also minimizing the number of predictions that lie outside the 95% predicted regression bounds. It is evident, however, that each surrogate modeling technique may offer some benefits based on the characteristics of the dataset and could be easily substituted into the regression analysis framework.

B. ASV Dataset Case Study

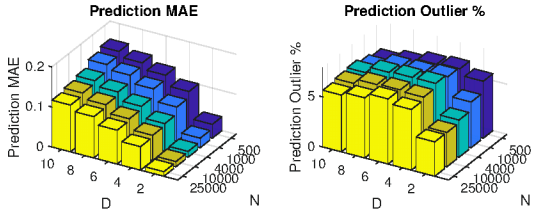
We now apply the framework of Section IV to simulation sets generated by an ASV navigation algorithm. The application of interest is that the ASV must avoid other vessels according to the International Regulations for



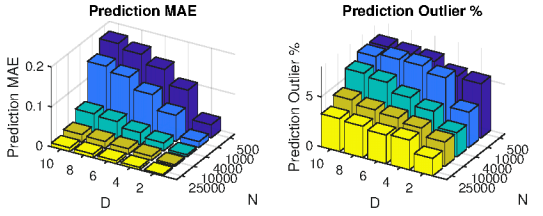
(a) Example test functions for $D = 2$ and $N = 1000$.



(b) QRF surrogate modeling



(c) GPR surrogate modeling



(d) DNN surrogate modeling

Fig. 3: Regression modeling performance on the prediction of $\Delta\hat{Y}_*$ for randomized test functions.

Prevention of Collisions at Sea (COLREGS) [23]. In short, based on the vessel encounter geometry, COLREGS dictate a set of maneuvering requirements that reduce the risk of collision. They can be thought of as "rules of the road" for marine surface craft. Thus, it is important to determine whether the ASV decision-making complies with COLREGS in all possible geometries of an oncoming target ship (TS) through the 3D testing space of Fig. 4a. The normalized performance score $\bar{y} \in [0, 1]$ of each scenario is calculated from a combination of mission criteria, safety criteria, and COLREGS compliance according to [24]. Using these testing and performance spaces, two versions of the ASV software were evaluated using 3000 adaptively-generated scenarios [11]: version A resulting in dataset S_A of Fig. 4b, and version B resulting in dataset S_B of Fig. 4c. Version A serves as the baseline whereas version B included software changes that encouraged the ASV to make turns in the direction away from the current location of the target ship for better performance in overtaking scenarios (scenarios with near-zero TS relative heading).

The regression analysis produces the regression dataset S_* in Fig. 4d, which predicts changes in performance $\Delta\hat{Y}_*$ over the entire testing space. The five-fold cross validation MAE of $\Delta\hat{Y}_*$ was 0.015 and the percentage of samples outside the prediction bounds was 1.7% (using 95% confidence intervals). Both of these measures indicate even better performance than that seen on the test functions when applying the regression modeling to real autonomy datasets. Qualitatively, Fig. 4d shows that the software changes to version B did in fact improve performance in overtaking scenarios with high TS speeds and a TS relative heading of 10 deg; however, these changes also produced unexpected performance regression in other areas of the testing space. The "top 3" regression clusters extracted from $\bar{S}_*^-$ are then shown in Fig. 4e. These clusters reveal that the most severe performance regression occurs in scenarios with a TS relative heading between 120–180 deg and is then split between low and high TS speeds.

Further analysis into the scenarios comprising the R_1 cluster reveals that the software change encouraging turns away from the target ship location produced an unintended side effect in high-speed crossing scenarios when ownship is expected to stand-on. Figure 4f shows a pair of representative scenarios from R_1, one from each S_A and S_B. Before the software change, the ASV turns to port to avoid the noncompliant target ship. Although this is undesirable behavior with regards to COLREGS, the maneuver successfully mitigated the collision risk. After the software change, however, the ASV now turns to starboard (away from the current location of the target ship), but crosses the bow of the target ship and comes within the collision radius of both vessels. The performance regression identified by R_1 encompasses a small region of the testing space, but the new ASV behavior within this region is extremely unsafe and would need to be addressed with additional updates.

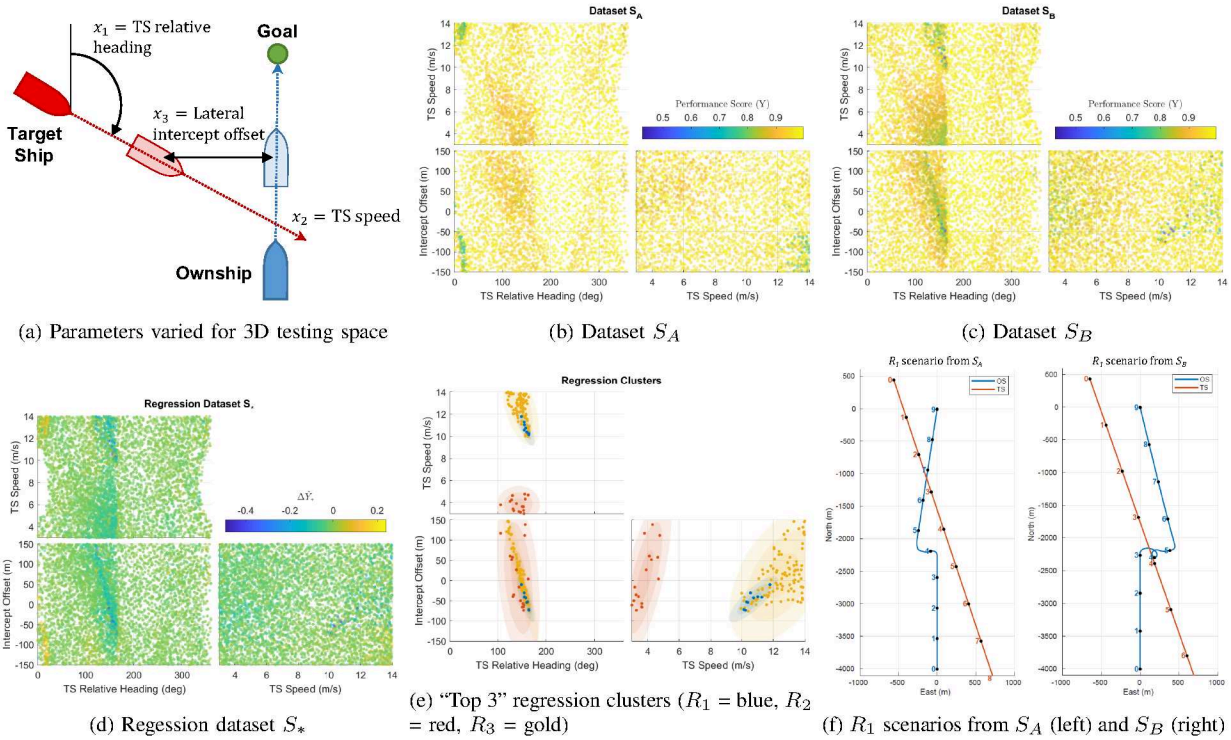
VI. Conclusion

This work presented a new algorithmic approach for characterizing performance regression during black-box autonomous system testing & evaluation. Specifically, the approach uses a combination of surrogate modeling and unsupervised clustering to extract unique groupings of test conditions that have decreased in performance. Monte Carlo analysis on randomized test functions, coupled with a case study on detecting performance regression in autonomous surface vessel software, show the efficacy of the approach to predict performance regression with high-confidence and report unique regression clusters to the user.

Future work is currently focused on algorithms to extract representative samples from each regression cluster. In this way, a user could easily examine the inherent behavior changes of a regression cluster (in a similar manner to Fig. 4f) in an automated fashion. Additional work is also planned to improve dimensionality reduction such that influential test parameters are further emphasized in the final regression clusters.

References

[1] F. Batsch, S. Kanarachos, M. Cheah, R. Ponticelli, and M. Blundell, "A taxonomy of validation strategies to ensure the safe operation of highly



(a) Parameters varied for 3D testing space

(b) Dataset S_A

(c) Dataset S_B

(d) Regession dataset S_*

(e) "Top 3" regression clusters (R_1 = blue, R_2 = red, R_3 = gold)

(f) R_1 scenarios from S_A (left) and S_B (right)

Fig. 4: Case study results for the regression analysis applied to testing autonomous surface vessel (ASV) software. The R_1 regression cluster of Fig. 4e identifies a performance regression visualized in Fig. 4f, where software updates nearly cause a collision by requiring the ASV to now turn to starboard during high-speed noncompliant encounters.

automated vehicles," *Journal of Intelligent Transportation Systems*, pp. 1–20, 2020.

[2] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software testing, verification and reliability*, vol. 22, no. 2, pp. 67–120, 2012.

[3] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive stress testing with reward augmentation for autonomous vehicle validatio," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 163–168.

[4] A. L. Christensen, R. O'Grady, M. Birattari, and M. Dorigo, "Fault detection in autonomous robots based on fault injection and learning," *Autonomous Robots*, vol. 24, no. 1, pp. 49–67, 2008.

[5] K. Meinke and P. Nycander, "Learning-based testing of distributed microservice architectures: Correctness and fault injection," in *SEFM 2015 Collocated Workshops*. Springer, 2015, pp. 3–10.

[6] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–41, 2019.

[7] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020.

[8] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing s-taliro as an automatic test generation framework for autonomous vehicles," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1470–1475.

[9] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1555–1562.

[10] M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," *Advances in Neural Information Processing Systems*, vol. 31, pp. 9827–9838, 2018.

[11] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, "Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles," *Journal of Systems and Software*, vol. 137, pp. 197–215, 2018.

[12] E. Rocklage, H. Kraft, A. Karatas, and J. Seewig, "Automated scenario generation for regression testing of autonomous vehicles," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 476–483.

[13] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and information systems*, vol. 51, no. 2, pp. 339–367, 2017.

[14] F. Melgani, G. Moser, and S. B. Serpico, "Unsupervised change detection methods for remote sensing images," in *Image and Signal Processing for Remote Sensing VII*, vol. 4541. International Society for Optics and Photonics, 2002, pp. 211–222.

[15] M. Song, Y. Zhong, and A. Ma, "Change detection based on multi-feature clustering using differential evolution for landsat imagery," *Remote Sensing*, vol. 10, no. 10, p. 1664, 2018.

[16] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Statistical change detection for multi-dimensional data," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 667–676.

[17] P. K. Novak, N. Lavrač, and G. I. Webb, "Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining." *Journal of Machine Learning Research*, vol. 10, no. 2, 2009.

[18] N. Meinshausen, "Quantile regression forests," *Journal of Machine Learning Research*, vol. 7, no. Jun, pp. 983–999, 2006.

[19] G. J. McLachlan and D. Peel, *Finite mixture models*. John Wiley & Sons, 2004.

[20] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[21] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[22] U. Grömping, "Variable importance assessment in regression: linear regression versus random forest," *The American Statistician*, vol. 63, no. 4, pp. 308–319, 2009.

[23] U. S. C. Guard, *Navigation Rules*. United States Department of Transportation, 1999.

[24] P. G. Stankiewicz and G. E. Mullins, "Improving evaluation methodology for autonomous surface vessel colregs compliance," in *OCEANS 2019-Marseille*. IEEE, 2019, pp. 1–7.