Out-of-Distribution Behaviors in Dynamic Off-Road Autonomous Systems

Ege Caglar, Rohan Panicker, Sidharth Talia, Rosario Scalise, Byron Boots
School of Computer Science & Engineering, University of Washington
{cagege, sidtalia, rosario, bboots}@cs.uw.edu; rohpan@uw.edu

***Abstract*—Off-road autonomy implies high-dynamics in varying contexts which introduce numerous opportunities for controllers to fail (both marginally and catastrophically). One major reason for these failures is often that the full-stack system is subject to scenarios that are deemed out-of-distribution (OOD) with respect to the design of the original controllers. In this work, we investigate a formalism for off-road OODs, collect a cross-embodied dataset, describe the data lifecycle, and profile a suite of baseline methods in their ability to accurately determine OOD states for off-road autonomous systems. We find that the Moving Average Derivative Threshold and Isolation Forest methods provide strong baselines. We believe this initial investigation will serve as a foundation for future methods and promote interest in OOD discovery as it pertains to off-road autonomy.**

I. Introduction

Robots deployed in autonomous off-road contexts generate rich data streams as they move through the world. With the popularity of data-driven methods increasing [1], alongside the ubiquity of hardware which enables total information capture [2], we are entering an era in which systems should use their sensor streams to determine whether or not their current behavior is supported by their nominal controllers. This awareness can be useful for downstream risk-mitigation [3], failure-recovery [4], and even sampling states to learn from in the future [5]. However, it is not always straightforward to determine what is considered nominal behavior and what is not, especially in a generic way. This is made even more challenging when taking into account the fact that some controllers might possess learned components or have been learned in an entirely end-to-end fashion.

The field of machine learning has begun to tackle this problem through the study of out-of-distribution (OOD) detection and analysis [6], [7]. This refers to how well a model which was trained on "in-distribution" (ID) data can generalize to data which comes from a fundamentally different distribution (often due to effects like domain-shift or label-shift). Although this framework has been applied to robotics in previous works, the majority have focused on the performance of perception algorithms and visual modalities [8], [9]. Even fewer have looked at OOD detection as a way to reason about the current behavior as it pertains to robot state directly.

In this work, we investigate how to formalize OOD behavior in the context of dynamic off-road autonomy. To achieve this, we consider the entire data lifecycle: including deployment, data collection, filtering, data conditioning, dimensionality reduction, and OOD discovery. We find that it is important to scrutinize each of these steps carefully and consider the holistic data lifecycle as the data is highly heterogeneous, context-sensitive, and ever-evolving in deployed systems. We apply this data lifecycle to two robot morphologies possessing onboard autonomy (the HOUND and the Unitree A1) and collect representative datasets for their behavior when deployed in off-road environments. Given that they encounter shifts in dynamics, scenarios for which their controllers were not specifically designed, and difficulties with state estimation, we hypothesize that OOD states are embedded in these datasests.

We test a set of baseline methods and evaluate their performance on the different datasets. Following this, we propose a set of methods we will explore in the future which make as few assumptions as possible and require little to no tuning.

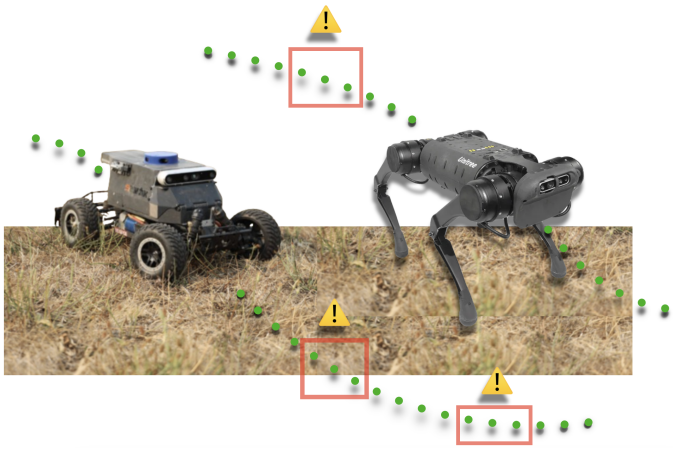

Fig. 1: The HOUND robot (left) and the Unitree A1 robot (right) following trajectories along which OOD behaviors such as slipping, getting stuck, and tipping over have occurred.

This work was (partially) funded by the US Army Research Laboratory CCDC, and Amazon.

In summary, we make the following contributions:

- A **formalization** of OOD behaviors in off-road autonomy.
- A cross-embodiment **dataset** containing OOD states.
- A survey of **baseline methods** for OOD discovery.

II. Methodology

A. Robot Platforms & Data Collection

To explore out-of-distribution events in dynamic off-road autonomous navigation, we examine both quadruped robots and wheeled robots. Our chosen robot platforms are the Unitree-A1, a quadrupedal robot, and the HOUND, a 1/10th-scale four-wheeled off-road autonomy platform.

*1) **A1 Hardware**:* The A1 robot weighs 12 kg and features legs that are designed for minimal inertia. It employs high torque density electric motors with planetary gear reduction for robust performance. Notably, it achieves precise ground force control without relying on force or torque sensors. Its high-performance actuators in the hip, thigh, and knee joints enable comprehensive 3D control of ground reaction forces. The robot is also equipped with contact sensors on each foot for contact detection. Each of the A1's actuators comprises a custom high torque density electric motor paired with a single-stage 9:1 planetary gear reduction. The legs are serially actuated, with the hip and knee actuators co-axially located at the hip joint of each leg to optimize performance. Each joint of the robot can deliver a maximum torque of 33.5 Nm and achieve a maximum speed of 21 rad/s.

*2) **HOUND Hardware**:* The HOUND's hardware architecture is same as that of MuSHR [10]. An NVIDIA Jetson Orin NX serves as the onboard single-board-computer (SBC), chosen for its balanced attributes in terms of cost and size-weight-and-power (SWaP) when compared to the Nano and AGX variants. The physical platform weighs around 4 kg and is capable of reaching speeds exceeding 12 m/s on tarmac surfaces.

For both robots, each trajectory is represented as $\tau_j = \{s_i^{(j)}\}_{i=0}^{T_j}$, where each $s_i^{(j)}$ refers to the state at that time step.

*3) **A1 Data Parameters**:* The state vector $s_i^{(j)}$ for the A1 robot is given by:

$$\mathbf{s}_i := \begin{bmatrix} Q_i & t_i \end{bmatrix}^\intercal$$

$$\mathbf{t_i} := \begin{bmatrix} p & v & R & \omega \end{bmatrix}^\intercal \in \mathbb{R}^{18}$$

where $p = \begin{bmatrix} x & y & z \end{bmatrix}^\intercal$, $v = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^\intercal$, and $R \in SO(3)$ is a 3×3 rotation matrix, and $\omega \in \mathbb{R}^3$ represents the angular velocity.

The joint state vector of the quadruped is given by

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{q}_i \\ \dot{\mathbf{q}}_i \end{bmatrix}$$

$$\mathbf{q}_i := \begin{bmatrix} q_1 & q_2 & \dots & q_{12} \end{bmatrix}^\intercal \in \mathbb{R}^{12}$$

$$\dot{\mathbf{q}}_i := \begin{bmatrix} \dot{q}_1 & \dot{q}_2 & \dots & \dot{q}_{12} \end{bmatrix}^\intercal \in \mathbb{R}^{12}$$

q_i and $\dot{q}_i$ are the joint position and joint velocity state vectors respectively.

*4) **HOUND Data Parameters**:*

$$\mathbf{s_i} := \begin{bmatrix} X_w & Y_w & Z_w & \phi & \theta & \psi & V_b & \omega_b & I_m \end{bmatrix}^\intercal \in \mathbb{R}^9$$

where X_w, Y_w, Z_w represent the world frame position, ϕ, θ, ψ represent the world frame roll-pitch-yaw angles, V_b represents the body frame velocity, and ω_b represents the body frame rotation rates, and I_m represents the motor current [11].

*5) **Off-Road Testing and Data Collection**:* Both mobile platforms were tested on uneven off-road terrains consisting of grass, dirt, and marshy lands via teleoperation. We utilized the controller for the HOUND shown in [11]. The A1 uses the default whole body Model Predictive control (MPC) controller. The A1 utilizes the identical sensors and onboard SBC found in the HOUND's backpack. An online data collection pipeline is established for both the A1 and HOUND. After collecting the data in ROS bags, we label anomalous events by replaying the bags and cross-referencing with videos to determine timestamps. Our analysis focuses on the trajectories of the robots. We ensure all sensors are fully calibrated and there is no chance of off-nominal behavior due to sensor drift or corruption.

*6) **Data Preprocessing**:* The A1 and HOUND data are sampled at frequencies of 500 Hz and 50 Hz, respectively. Due to electromagnetic interference from the A1's motors, the motor PWM signals are susceptible to noise. To mitigate this, a second-order Butterworth low-pass filter with a cut-off frequency of 50 Hz is employed for the A1's joint state values. We opted for a cut-off frequency of 50 Hz as lower thresholds would lead to unnecessary signal reduction, causing signal attenuation rather than noise attenuation. The filter is set to the same bandwidth and is uniformly applied to all input signals during the preprocessing stage, irrespective of whether individual signals necessitated filtering. This approach was intentionally adopted to ensure data consistency and streamline the preprocessing pipeline. By applying the filter universally, we maintained uniformity in data processing, which facilitated subsequent analysis and comparisons across the dataset.

B. Conditioning the data

For data preprocessing, a standardization followed by normalization procedure is used. This approach helps to both center the data around zero with a standard deviation of 1 and scale it to a specific bounded range, such as between 0 and 1. The standardization process involves subtracting the mean (μ) of each feature from the data and then dividing by the standard deviation (σ). This is represented by the equation:

$$z = \frac{x - \mu}{\sigma}$$

Where:

x : Original value

μ : Mean of the feature

σ : Standard deviation of the feature

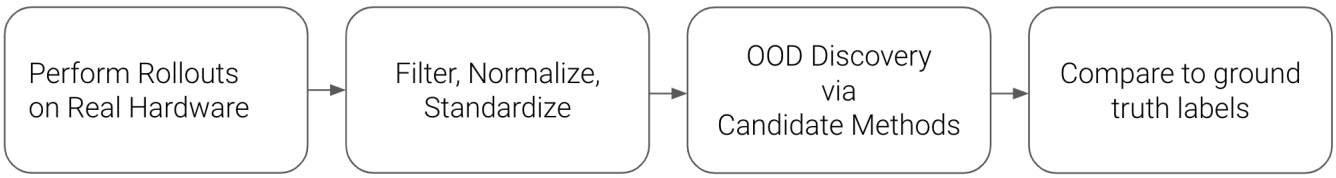

Fig. 2: Simple schematic of experimental process.

After standardization, the data is normalized using min-max scaling to a specific range. The normalization equation is given by:

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

This process ensures that the data is both centered and scaled appropriately, making it suitable for many machine learning algorithms that benefit from standardized and normalized input data.

C. Formalizing OOD Behavior

Consider an MDP $M = (S, A, T, R)$ modelling the task of locomotion of the A1 quadruped. Here, S refers to the state space, A to the action space, T is the (transition) conditional probability distribution for the next state given a pair of current state and action, and $R : (S \times A) \to \mathbb{R}$ is a real-valued reward function.

Given a policy $\pi : S \to A$ designed for this MDP and an initial state $s_0 \in S$, the execution of this policy for time steps $t \in \{0, \ldots, T\}$ can be represented as a rollout $\{(s_i, a_i, r_i)\}_{i=0}^{T}$, where $s_i \in S$ is the state visited, $a_i \in A$ is the action taken, and $r_i \in \mathbb{R}$ is the reward received at timestep i. Throughout this rollout, for each timestep i, the next state s_{i+1} has the distribution $f_T(s_{i+1} \mid s_i, a_i) = f_T(s_{i+1} \mid s_i, \pi(s_i))$, which solely depends on the current state s_i. Hence, the trajectory of this agent, given by the states $\tau = \{s_i\}_{i=0}^{T}$ can be modeled as a Markov chain with the transition distribution given by $f_{T'}(s' \mid s) = f_T(s' \mid s, \pi(s))$ for any states $s, s' \in S$.

Policies such as π are designed to provide actions for any state in S that result in high reward in the future and thus also in expected behavior. However, often in practice they provide useful actions only for a subset $C \subset S$, and their performance deteriorate outside this set, due to factors such as distribution shift between training set of a learned policy and real-life conditions, the inherent instability of certain states or the uncertainty in sensor measurements. However, this set is often not explicitly computable due to the complexity of the policy and MDP.

In this case, we can estimate the complement of C (denoted $C' = S \setminus C$) from sampled trajectories $\tau_1, \ldots, \tau_N$ by labelling time steps where off-nominal behavior that brings the agent further from the task is observed. Denoting labels for each trajectory j and timestamp i as τ_j as $y_i^{(j)} \in \{0, 1\}$, where 0 refers to nominal behavior and 1 refers to off-nominal behavior, our goal is to estimate which states of an unseen trajectory $\tau' = \{s'_i\}_{i=0}^{T}$ belong to C without further access to the underlying MDP or controller. A reasonable assumption we make for this problem is that the agent usually encounters states in C, where the nominal behavior is present. Then, any state $s' \in C'$ would also be considered out-of-distribution with respect to the "nominal" states. Specifically, we make the assumption that $\mathcal{P}(s'_i = c \mid s_0) \leq \mathcal{P}(s'_i = c' \mid s_0)$ for $c \in C$ and $c' \in C'$.

D. Working with High Dimensionality

Currently, our dataset for the A1 and HOUND includes many features including linear/angular positions, velocities, accelerations of the base and each joint. Each type of feature lead to different trajectories over a rollout, for example, joint positions follow a near-sinusoidal pattern whose phase depend on the joint, whereas pitch and roll of the base are closer to constant during nominal operation.

This high-dimensional setting poses a challenge to OOD detection, since only a subset of the features are sensitive to the perturbations that often occur, and the shift in these features might have different patterns. For a simple baseline, we have only selected the base orientation (roll, pitch, yaw values in radians) of both robots for our OOD detection pipeline, using the intuition that during near-crashes or tip-overs, base orientation changes rapidly relative to nominal operation. While all off-nominal operation is dependent on the task and the resulting expectations about the robot's behavior, we acknowledge the need for a more general method that is able to utilize the entire state, without knowledge of which features are salient. In future work, we plan to use dimensionality reduction methods such as sparse PCA (principal component analysis) and DMD (dynamic mode decomposition) for better accuracy and generalization.

III. Experiments

Working with the base orientations over time, we have tested 4 baselines: random choice, moving average threshold, moving average derivative threshold, and isolation forest. Here are the descriptions of each method:

1) **Random Choice:** For each trajectory τ_j and timestep $i \in \{0, \ldots, T_j\}$, we sample $f(\tau_j, i) \sim \text{Ber}(1/2)$ (where $f(\tau_j, i) = 1$ denotes OOD data at that timestep), where each prediction is independently and identically distributed.
2) **Moving Average Threshold:** For each trajectory τ_j, after normalizing each feature with respect to the set of

observations of that feature, we take a moving average of each feature separately with window size $w = 300$, resulting in a set of averaged states $\{m_i^{(j)}\}_{i=1}^{T_j}$. For the first $w-1$ timesteps, to avoid NaN values we instead use the average of all timesteps up to i for the signal at timestep i. Then, we estimate the OOD labels by $f(\tau_j, i) = \mathbb{1}(|m_i^{(j)}| > C_1)$ for a positive constant $C_1 = 0.6$, where $\mathbb{1}(x)$ is the indicator function.

3) **Moving Average Derivative Threshold:** For each trajectory τ_j and timestep $i \in \{0, \ldots, T_j\}$. Taking the time in seconds since the initial state as $t_i^{(j)} \in \mathbb{R}_+$, we compute the same moving average data $\{m_i^{(j)}\}_{i=0}^{T_j}$ as above, and then estimate the derivative at each timestep i by $\left\{(dm/dt)_i^{(j)}\right\}_{i=1}^{T_j}$. Then, we estimate the OOD labels by $f(\tau_j, i) = \mathbb{1}(|(dm/dt)_i^{(j)}| > C_2)$ for a positive $C_2 = 0.7$.

4) **Isolation Forest:** Moving average (and moving average derivative) methods on normalized data create a global threshold for how much the data deviates from its mean. Conversely, an isolation forest [1] starts by taking the set $\{s_i^{(j)}\}_{i=0}^{T_j} \in \tau_j$ and initializing some $M = 300$ kd-trees, where each node represents a split axis-aligned rectangular sub-region of the k-dimensional state space, and children of a node create a partition of a region. Then, each tree is extended by splitting a region into half randomly until each data point in the region of a leaf node has the same value, or when a max depth of $\lceil \log_2(T_j+1) \rceil$ is reached. The average depth of each data point is then higher when the data point is more likely to be an outlier, and a ratio $r = 0.3$ of the data points with maximum average depth are selected as outliers.

The hyperparameters w, C_1, C_2, r, M are selected manually to maximize the accuracy for two labelled rollouts π_{N+1}, π_{N+2}, one for each robot (A1 and HOUND). These rollouts are held out from the rest of the analysis, and are only used for hyperparameter search.

For each rollout and method, we estimate the OOD labels $\{y_i^{(j)}\}_{i=0}^{T_j}$ as $\{f(\tau_j, i)\}_{i=0}^{T_j}$ and then consider the accuracy, Jaccard (IoU) score and F1 score as metrics. The average metrics over the A1 and HOUND datasets are given for each method in Table I and Table II. Line plots of the features over time are given for each robot type (A1, HOUND) and data transformation (normalization, moving average, moving average derivative) below in Figure 3 and Figure 4.

TABLE I: Average Evaluation Metrics for Different OOD Detection Methods on A1

Method	Accuracy	Jaccard (IoU)	F1 Score
Random Choice	0.498	0.250	0.385
Moving Average Threshold	0.427	0.353	0.494
Moving Average Derivative Threshold	0.623	0.484	0.613
Isolation Forest	0.756	0.391	0.532

[1]We use the `scikit-learn` [12] package for the Isolation Forest algorithm.

TABLE II: Average Evaluation Metrics for Different OOD Detection Methods on HOUND

Method	Accuracy	Jaccard (IoU)	F1 Score
Random Choice	0.497	0.057	0.082
Moving Average Threshold	0.267	0.095	0.156
Moving Average Derivative Threshold	0.309	0.096	0.158
Isolation Forest	0.735	0.101	0.167

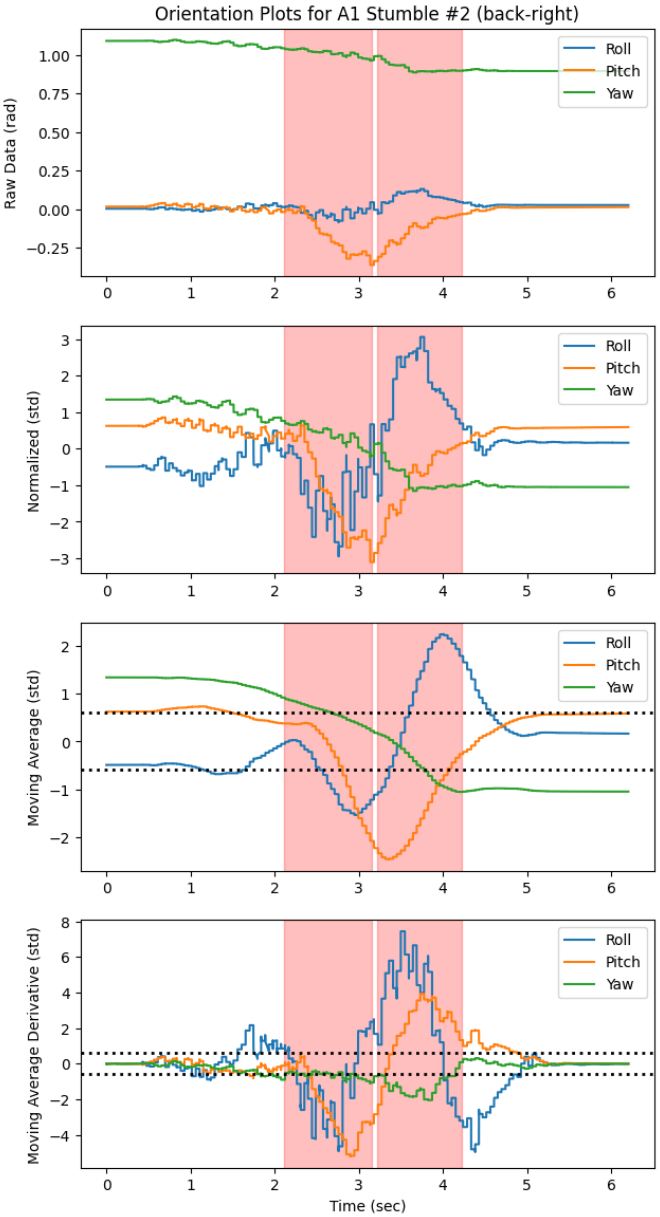

Fig. 3: The line plots of the selected features (roll, pitch, yaw) over time, with data transformations applied to detect OOD behavior.

IV. Conclusion

With respect to the A1 dataset, we see that while Isolation Method gives a higher accuracy than other methods, Moving Average Derivative Threshold gives better F1 and IoU scores. Since the OOD behavior appears less frequently than nominal

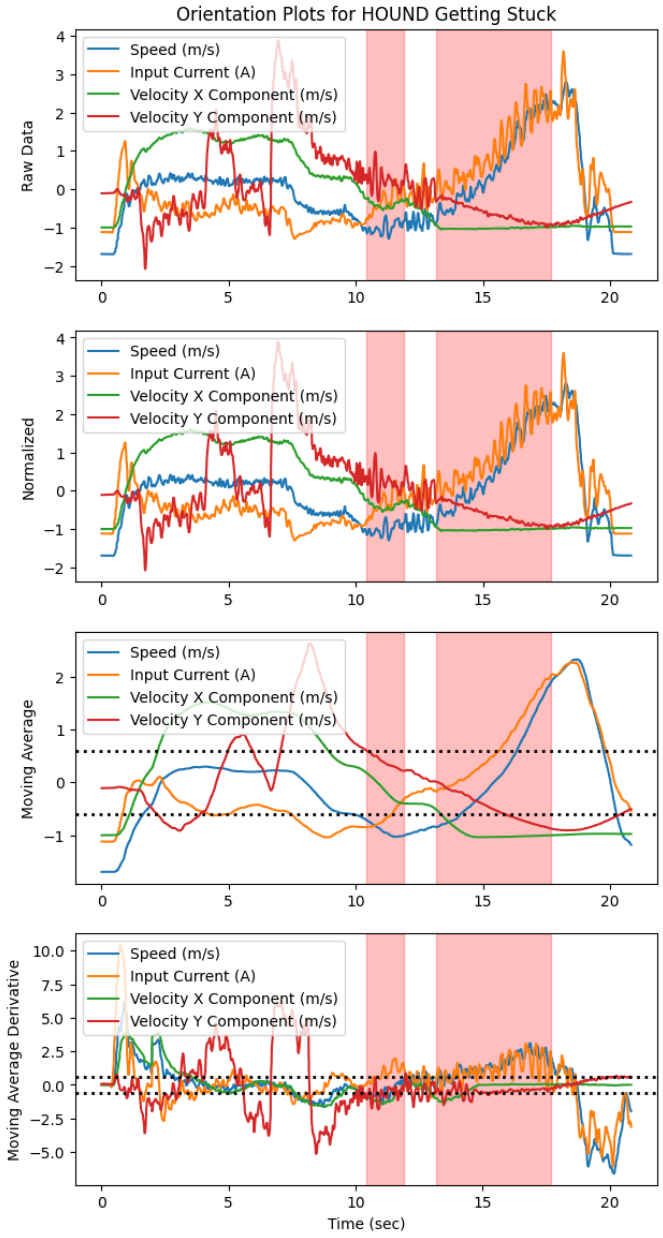

Fig. 4: The line plots of the selected features (speed, components of velocity, motor input current) over time, with data transformations applied to detect OOD behavior.

behavior in our dataset, label balance cannot be assumed in this case, and thus F1 or IoU metrics are likely more appropriate for measuring performance. While the Moving Average Derivative Threshold resulted in the highest performance in these metrics, we note that the Isolation Method has comparable performance across A1 datasets. Compared to the Random Choice, which is our naive baseline, both methods provide a significant improvement in all metrics.

However, with respect to the HOUND dataset, we see that the Isolation Forest method results in the highest performance in all 3 metrics (accuracy, IoU and F1 score), while the IoU and F1 metrics of Moving Average Derivative Threshold are similar as well. Both methods still provide better F1 and IoU scores than Random Choice, which indicates that the off-nominal behavior observed during a real-life rollout correlates with out-of-distribution states in the corresponding trajectories. For this reason, we consider both methods strong baselines, especially compared to Random Choice.

We will continue this work by comparing more sophisticated unsupervised methods such as dynamics mode decomposition to obtain results that minimize the use of heuristics, such as manual dimensionality reduction or threshold tuning. We will also expand our dataset by an order of magnitude so that we can better train data-intensive models and test them against our baselines. While our current dataset only considers a small subset of the possible out-of-distribution states, it will be extended to include different controllers and off-nominal behaviors as well, enabling better out-of-distribution detection and generalization. Finally, we will consider evaluating our methods on externally collected datasets, especially as more are released in the coming months. Our ultimate goal is to develop and validate a method that can be used on arbitrary off-road autonomy datasets to uncover the truly informative and interesting underlying OOD moments embedded deep in the data.

References

[1] D. Soudbakhsh, A. M. Annaswamy, Y. Wang, S. L. Brunton, J. Gaudio, H. Hussain, D. Vrabie, J. Drgona, and D. Filev, "Data-driven control: Theory and applications," in *2023 American Control Conference (ACC)*, 2023, pp. 1922–1939.

[2] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[3] E. Candela, O. Doustaly, L. Parada, F. Feng, Y. Demiris, and P. Angeloudis, "Risk-aware controller for autonomous vehicles using model-based collision prediction and reinforcement learning," *Artificial Intelligence*, vol. 320, p. 103923, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370223000693

[4] R. Scalise, E. Caglar, B. Boots, and C. C. Kessens, "Self-righting and recovery in the wild: Challenges and benchmarks," *ICRA*, 2024.

[5] C. Choi, Y. Lee, A. Chen, A. Zhou, A. Raghunathan, and C. Finn, "Autoft: Robust fine-tuning by optimizing hyperparameters on ood data," *arXiv preprint arXiv:2401.10220*, 2024.

[6] S. Sagawa, P. W. Koh, T. Lee, I. Gao, S. M. Xie, K. Shen, A. Kumar, W. Hu, M. Yasunaga, H. Marklund, S. Beery, E. David, I. Stavness, W. Guo, J. Leskovec, K. Saenko, T. Hashimoto, S. Levine, C. Finn, and P. Liang, "Extending the wilds benchmark for unsupervised adaptation," in *International Conference on Learning Representations (ICLR*, 2022.

[7] J.-C. Gagnon-Audet, K. Ahuja, M.-J. Darvishi-Bayazi, P. Mousavi, G. Dumas, and I. Rish, "Woods: Benchmarks for out-of-distribution generalization in time series," *arXiv preprint arXiv:2203.09978*, 2022.

[8] M. Foutter, R. Sinha, S. Banerjee, and M. Pavone, "Self-supervised model generalization using out-of-distribution detection," in *First Workshop on Out-of-Distribution Generalization in Robotics at CoRL 2023*, 2023. [Online]. Available: https://openreview.net/forum?id=z5XS3BY13J

[9] S. Ancha, P. R. Osteen, and N. Roy, "Deep evidential uncertainty estimation for semantic segmentation under out-of-distribution obstacles."

[10] S. S. Srinivasa, P. Lancaster, J. Michalove, M. Schmittle, C. Summers, M. Rockett, R. Scalise, J. R. Smith, S. Choudhury, C. Mavrogiannis, and F. Sadeghi, "Mushr: A low-cost, open-source robotic racecar for education and research," 2023.

[11] S. Talia, M. Schmittle, A. Lambert, A. Spitzer, C. Mavrogiannis, and S. S. Srinivasa, "Hound: An open-source, low-cost research platform for high-speed off-road underactuated nonholonomic driving," 2023.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.