Online Latent Factor Representation Learning

Alejandro Murillo-González and Lantao Liu
Indiana University–Bloomington
{almuri, lantao}@iu.edu

Abstract*—Autonomous robots operating in complex, unstructured environments face significant challenges due to latent, unobserved factors that obscure their understanding of both their internal state and the external world. Addressing this challenge would enable robots to develop a more profound grasp of their operational context. To tackle this, we propose a novel framework for online learning of hidden state representations, with which the robots can adapt in real-time to uncertain and dynamic conditions that would otherwise be ambiguous and result in suboptimal or erroneous behaviors. Our approach is formalized as a Generalized Hidden Parameter Markov Decision Process, which explicitly models the influence of unobserved parameters on both transition dynamics and reward structures. Our core innovation lies in learning online the joint distribution of state transitions, which serves as an expressive representation of latent ego- and environmental-factors. This probabilistic approach supports the identification and adaptation to different operational situations, improving robustness and safety. Through a multivariate extension of Bayesian Online Changepoint Detection, our method segments changes in the underlying data generating process governing the robot's dynamics. The robot's transition model is then informed with a symbolic representation of the current situation derived from the joint distribution of latest state transitions, enabling adaptive and context-aware decision-making. To showcase the real-world effectiveness, we validate our approach in the challenging task of unstructured terrain navigation, where unmodeled and unmeasured terrain characteristics can significantly impact the robot's motion. Our experiments reveal significant improvements in data efficiency, policy performance, and the emergence of safer, adaptive navigation strategies. Accompanying video: https://youtu.be/VKR18WaSCAk.

This manuscript summarizes our work on online and unsupervised latent factor representation learning. For detailed proofs, additional results and in-depth analysis, refer to: [36].

I. Introduction

Reliable robot deployment in unstructured and dynamic environments, requires systems that adapt to unforeseen challenges and operate effectively under uncertainty. Consider, for example, an Unmanned Ground Vehicle (UGV) navigating steep and rugged terrain. Such a robot may become immobilized due to unexpected factors, ranging from unmodeled terrain properties to variations in its own internal dynamics. These factors are inherently difficult, if not impossible, to anticipate exhaustively [10, 28, 17, 50, 56, 55]. This challenge is further exacerbated by the impracticality of equipping robots with every conceivable sensor and analysis algorithm needed to fully capture the world's and robot's full state in all scenarios. Consequently, robots must learn to operate effectively using only partial information derived from their observable state—the limited set of measurements and estimates available in real time. This raises the need for adaptive mechanisms

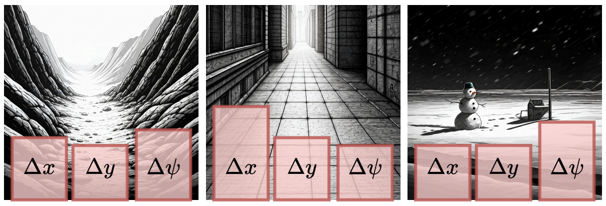

Figure 1: Motivating Example. When the robot has incomplete state information, the same action, $\mathbf{a}$, can yield significantly different outcomes. In this example, unmodeled terrain types—factors absent from the state and transition model—affect an UGV's response, resulting in distinct changes in position and heading (Δx, Δy, and $\Delta \psi$, respectively) for the same control action (linear and angular velocity). Environment images generated with Gemini AI.

capable of inferring and representing the hidden aspects of the robot and its environment to inform decision-making and enhance resilience in uncertain conditions [34, 13, 25, 23].

Furthermore, when a robot relies on partial information to operate on a complex environment, the risk of ambiguous scenario representation becomes significant [35, 41, 5]. Such ambiguity is undesirable, as it increases uncertainty during decision-making, often leading to suboptimal plans and, in some cases, unsafe or hazardous behaviors. This challenge is exemplified in Figure 1: a UGV with incomplete state information—lacking a (robust) scene interpretation module to detect the terrain type it is traversing—executes an action from what appears to be the same state. However, the outcomes of these actions differ drastically, revealing a discrepancy that cannot be attributed solely to aleatoric errors (those arising from process or observation noise). This highlights the critical need for robust methods to infer and resolve hidden state ambiguities to ensure reliable robot operation.

To address this, we propose an online, unsupervised framework for learning compact representations of the unmodeled and unobserved latent factors that give rise to such ambiguities. Our method balances representation capacity, data efficiency, and minimal inductive bias by learning, in real time, the joint distribution of state transitions—termed the robot's "*situation*"—which captures both its internal state and the prevailing environmental conditions. By extending Bayesian Online Changepoint Detection (BOCD) to the multivariate case, the robot continuously models its current situation, detects abrupt changes, and adapts its behavior accordingly, all without privileged information or multi-stage training, in contrast to recent approaches [29, 26, 27, 42, 32]. Leveraging only proprioceptive sensing, this approach yields an expressive probabilistic representation of the underlying data-generating process (UDGP), enabling reliable task execution in uncertain, unstructured environments.

Our main contributions are threefold. First, we present an efficient method for online robot adaptation based on identified latent factors, via a *multivariate* extension of BOCD to estimate the transition distribution in real time. Second, we demonstrate how these situation distributions can be mapped to symbolic representations, granting the dynamics model access to relevant context for adaptation. Finally, we validate our framework on a UGV navigating unstructured terrain, showing that our situationally-aware (SA) dynamics model, trained via Model-Based Reinforcement Learning (MBRL), yields superior, data-efficient policies and exhibits emergent adaptive behaviors that improve navigation safety and effectiveness.

II. Related Work

We proceed to cover policy and dynamics learning with learned auxiliary representations as well as unstructured terrain navigation. An extended version is available at: [36].

RL with Learned Auxiliary Representations. In *model-free RL*, approaches like Lee et al. [29] and Rapid Motor Adaptation (RMA) [26, 27] utilize temporal convolutional networks or environmental factor encoders to infer latent representations from proprioceptive history or privileged information. These representations are then used to inform control policies, often requiring multi-stage training processes and sometimes relying on a teacher policy or an adaptive curriculum. Follow-up works, such as [42, 32], extend this concept to in-hand object rotation and manipulation using depth perception. In contrast to these methods, our approach directly learns representations from proprioception without needing privileged information or multi-stage training, allowing for immediate symbol detection and real-world deployment without further fine-tuning. Within *model-based RL*, researchers address challenges like high-dimensional state spaces and compounding error. Havens et al. [20] propose compressing the state space to avoid learning unimportant features, while [46, 47] learn "predictable behaviors" and corresponding dynamics to facilitate model-based planning in real-world scenarios. Lee et al. [30] introduced Context-aware Dynamics Models (CaDM) that learn a context encoder and multiple dynamics models conditioned on a latent vector encoding dynamics-specific information. Our method also falls under the model-based paradigm, but it differs by employing a single dynamics model that adapts to all contexts, rather than requiring multiple specialized models. Furthermore, our structured symbolic representation explicitly captures the UDGP of the dynamics, offering a more interpretable alternative to unstructured latent representations.

Unstructured Terrain Navigation. We validate our approach through the challenging task of unstructured terrain navigation, where unmodeled terrain characteristics significantly impact robot motion. This field is typically addressed from two perspectives: *adaptation* and *path planning*. Adaptive methods, such as those by Xu et al. [54] and Wang et al. [52], focus on enabling robots to recover from unexpected scenarios and manage uncertainty. Terrain-aware approaches like [49, 50] learn control offsets for consistent navigation. Conversely, path planning methods [57, 21, 31] analyze terrain using robot sensors to find safer trajectories, sometimes with a focus on risk-awareness [48, 3, 4]. Learning-based methods also consider ground-robot interaction and surface data [44]. Our method intersects both perspectives by developing dynamics models more robust to terrain challenges, which are then used for local motion planning, enabling safe traversal under partial state information.

III. Preliminaries

A. Generalized Hidden Parameter Markov Decision Process

Perez et al. [39] present GHP-MDPs to account for unobserved, latent parameters which influence the dynamics and reward function. These latent factors are not directly observable by the agent, but the agent must infer them through interaction with the environment to optimize its policy.

Definition 1. *A Generalized Hidden Parameter MDP (GHP-MDP) is defined by a tuple $\mathcal{M}_{GHP} = (S, A, \Theta, T, R, \gamma)$, where S is the set of observable states and A is the set of actions available to the agent. Θ is the set of latent variables that are not observable by the agent but influence the system's dynamics and rewards. $T : S \times \Theta \times A \times S \to [0, 1]$ is the transition function which gives the probability of transitioning from state s to state s' under action a and hidden parameter θ. $R : S \times \Theta \times A \to \mathbb{R}$ is the reward function. And the discount factor $\gamma \in [0, 1]$ modulates the agent's desire for immediate and future rewards.*

B. Bayesian Online Changepoint Detection (BOCD)

The problem of changepoint detection is concerned with determining the point where the observed data distribution changes in an ordered set of measurements, such as in time-series [12]. BOCD was introduced by Adams and MacKay [1] to tackle this problem by framing it as an estimation of the posterior distribution of the current "run length" r_t, meaning how likely it is that the measurement at time-step t belongs to the same data generating process that started r_t timesteps ago; while also obtaining the parameters $\boldsymbol{\eta}$ defining such process.

Intuitively, BOCD continuously monitors a data stream to identify points where the underlying data distribution changes. It starts with a prior belief about where changes might occur and updates this belief by evaluating how well the current data fits different scenarios of changepoints (segment/run lengths). At each time step, the algorithm uses the new data to adjust the likelihood of potential changepoints, incorporating prior knowledge and observed evidence. The method recursively updates this distribution to assess the existence of changes in the UDGP of the current segment.

Formally, the method assumes that the sequence of observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$ can be segmented into non-overlapping intervals, where a changepoint corresponds to the point $\boldsymbol{x}_i$ marking the transition between two adjacent intervals. The set

$\boldsymbol{x}_t^{(r)}$ contains the points estimated to belong to run r_t. Furthermore, the data within each interval ρ is *i.i.d.* from $P(\boldsymbol{x}_t|\boldsymbol{\eta}_\rho)$, and the parameters $\boldsymbol{\eta}_\rho, \rho = 1, 2, \ldots$ are also *i.i.d.* Note that the last assumption is about the *parameters* of the underlying distributions describing the data generating process of each segment. Finally, we should be able to compute the predictive distribution $P(\boldsymbol{x}_{t+1}|r_t, \boldsymbol{x}_t^{(r)})$ and define a conditional prior on the changepoint $P(r_t|r_{t-1})$. See Appendix A for more details.

IV. Method

We model online the changes in the distribution governing the robot's transition dynamics. These changes encapsulate the influence of latent, unobserved factors—such as unmeasured variables or emergent phenomena—that are challenging to predict or exhaustively anticipate. With the learned representation the robot refines its transition model, improving downstream performance. Due to space considerations, refer to [36] for in-depth descriptions and proofs.

A. Online Transition Distribution Modeling

The transition distribution $T(s, a, s')$ in a MDP defines a conditional probability distribution that specifies the likelihood of transitioning to a new state s' given the current state s and action a. That is, $T(s, a, s') = P(s' \mid s, a)$. In particular, $P(s' \mid s, a) \propto \Pr(S_t = s, A_t = a, S_{t+1} = s')$. However, as previously noted, in real-world applications, estimates of $P(s' \mid s, a)$ usually fail to accurately capture the true dynamics of the world. Generally, this shortfall arises from latent, unobserved factors—such as unmeasured variables or phenomena for which the robot lacks the necessary sensors—that significantly impact state transitions. Thus, *our key insight* is that by modeling online the joint transition distribution $\Pr(S_t = s, A_t = a, S_{t+1} = s')$ that best explains the UDGP governing the current state transitions, we can account for the latent factors that influence the world dynamics and consequently improve performance on downstream tasks.

To achieve this, we break the problem of online transition distribution modeling into determining local joint transition distributions. By "local", we mean around the latest set or trajectory of states experienced by the agent. For example, an UGV stuck in a slippery area during hill climbing will estimate a different local joint transition distribution than the same robot when it is moving over a flat and smooth sidewalk.

We represent the local dynamics T_L of the robot via a multivariate Normal distribution parameterized by its mean vector $\boldsymbol{\mu}$ and precision matrix Λ (inverse of the covariance):

$$T_L(\boldsymbol{x}|\boldsymbol{\mu}, \Lambda) = \frac{|\Lambda|^{1/2}}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Lambda (\boldsymbol{x}-\boldsymbol{\mu})\right). \quad (1)$$

By focusing on the transitions within a neighborhood of states, the multivariate Normal distribution strikes a balance between representational power and computational efficiency.

Specifically, we use Eq. (1) to model the *joint distribution* of the *(state, action, next state)*-tuples, which we call the ***situation*** of the robot from now on. Thus, $\boldsymbol{x} = [s_t; a_t; s_{t+1}]^\top \in \mathbb{R}^{d_x + d_a + d_x}$, which corresponds to concatenated state $s_t \in \mathbb{R}^{d_x}$, action $a_t \in \mathbb{R}^{d_a}$ and next state $s_{t+1} \in \mathbb{R}^{d_x}$ vectors. This will describe the situation of the robot, as we will be able to distinguish via the *next state* components of T_L whenever the same action under similar conditions (states) leads to different outcomes. Since T_L encodes the effect of the unknown factor, it can inform T and adapt to the detected dynamics change.

However, to effectively model the local dynamics using Eq. (1), we need to estimate $\boldsymbol{\eta} = \{\boldsymbol{\mu}, \Lambda\}$ online and simultaneously detect when a different $\boldsymbol{\eta}'$ describes the current situation of the robot better. For this, we extend BOCD to a *multivariate* setting and apply it for online and unsupervised discovery of the multiple sets $\boldsymbol{\eta}_\rho$, as shown next.

1) Multivariate BOCD: Assume a process with measurements arriving one at a time. For example, a robot's state estimate. Each of these incoming data points can be considered as an observation from a statistical distribution. We introduce a *multivariate* extension of BOCD to detect and model online the UDGP of the incoming data points. Formally, the data points are assumed to be drawn from a multivariate normal distribution characterized by a set of parameters $\boldsymbol{\eta} = \{\boldsymbol{\mu}, \Lambda\}$. As a result, at time $t = N$, we have collected a set of observations $\mathcal{D} = \{\boldsymbol{x}_i : \boldsymbol{x}_i \overset{\text{iid}}{\sim} \mathcal{N}(\boldsymbol{\eta})\}_{i=1}^N$. This means:

$$p(\mathcal{D}|\boldsymbol{\mu}, \Lambda) = \prod_{\boldsymbol{x}_i \in \mathcal{D}} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}, \Lambda) \quad (2)$$
$$= \left(\frac{|\Lambda|}{(2\pi)^d}\right)^{n/2} \cdot \exp\left(-\frac{1}{2}\sum_{i=1}^n (\boldsymbol{x}_i - \boldsymbol{\mu})^\top \Lambda (\boldsymbol{x}_i - \boldsymbol{\mu})\right).$$

We are interested in detecting the time $t' > t$ when the incoming data no longer comes from $\mathcal{N}(\boldsymbol{\eta})$ and determine the new values $\boldsymbol{\eta}'$ that describe the new UDGP.

Since we want to understand when the parameters $\boldsymbol{\eta}$ of the UDGP have changed, we need to define a prior for them. For this, we propose the *Normal-Wishart distribution*. This conjugate prior is particularly useful because it allows for closed form belief updates for $\boldsymbol{\mu} \in \mathbb{R}^d$ and $\Lambda \in \mathbb{R}^{d \times d}$ as more data becomes available. It is defined as follows [37]:

$$p(\boldsymbol{\mu}, \Lambda) = \frac{1}{Z}|\Lambda|^{1/2} \exp\left(-\frac{\kappa}{2}(\boldsymbol{\mu} - \mu_0)^T \Lambda (\boldsymbol{\mu} - \mu_0)\right)$$
$$\cdot |\Lambda|^{(\kappa - d - 1)/2} \exp\left(-\frac{1}{2}\operatorname{tr}(T^{-1}\Lambda)\right), \quad (3)$$

where $Z = \left(\frac{\kappa}{2\pi}\right)^{d/2} |T|^{\kappa/2} 2^{d\kappa/2} \Gamma_d(\kappa/2)$ is a normalizing factor and $\operatorname{tr}(\cdot)$ denotes the trace of a matrix.

Essentially, the Normal–Wishart prior places a Normal prior with mean $\boldsymbol{\mu}_0$ and scale $\kappa_0^{-1}\Lambda^{-1}$ on $\boldsymbol{\mu}$ and a Wishart prior with scale matrix T and ν_0 degrees of freedom on the precision Λ. Its conjugacy and parameters (κ_0, ν_0) yield flexible, assumption-light modeling of multivariate normals [9, 37].

With the sampling model Eq. (2) and prior Eq. (3) determined, we proceed to make the connection to BOCD.

Specifically, we will determine the closed form solutions for the growth probability Eq. (17) and changepoint probability Eq. (18), from which we can recover the run length probability Eq. (19). We assume that the prior on the changepoint probability is $P_{\texttt{gap}}(g) \sim \texttt{geometric}(\lambda)$. This makes the process memoryless and the hazard function constant at $H(\tau) = 1/\lambda$, where λ is the expected time between changepoints [1].

Lemma 1 (Growth and Changepoint Probabilities). *Let $\boldsymbol{x}_t$ be the observation received at time t. Let $\boldsymbol{\eta}^{(r)} = \{\boldsymbol{\mu}_i^{(r)}, T_i^{(r)}, \nu_i^{(r)}, \kappa_i^{(r)}\}_{i=t_{cp}}^{t}$ be the parameters of the UDGP (r) that started at time $t_{cp} < t$. Then,*

- *The **Growth Probability**, or probability that we stayed in the same UDGP, at time t is:*

$$P(r_t = r_{t-1}+1, \boldsymbol{x}_{t_{cp}:t}) = \frac{\lambda - 1}{\lambda} P(r_{t-1}, \boldsymbol{x}_{t_{cp}:t-1}) \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_{t-1}^{(r)}, \Lambda_{t-1}^{(r)}). \tag{4}$$

- *The **Changepoint Probability**, or probability that we moved to a different UDGP, at time t is:*

$$P(r_t = 0, \boldsymbol{x}_{t_{cp}:t}) = \frac{1}{\lambda} \sum_{\tau=0}^{r_{t-1}} P(\tau, \boldsymbol{x}_{t_{cp}:t-1}) \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_{\tau}^{(r)}, \Lambda_{\tau}^{(r)}). \qquad \square \tag{5}$$

Subsequently, as new observations arrive, we need to update the run length distribution to determine whether or not the UDGP has changed. Next, we describe in Lemma 2 the efficient and optimal closed-form solution.

Lemma 2 (Online Distribution Parameters Learning). *Given the ordered set of situation parameters $\boldsymbol{\eta}^{(r)}$ and a new observation $\boldsymbol{x}_t$, the posterior parameters $\boldsymbol{\mu}_t^{(r)}, T_t^{(r)}, \nu_t^{(r)}, \kappa_t^{(r)}$ for the current UDGP are:*

$$\boldsymbol{\mu}_t^{(r)} = \frac{\kappa_{t-1}^{(r)} \boldsymbol{\mu}_{t-1}^{(r)} + \boldsymbol{x}_t}{\kappa_{t-1}^{(r)} + 1}; \tag{6}$$

$$T_t^{(r)} = T_{t-1}^{(r)} + S + \frac{\kappa_{t-1}^{(r)}}{\kappa_{t-1}^{(r)} + 1} \Delta; \tag{7}$$

$$\nu_t^{(r)} = \nu_{t-1}^{(r)} + 1; \tag{8}$$

$$\kappa_t^{(r)} = \kappa_{t-1}^{(r)} + 1; \tag{9}$$

where $S = \sum_{i=1}^{t-1} (\boldsymbol{x}_i - \boldsymbol{x}_t)(\boldsymbol{x}_i - \boldsymbol{x}_t)^\top$ and $\Delta = (\boldsymbol{\mu}_{t-1}^{(r)} - \boldsymbol{x}_t)(\boldsymbol{\mu}_{t-1}^{(r)} - \boldsymbol{x}_t)^\top$. $\square$

2) Online Situation Modeling: To determine when T_L changes we use our multivariate BOCD. Algorithm 1 describes the process: we first determine if we changed situations (Lemma 1) and then model the current situation (Lemma 2).

Algorithm 1 Online Situation Modeling (`SM`)

```
1: Get $\boldsymbol{x}_t = [s_{t-1}; a_{t-1}; s_t]^\top$
2: Get the *Growth Probability*, $j$ = *Eq.* (4)
3: Get the *Changepoint Probability*, $k$ = *Eq.* (5)
4: *Update* parameters $\boldsymbol{\eta}_{t-1}$ according to *Eqs.* (6) – (9)
5: situation_change = $\mathbb{1}(k > j)$
6: *return* situation_change, $\boldsymbol{\eta}_t$
```

B. *Online Situation Identification*

To estimate T_L (Eq. 1) online we use the Maximum Likelihood Estimate (MLE) of T_L on the current run data $\boldsymbol{x}_t^{(r)}$. The estimated parameters are added to a set $\boldsymbol{\Psi}$ of identified situations. To determine the varying window size of the sequence of past state measurements that make $\boldsymbol{x}_t^{(r)}$ we add incoming state estimates to $\boldsymbol{x}_t^{(r)}$ until a new changepoint resets it to the empty set. The Mahalanobis distance [33] let us identify which situation $\psi \in \boldsymbol{\Psi}$ the robot is experiencing:

$$M(\psi) := \left(\frac{1}{|\, \boldsymbol{x}_t^{(r)} \,|} \sum_{\boldsymbol{x}_i \in \boldsymbol{x}_t^{(r)}} \sqrt{(\mathbf{x}_i - \boldsymbol{\mu}_\psi)^\top \Lambda_\psi (\mathbf{x}_i - \boldsymbol{\mu}_\psi)} \right), \tag{10}$$

where $\psi = (\boldsymbol{\mu}_\psi, \Lambda_\psi)$. In this way we can estimate how far each point in the current run data $\boldsymbol{x}_t^{(r)}$ is to the distributions describing the detected local transition models T_L. Then, using a threshold τ we determine if the distribution of the closest situation is appropriate to represent the current local dynamics of the robot. If it is not, we create a new situation $\psi = (\boldsymbol{\mu}, \Lambda)$ that we add to $\boldsymbol{\Psi}$ during training. At test time, we use the closests experienced situation in $\boldsymbol{\Psi}$ to the one the robot is currently experiencing. Algorithm 2 describes this.

Algorithm 2 Online Situation Identification (`SI`)

```
1: Get $\boldsymbol{x}_t = [s_{t-1}; a_{t-1}; s_t]^\top$
2: Get data from current run $\boldsymbol{x}_t^{(r)}$
3: situation_change, $\boldsymbol{\eta}_t \leftarrow$ SM($\boldsymbol{x}_t, \boldsymbol{x}_t^{(r)}, \boldsymbol{\eta}_{t-1}$)        ▷ Alg. 1
4: $\psi^* \leftarrow \arg\min_{\psi \in \boldsymbol{\Psi}} M(\psi)$        ▷ $M(\psi)$ = Eq. (10)
5: **if** situation_change **then**
6:     **if** $M(\boldsymbol{\mu}_{\psi^*}, \Lambda_{\psi^*}) \geq \tau$ and training **then**
7:         Add $(\boldsymbol{\mu}, \Lambda) \leftarrow$ MLE($\boldsymbol{x}_t^{(r)}$) to $\boldsymbol{\Psi}$        ▷ Create Situation
8:         $\psi^* \leftarrow (\boldsymbol{\mu}, \Lambda)$
9:     **end if**
10:    $\boldsymbol{x}_t^{(r)} \leftarrow \emptyset$        ▷ Variable-sized window of past states
11: **else**
12:    Add $\boldsymbol{x}_t$ to $\boldsymbol{x}_t^{(r)}$
13: **end if**
14: *return* $\psi^*$
```

C. *Situationally-Aware Dynamics*

We proceed to leverage the modeled situations to refine the transition function T. As discussed before, we argue that for effective unstructured terrain planning and control we need to account for the hidden state. We do this by representing the hidden latent factors via a parameter $\theta \in \Theta$ that informs T about the unmodeled world characteristics that also affect the prediction. In such cases, we have a GHP-MDP with transition:

$$T(s, \theta, a, s') = \Pr(S_{t+1} = s' \mid S_t = s, \Theta_t = \theta, A_t = a), \tag{11}$$

where S_t, Θ_t, A_t and S_{t+1} are random variables representing the state of the world, the current representation of the latent factors and the actions taken by the agent. We model the right-hand side of Eq. (11) as an ensemble $\mathcal{M}_m : S \times \Theta \times A \rightarrow S$ of m probabilistic (Gaussian) neural networks. This architecture choice estimates the predictive distribution over the next state and considers the epistemic and aleatoric uncertainty [6].

1) Representing the Local Transition Model: To provide $\mathcal{M}_m$ the current situation information we need to map each $\psi \in \boldsymbol{\Psi}$ to a symbol $\theta \in \Theta$. For this, we define the function

$\mathbf{\Theta} : \mathbf{\Psi} \rightarrow \Theta$. In our case, we need $\mathbf{\Theta}$ to distinguish between multiple instances of the local distribution T_L (Eq. 1), all of which belong to a family of distributions that share a common form but vary in their parameters.

We find that the Moment Generating Function (MGF) provides an useful tool for this purpose, as it uniquely characterizes a distribution [8, Theorem 1]. In general, for a multivariate Gaussian random vector $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, the MGF is defined $M_{\boldsymbol{X}}(\boldsymbol{t}) = \mathbb{E}[\exp(\boldsymbol{t}^\top \boldsymbol{X})] = \exp(\boldsymbol{t}^\top \boldsymbol{\mu} + \frac{1}{2}\boldsymbol{t}^\top \Sigma \boldsymbol{t})$. Thus, to symbolically represent the current situation ψ we use its MGF, with a fixed, nonzero $\boldsymbol{t} \in \mathbb{R}^d$, as follows:

$$\mathbf{\Theta}_{\boldsymbol{t}}(\psi) = \log M_{\boldsymbol{X}}(\boldsymbol{t}; \psi) = \boldsymbol{t}^\top \boldsymbol{\mu}_\psi + \frac{1}{2}\boldsymbol{t}^\top \Lambda_\psi^{-1} \boldsymbol{t}, \qquad (12)$$

where we apply the log function for numerical convenience. Although it could happen that for a given $\boldsymbol{t}$ the representations of two situations would end up being equal, this is highly unlikely and we did not experience any such collisions. Then, whenever $\mathcal{M}_m$ is being called, we augment the robot's state with the current situation's symbol θ. Note that the current situation is constantly updated as the output from Algorithm 2, which is executed every time a new observation $\boldsymbol{x}_t$ is obtained.

2) Dynamics Learning: The model $\mathcal{M}_m$ is learned by alternating exploration and exploitation, as is common in the MBRL framework. Throughout training we append the set of tuples $\{(s_i, \theta_i, a_i, s_{i+1})\}_{i=1}^{N_e}$, where N_e is the number of steps in the episode, to the replay buffer $\mathcal{R}$. Note that different from traditional MBRL, we include the representation θ_i of the current situation. The collected data is then used to train the dynamics model $\mathcal{M}_m$. In subsequent episodes, the agent uses the updated model to make better informed decisions.

Given that the period of time the robot experiences in each situation is different, the replay buffer $\mathcal{R}$ will naturally have an unbalanced sample of situations. Thus, to ensure that the dynamics $\mathcal{M}_m$ are learned appropriately for all situations $\psi \in \mathbf{\Psi}$, we introduce a loss weighted by the cardinality of each situation. Specifically, we want to minimize the Negative Log-Likelihood (NLL):

$$\mathcal{L}(\theta) = -\sum_{i=1}^{|\mathcal{R}|} w(\theta_i)\left(\log \hat{\sigma}_i^2 \|\hat{\mu}_i - y_i\|_2^2 + \log \hat{\sigma}_i^2\right), \qquad (13)$$

where $\hat{\mu}_i$ and $\hat{\sigma}_i^2$ are the predicted mean and variance, and y_i is the measured outcome, and each prediction is weighted by $w(\theta_i) = |\mathcal{R}| / \sum_{j=1}^{|\mathcal{R}|} \mathbb{I}(\theta_j = \theta_i)$, where $\mathbb{I}(\theta_j = \theta_i)$ is an indicator function equal to 1 if $\theta_j = \theta_i$ and 0 otherwise. This frequency-weighted loss enhances the model's focus on rare values of θ_i, encouraging balanced learning across all samples.

3) Planning and Control: We use our SA dynamics model $\mathcal{M}_m$ together with the Model Predictive Path Integral (MPPI) method [16]. The expected return is computed based on the predicted future states while assuming that the situation symbol θ remains constant throughout the roll-outs. To select an action, MPPI finds every timestep a solution to:

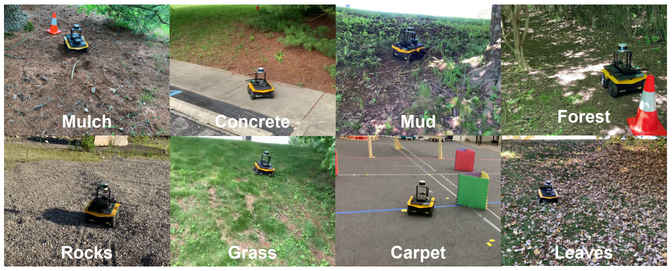

Figure 2: Using the learned SA Dynamics Model the robot can successfully traverse multiple unstructured terrains.

$$a^\star_{t:t+H} = \arg\min_{a_{t:t+H}} \sum_{h=0}^{H-1} \left(\|\hat{s}_{t+h} - s_{\text{goal}}\|_2 + \rho \mathcal{B}(\hat{s}_{t+h})\right), \qquad (14)$$

where $\hat{s}_{t+h}$ denotes the predicted state at time $t+h$, s_{goal} is the goal state, $\rho > 0$ is a weight hyperparameter and $\mathcal{B}(\hat{s}_{t+i})$ is a log-barrier function for safe navigation around obstacles (s_{obs}): $\mathcal{B}(\hat{s}_{t+i}) = -\log \|\hat{s}_{t+h} - s_{obs}\|_2$.

V. Results

We proceed to evaluate two questions: **(Q1)** Does SA enable effective online hidden state representation learning? and **(Q2)** Can we use SA for faster dynamics learning?.

To evaluate latent factor identification, the observation space is limited to proprioceptive odometry, excluding explicit environmental cues such as terrain or friction. These factors must be implicitly encoded in the hidden state, enabling the model to adapt to varying conditions without external sensory input. More results are available in Murillo-Gonzalez and Liu [36].

Experimental Setup: The state space $s \in \mathbb{R}^{12}$ corresponds to $s = \begin{bmatrix} \mathbf{p} & \mathbf{e} & \mathbf{v} & \mathbf{\Omega} \end{bmatrix}^\top$, where $\mathbf{p} = [x, y, z] \in \mathbb{R}^3$ is the robot's pose, $\mathbf{e} = [\phi, \zeta, \gamma] \in \mathbb{R}^3$ are the roll, pitch, and yaw angles (orientation), $\mathbf{v} = [v_x, v_y, v_z] \in \mathbb{R}^3$ are the linear velocity components, and $\mathbf{\Omega} = [\omega_x, \omega_y, \omega_z] \in \mathbb{R}^3$ are the angular velocity components. The action space $a = \begin{bmatrix} v_{\text{cmd}} & \omega_{\text{cmd}} \end{bmatrix}^\top$, where $v_{\text{cmd}} \in [-1, 1]$ and $\omega_{\text{cmd}} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ includes the commanded linear and angular velocities, respectively.

The *situation* space $\mathbf{x} \in \mathbb{R}^{26}$ corresponds to the concatenation of state and action vectors $\mathbf{x}_t = \begin{bmatrix} s_{t-1}; & a_{t-1}; & s_t \end{bmatrix}^\top$.

For more details refer to Appendix B.

Evaluation Environments: We validate our approach on unstructured terrains shown in Figure 2, where latent surface features influence the robot's motion. We use Clearpath's Inspection World [7] in Gazebo [24] for simulation.

Baselines: To highlight the benefits of SA, we compare our approach against several *learning-based* and *physics-based* baselines. These include **CaDM** [30], a context-aware dynamics model similar to ours, and **PE+LSTM**, which uses an LSTM for parallel hidden state recovery. We also benchmark leading MBRL methods like **TD-MPC2** [19] and **PETS** [6], along with **CDL-CMI** [53], a causally-inspired dynamics model. For a comprehensive comparison, we include policies trained with model-free methods (**PPO** [45], **SAC** [18], **TD3** [15]) and a classical **Dubins** physics-based model [11].

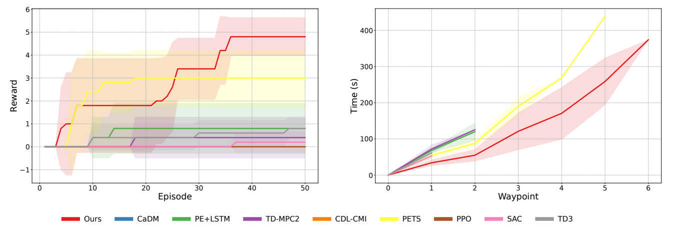

Figure 3: Dynamics Learning Results. Our model achieves a higher reward and completes the tasks in less time. *(Left)* Training task reward. *(Right)* Mean ± std. dev. of the time to reach the task waypoints. Experiments repeated five times.

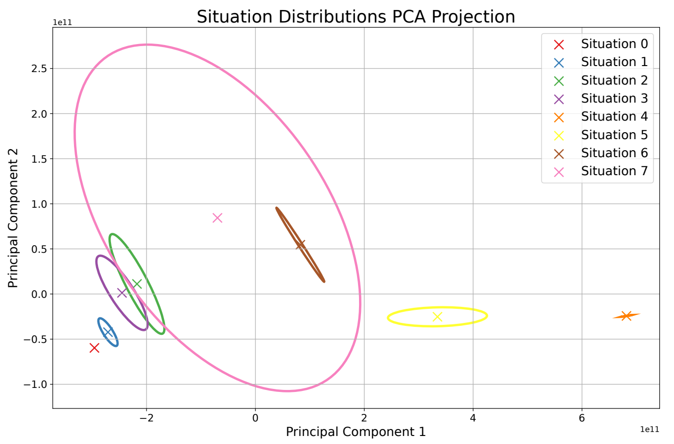

Figure 4: PCA decomposition of the identified Situations (26-dimensional local transition distributions).

A. *Situationally-Aware Dynamics Learning*

The SA module processes the 26-dimensional situation observations to return the current hidden state representation $\theta \in \Theta$, symbolizing the robot's local dynamics. The learned dynamics model $\mathcal{M}_m$ then uses θ_t along with the robot's orientation and commanded velocities to predict state changes.

1) Training Results: To train the model and baselines we design a *sparse* waypoint mission in the Inspection World simulator shown in Figure 5. However, the region around waypoint 3 is steep and makes the robot slip whenever it tries to go there directly. This makes it necessary to come up with a non-trivial plan. To trade-off exploration and exploitation, an episode ends if the robot takes longer than 90 seconds to reach the next waypoint. We train sequentially on a single simulator.

Figure 3 shows that *the situationally-aware dynamics enable faster learning and higher average reward* (number of waypoints reached). Additionally, on average our model is able to complete the task faster and, more significantly, *it is the only one able to complete the task within 50 episodes*.

Quantitatively evaluating the performance of online hidden state representation learning is inherently challenging for this task, as the true hidden state and its transitions are naturally unobservable. To address this, we qualitatively assess the performance of the situational awareness module through two complementary analyses. First, we examine whether the module identified meaningful situation changepoints during the robot's operation, as illustrated in Figure 7. *The results show that the model effectively segmented the local transition dynamics under varying patterns, accurately capturing and representing the changes in the dynamics.* Second, we visualize the diversity of learned hidden state representations by mapping the 26-dimensional local transition dynamics T_L identified onto the first two principal components, as shown in Figure 4. *This visualization reveals distinct clusters corresponding to different situations, demonstrating that each representation encodes unique patterns in the transition dynamics without collapsing into a single, generalized representation.*

B. *Situationally-Aware Unstructured Terrain Navigation*

We extensively evaluate our SA dynamics model in the real-world, comparing it against the physics-based Dubins model and the strongest learning-based baselines in simulation. Notably, we do not perform additional fine-tuning to go from simulation to reality. As illustrated in Figure 2 and in the video (https://youtu.be/VKR18WaSCAk), our method reliably handles various challenging terrains, demonstrating strong robustness and adaptability.

Emerging Behaviors. A highlight of the real-world experiments is the emergence of behavioral patterns that improved safety and performance. For instance,

- *Backing Up:* Whenever forward motion is impeded, the robot backs up and moves with a slight orientation change to overcome the terrain irregularities. This mirrors the behavior of human drivers when their vehicles get stuck. The clips at the: 1:00, 1:27 *(trial 4)*, 2:06, 2:36 *(trial 4)*, 3:30 & 5:25 *(rocks)* minute marks have evidence of this.
- *Velocity Control:* At minute 4:00, the robot maintains a slower pace while navigating downhill and then deliberately accelerates (minute 4:15) upon a situation change–reaching safer, more stable terrain.
- *Trajectory Smoothness:* On very rugged and dangerous terrains the robot followed curved patterns (see the clips at minutes 1:42, 2:22 & 3:18). But, when the terrain was not as rugged and steep, the robot approached its targets directly, relying only on the velocity control to maintain its integrity (4:00, 4:22 & 5:22 minute marks).

It is important to clarify that during training we did not try to enforce learning of such patterns in any way. We hypothesize these behaviors results from situation changes that lead the optimization process carried out by MPPI to be constrained according to the dynamics of the new situation, thus generating new and diverse action plans.

VI. Conclusion

We presented a framework for adaptive decision-making by modeling latent world- and ego-factors through the joint distribution of state transitions, or the robot's *situation*. By extending BOCD to a multivariate setting, our method enables online, unsupervised adaptation without privileged data. Results show improved performance in discovering latent factors while maintaining efficiency and reducing the inductive biases.

References

[1] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[2] Chunge Bai, Tao Xiao, Yajie Chen, Haoqian Wang, Fang Zhang, and Xiang Gao. Faster-lio: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels. *IEEE Robotics and Automation Letters*, 7(2):4861–4868, 2022. doi: 10.1109/LRA.2022.3152830.

[3] Xiaoyi Cai, Michael Everett, Jonathan Fink, and Jonathan P How. Risk-aware off-road navigation via a learned speed distribution map. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2931–2937. IEEE, 2022.

[4] Xiaoyi Cai, Michael Everett, Lakshay Sharma, Philip R Osteen, and Jonathan P How. Probabilistic traversability model for risk-aware motion planning in off-road environments. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11297–11304. IEEE, 2023.

[5] Carlos Celemin and Jens Kober. Knowledge-and ambiguity-aware robot learning from corrective and evaluative feedback. *Neural Computing and Applications*, 35 (23):16821–16839, 2023.

[6] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[7] Clearpath. Clearpath additional simulation worlds. https://github.com/clearpathrobotics/cpr_gazebo, 2020. URL https://clearpathrobotics.com/blog/2020/07/clearpath-robots-get-new-gazebo-simulation-environments/.

[8] John H Curtiss. A note on the theory of moment generating functions. *The Annals of Mathematical Statistics*, 13(4):430–433, 1942.

[9] Morris H DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, 1970.

[10] Noel E Du Toit and Joel W Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2011.

[11] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957. ISSN 00029327, 10806377. URL http://www.jstor.org/stable/2372560.

[12] Paul Fearnhead and Guillem Rigaill. Changepoint detection in the presence of outliers. *Journal of the American Statistical Association*, 114(525):169–183, 2019.

[13] Dave Ferguson, Anthony Stentz, and Sebastian Thrun. Pao for planning with hidden state. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 3, pages 2840–2847. IEEE, 2004.

[14] Catherine Forbes, Merran Evans, Nicholas Hastings, and Brian Peacock. *Statistical distributions*. John Wiley & Sons, 2011.

[15] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[16] Manan S Gandhi, Bogdan Vlahov, Jason Gibson, Grady Williams, and Evangelos A Theodorou. Robust model predictive path integral control: Analysis and performance guarantees. *IEEE Robotics and Automation Letters*, 6(2):1423–1430, 2021.

[17] Kevin Green, Ross L Hatton, and Jonathan Hurst. Planning for the unexpected: Explicitly optimizing motions for ground uncertainty in running. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1445–1451. IEEE, 2020.

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[19] Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023.

[20] Aaron Havens, Yi Ouyang, Prabhat Nagarajan, and Yasuhiro Fujita. Learning latent state spaces for planning through reward prediction. *arXiv preprint arXiv:1912.04201*, 2019.

[21] Hassan Jardali, Mahmoud Ali, and Lantao Liu. Autonomous mapless navigation on uneven terrains. *arXiv preprint arXiv:2402.13443*, 2024.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Taisuke Kobayashi, Shingo Murata, and Tetsunari Inamura. Latent representation in human–robot interaction with explicit consideration of periodic dynamics. *IEEE Transactions on Human-Machine Systems*, 52(5):928–940, 2022.

[24] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2149–2154, Sendai, Japan, Sep 2004.

[25] Oliver Kroemer, Herke Van Hoof, Gerhard Neumann, and Jan Peters. Learning to predict phases of manipulation tasks as hidden states. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4009–4014. IEEE, 2014.

[26] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[27] Ashish Kumar, Zhongyu Li, Jun Zeng, Deepak Pathak, Koushil Sreenath, and Jitendra Malik. Adapting rapid motor adaptation for bipedal robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Sys-*

tems (IROS), pages 1161–1168. IEEE, 2022.

[28] Hanna Kurniawati, Tirthankar Bandyopadhyay, and Nicholas M Patrikalakis. Global motion planning under uncertain motion, sensing, and environment map. *Autonomous Robots*, 33:255–272, 2012.

[29] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5 (47):eabc5986, 2020.

[30] Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 5757–5766. PMLR, 2020.

[31] Abe Leininger, Mahmoud Ali, Hassan Jardali, and Lantao Liu. Gaussian process-based traversability analysis for terrain mapless navigation. *arXiv preprint arXiv:2403.19010*, 2024.

[32] Yichao Liang, Kevin Ellis, and João Henriques. Rapid motor adaptation for robotic manipulator arms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16404–16413, 2024.

[33] Prasanta Chandra Mahalanobis. On test and measures of group divergence: theoretical formulae. 1930.

[34] R Andrew McCallum. Hidden state and reinforcement learning with instance-based state identification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(3):464–473, 1996.

[35] Daniel Meyer-Delius, Maximilian Beinhofer, Alexander Kleiner, and Wolfram Burgard. Using artificial landmarks to reduce the ambiguity in the environment of a mobile robot. In *2011 IEEE International Conference on Robotics and Automation*, pages 5173–5178. IEEE, 2011.

[36] Alejandro Murillo-Gonzalez and Lantao Liu. Situationally-aware dynamics learning. *arXiv preprint arXiv:2505.19574*, 2025.

[37] Kevin P Murphy. Conjugate bayesian analysis of the gaussian distribution. 2007. URL https://www.cs.ubc.ca/~murphyk/Papers/bayesGauss.pdf.

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html.

[39] Christian Perez, Felipe Petroski Such, and Theofanis Karaletsos. Generalized hidden parameter mdps: Transferable model-based rl in a handful of trials. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5403–5411, 2020.

[40] Luis Pineda, Brandon Amos, Amy Zhang, Nathan O. Lambert, and Roberto Calandra. Mbrl-lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021. URL https://arxiv.org/abs/2104.10159.

[41] Pradip Pramanick, Chayan Sarkar, Snehasis Banerjee, and Brojeshwar Bhowmick. Talk-to-resolve: Combining scene understanding and spatial dialogue to resolve granular task ambiguity for a collocated robot. *Robotics and Autonomous Systems*, 155:104183, 2022.

[42] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023.

[43] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[44] Adarsh Jagan Sathyamoorthy, Kasun Weerakoon, Tianrui Guan, Jing Liang, and Dinesh Manocha. Terrapn: Unstructured terrain navigation using online self-supervised learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7197–7204. IEEE, 2022.

[45] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[46] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

[47] Archit Sharma, Michael Ahn, Sergey Levine, Vikash Kumar, Karol Hausman, and Shixiang Gu. Emergent real-world robotic skills via unsupervised off-policy reinforcement learning. *arXiv preprint arXiv:2004.12974*, 2020.

[48] Lakshay Sharma, Michael Everett, Donggun Lee, Xiaoyi Cai, Philip Osteen, and Jonathan P How. Ramp: A risk-aware mapping and planning pipeline for fast off-road ground robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5730–5736. IEEE, 2023.

[49] Sriram Siva, Maggie Wigness, John Rogers, and Hao Zhang. Robot adaptation to unstructured terrains by joint representation and apprenticeship learning. In *Robotics: science and systems*, 2019.

[50] Sriram Siva, Maggie Wigness, John Rogers, and Hao Zhang. Enhancing consistent ground maneuverability by robot adaptation to complex off-road terrains. In *Conference on Robot Learning*, 2021.

[51] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Ax-

iomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[52] Sean J Wang, Samuel Triest, Wenshan Wang, Sebastian Scherer, and Aaron Johnson. Rough terrain navigation using divergence constrained model-based reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

[53] Zizhao Wang, Xuesu Xiao, Zifan Xu, Yuke Zhu, and Peter Stone. Causal dynamics learning for task-independent state abstraction. *arXiv preprint arXiv:2206.13452*, 2022.

[54] Junhong Xu, Kai Yin, Zheng Chen, Jason M Gregory, Ethan A Stump, and Lantao Liu. Kernel-based diffusion approximated markov decision processes for autonomous navigation and control on unstructured terrains. *The International Journal of Robotics Research*, page 02783649231225977, 2024.

[55] Junhong Xu, Kai Yin, Jason M Gregory, Kris Hauser, and Lantao Liu. Boundary-aware value function generation for safe stochastic motion planning. *The International Journal of Robotics Research*, page 02783649241238766, 2024.

[56] Jianhua Yin, Lingxi Li, Zissimos P Mourelatos, Yixuan Liu, David Gorsich, Amandeep Singh, Seth Tau, and Zhen Hu. Reliable global path planning of off-road autonomous ground vehicles under uncertain terrain conditions. *IEEE Transactions on Intelligent Vehicles*, 2023.

[57] Se-Wook Yoo, E-In Son, and Seung-Woo Seo. Traversability-aware adaptive optimization for path planning and control in mountainous terrain. *IEEE Robotics and Automation Letters*, 2024.

[58] Youwei Yu, Yanqing Liu, Fengjie Fu, Sihan He, Dongchen Zhu, Lei Wang, Xiaolin Zhang, and Jiamao Li. Fast extrinsic calibration for multiple inertial measurement units in visual-inertial system. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–07. IEEE, 2023.

APPENDIX

A. Bayesian Online Changepoint Detection (BOCD)

The problem of changepoint detection is concerned with determining the point where the observed data distribution changes in an ordered set of measurements, such as in time-series [12]. BOCD was introduced by Adams and MacKay [1] to tackle this problem by framing it as an estimation of the posterior distribution of the current "run length" r_t, meaning how likely it is that the measurement at time-step t belongs to the same data generating process that started r_t timesteps ago; while also obtaining the parameters $\boldsymbol{\eta}$ defining such process.

Intuitively, BOCD continuously monitors a data stream to identify points where the underlying data distribution changes. It starts with a prior belief about where changes might occur and updates this belief by evaluating how well the current data fits different scenarios of changepoints (segment/run lengths). At each time step, the algorithm uses the new data to adjust the likelihood of potential changepoints, incorporating prior knowledge and observed evidence. The method recursively updates this distribution to assess the existence of changes in the UDGP of the current segment.

Formally, the method assumes that the sequence of observations $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T$ can be segmented into non-overlapping intervals, where a changepoint corresponds to the point $\boldsymbol{x}_i$ marking the transition between two adjacent intervals. The set $\boldsymbol{x}_t^{(r)}$ contains the points estimated to belong to run r_t. Furthermore, the data within each interval ρ is *i.i.d.* from $P(\boldsymbol{x}_t|\boldsymbol{\eta}_\rho)$, and the parameters $\boldsymbol{\eta}_\rho, \rho = 1, 2, \ldots$ are also *i.i.d.* Note that the last assumption is about the *parameters* of the underlying distributions describing the data generating process of each segment. Finally, we should be able to compute the predictive distribution $P(\boldsymbol{x}_{t+1}|r_t, \boldsymbol{x}_t^{(r)})$ and define a conditional prior on the changepoint $P(r_t|r_{t-1})$.

To initialize the algorithm, BOCD considers two scenarios: *(a)* the changepoint took place before the initial observation, then $P(r_0 = 0) = 1$, and *(b)* we are seeing a recent subset of the data, then $P(r_0 = \tau) = \frac{1}{Z}\sum_{t=\tau+1}^{\infty} P_{\text{gap}}(g = t)$, where Z is a normalizing constant and $P_{\text{gap}}(g)$ is the multinomial prior across the changepoints' domain.

The conditional changepoint prior $P(r_t|r_{t-1})$ is the key for the algorithm's performance, since it is defined to have non-zero mass in two of three cases:

$$P(r_t|r_{t-1}) = \begin{cases} H(r_{t-1}+1) & \text{if } r_t = 0, \\ 1 - H(r_{t-1}+1) & \text{if } r_t = r_{t-1}+1, \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

where $H(\tau)$ is the *hazard* function [14]. Subsequently, finding the distribution of the run length $P(r_t|\boldsymbol{x}_{1:t})$, involves recursive computation over the run length and the observations:

$$\begin{aligned} P(r_t, \boldsymbol{x}_{1:t}) &= \sum_{r_{t-1}} P(r_t, r_{t-1}, \boldsymbol{x}_{1:t}) \quad (16) \\ &= \sum_{r_{t-1}} P(r_t, \boldsymbol{x}_t | r_{t-1}, \boldsymbol{x}_{1:t-1}) P(r_{t-1}, \boldsymbol{x}_{1:t-1}) \\ &= \sum_{r_{t-1}} P(r_t|r_{t-1}) P(\boldsymbol{x}_t|r_{t-1}, \boldsymbol{x}_t^{(r)}) P(r_{t-1}, \boldsymbol{x}_{1:t-1}). \end{aligned}$$

where the predictive distribution $P(\boldsymbol{x}_t|r_{t-1}, \boldsymbol{x}_{1:t})$ depends only on the data of the current run $\boldsymbol{x}_t^{(r)}$.

$P(r_t|\boldsymbol{x}_{1:t})$ is updated with each new observation $\boldsymbol{x}_t$, by first obtaining its predictive probability $\pi_t^{(r)} = P(\boldsymbol{x}_t|\boldsymbol{\eta}_t^{(r)})$, and then calculating the *growth probability* (probability that the run has not ended and we are still observing data from the same UDGP):

$$\begin{aligned} P(r_t = r_{t-1}+1, \boldsymbol{x}_{1:t}) &= \quad (17) \\ & P(r_{t-1}, \boldsymbol{x}_{1:t-1}) \pi_t^{(r)} (1 - H(r_{t-1})). \end{aligned}$$

and *changepoint probability* (probability that the UDGP has changed):

$$P(r_t = 0, \boldsymbol{x}_{1:t}) = \sum_{r_{t-1}} P(r_{t-1}, \boldsymbol{x}_{1:t-1}) \pi_t^{(r)} H(r_{t-1}). \quad (18)$$

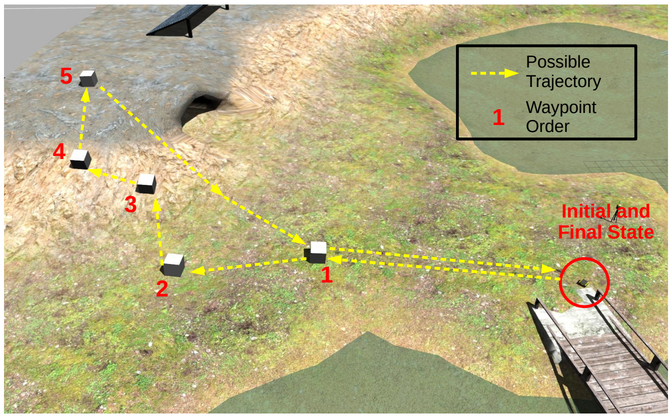

Figure 5: Training task in the *Inspection World Environment* [7]. The region around waypoint #3 is slippery and difficult to reach.

From which we can obtain,

$$P(r_t|\boldsymbol{x}_{1:t}) = \frac{P(r_t, \boldsymbol{x}_{1:t})}{P(\boldsymbol{x}_{1:t})} = \frac{P(r_t, \boldsymbol{x}_{1:t})}{\sum_{r_t} P(r_t, \boldsymbol{x}_{1:t})}. \tag{19}$$

Finally, after obtaining $P(r_t|\boldsymbol{x}_{1:t})$ using the latest observation $\boldsymbol{x}_t$, we update the sufficient statistics of the data generating process' model.

B. Additional Experimental Details

All code was developed using ROS Noetic [43], Python 3.10.11 and PyTorch 2.3.0+cu121 [38]. For the MPPI we use the `MBRL-Lib` package [40]. In the real-world, for state estimation we used FasterLIO with IMU preintegration on manifold [2, 58] with the Velodyne 16 LiDAR and IMU 3DM-GX5-AHRS. We ran all code on-board the Jackal differential drive robot with an NVIDIA Jetson Orin.

The model $\mathcal{M}_m$ is composed of $m = 5$ probabilistic neural networks, with three dense layers with 200 neurons each and LeakyReLU non-linearities in between. The output is the mean and log-variance of a Gaussian distribution. Inputs and outputs are normalized using data statistics computed during training. At inference time, predictions are de-normalized with respect to the data statistics obtained during training. We use the Adam optimizer with default hyperparameters [22]. The batch size is 256 with an early-stopping threshold of 1e-3. The model $\mathcal{M}_m$ starts with randomly initialized parameters and is trained at the beginning of each episode with the updated replay buffer $\mathcal{R}$.

The sparse-waypoints training task in the Inspection World simulator is shown in Figure 5.

C. Additional Difficult Terrain Navigation Results

1) Effect of the Hidden State Representation: We use the Integrated Gradients [51] feature attribution method to verify that our situationally-aware dynamics model relies on the different situations to make its predictions, thus influencing the downstream planner's control strategies. Figure 6 shows that on average, 60% of the prediction is explained by the current situation the robot is in, while the remaining input features

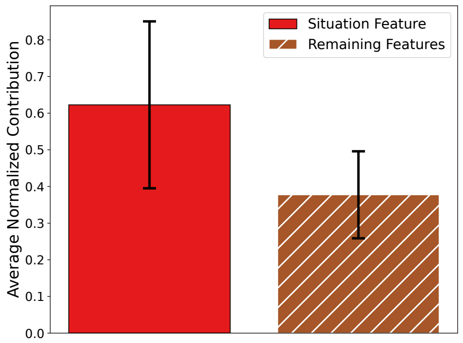

Figure 6: Dynamics model input features contributions to the prediction.

together account for roughly 40%. This clearly highlights that the model is aware of the situation the robot is experiencing and modulates its predictions accordingly.

2) Multivariate BOCD: Figure 7 presents the full observation history during execution of a task by Jackal in the Inspection world, offering a more detailed view that highlights how effectively our multivariate extension of BOCD detects changes in the UDGP of the robot's dynamics during an unstructured terrain navigation task.

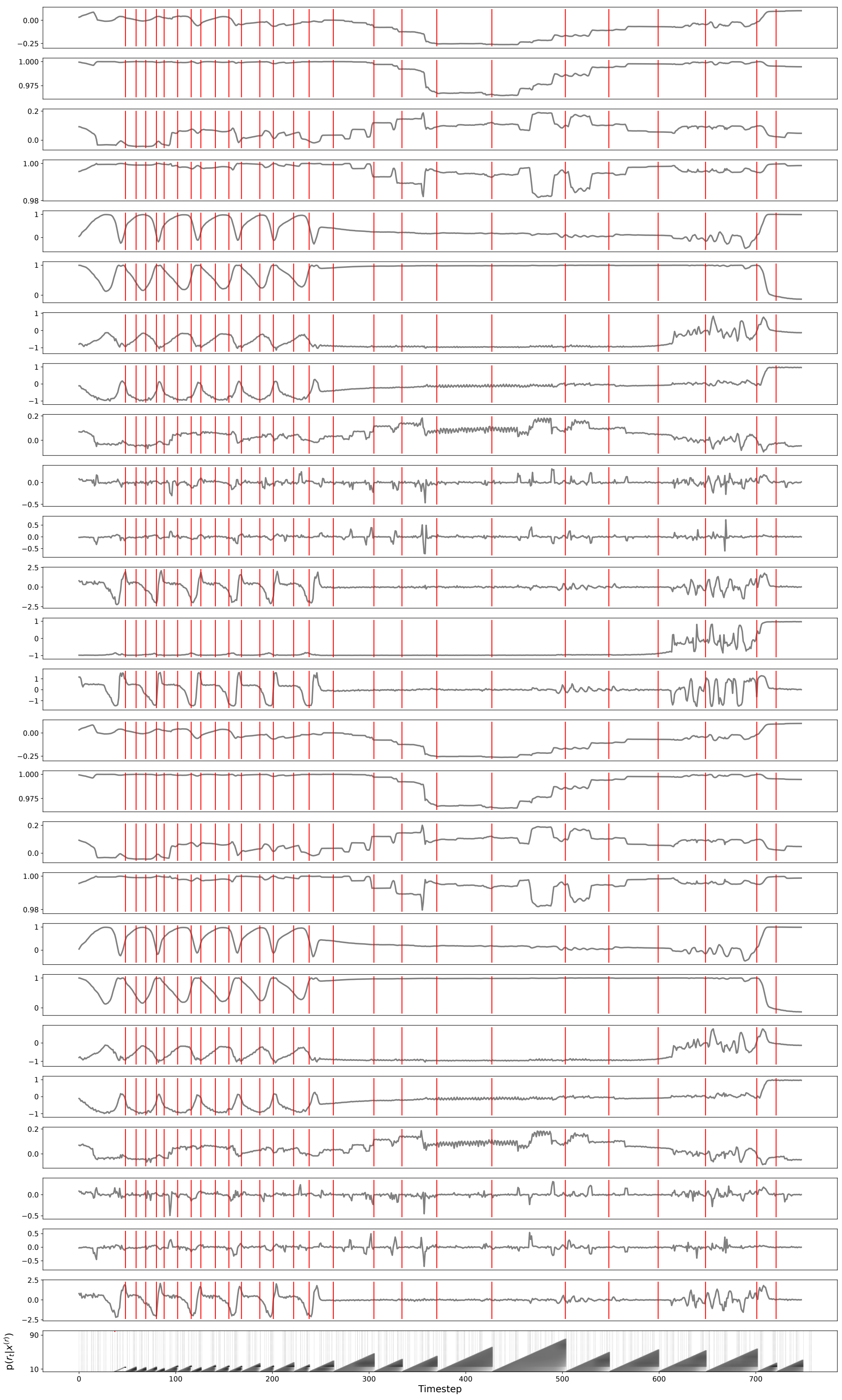

Figure 7: Observations received by the proposed situation identification algorithm during a mission in the Inspection World. The red vertical lines mark the time when we predict a situation change happened. The last row shows the run length probabilities indicating how likely it is that the robot stayed in the same situation (run length grows) or a situation change happened (run length becomes zero).