From Tabula Rasa to Emergent Abilities: Discovering Robot Skills via Real-World Unsupervised Quality-Diversity

Luca Grillotti **Lisa Coiffard** **Oscar Pang** **Maxence Faldor** **Antoine Cully**
Adaptive & Intelligent Robotics Lab
Imperial College London

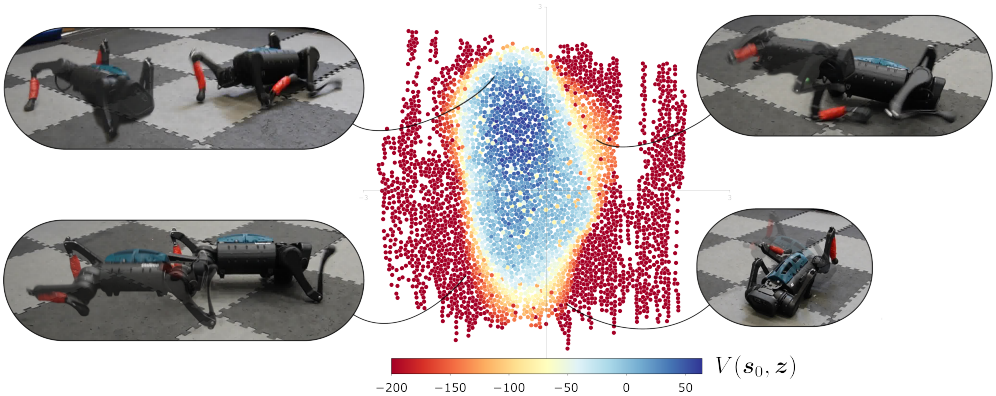

Figure 1: We propose Unsupervised Real-world Skill Acquisition (URSA), a framework for unsupervised quality-diversity in real-world environments. Each skill is plotted in the latent space with color indicating its estimated value, $V(s_0, z)$, highlighting the diversity of learned behaviors.

Abstract: Autonomous skill discovery aims to enable robots to acquire diverse behaviors without explicit supervision. Learning such behaviors directly on physical hardware remains challenging due to safety and data efficiency constraints. Existing methods, including Quality-Diversity Actor-Critic (QDAC), require manually defined skill spaces and carefully tuned heuristics, limiting real-world applicability. We propose Unsupervised Real-world Skill Acquisition (URSA), an extension of QDAC that enables robots to autonomously discover and master diverse, high-performing skills directly in the real world. We demonstrate that URSA successfully discovers diverse locomotion skills on a Unitree A1 quadruped in both simulation and the real world. Our approach supports both heuristic-driven skill discovery and fully unsupervised settings. We also show that the learned skill repertoire can be reused for downstream tasks such as real-world damage adaptation, where URSA outperforms all baselines in 5 out of 9 simulated and 3 out of 5 real-world damage scenarios. Our results establish a new framework for real-world robot learning that enables continuous skill discovery with limited human intervention, representing a significant step toward more autonomous and adaptable robotic systems. Demonstration videos are available at adaptive-intelligent-robotics.github.io/URSA.

1 Introduction

Animals exhibit the remarkable ability to develop diverse movement patterns through autonomous interaction with their environment. Inspired by this, autonomous skill discovery aims to endow robots with a similar capacity: the ability to learn a wide repertoire of behaviors through self-directed

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

experience. Such behavioral diversity is critical for general-purpose robots that must operate in dynamic, unpredictable environments. While specialized robots have shown strong performance in domains like PCB insertion [1] and legged locomotion [2], they remain limited to a narrow set of pre-defined behaviors. In contrast, a robot equipped with a diverse skill set can adapt to new tasks and conditions, such as recovering from damage [3, 4, 5] and navigating on planar terrains [6, 7].

Quality-Diversity (QD) algorithms [8, 9] offer a promising foundation for building such skill repertoires, with the aim to discover a set of behaviors that are both diverse (e.g. different gaits) and individually high-performing (e.g. forward velocity). While traditional QD methods rely on manually defined behavior descriptors, recent work [10, 11, 12] learns these skill representations directly from data, in an unsupervised manner. However, applying QD to physical robots remains challenging—either it depends on simulation, which requires accurate modelling and reliable sim-to-real transfer, or it must operate directly in the real world, where methods like evolutionary algorithms tend to be too sample-inefficient for practical deployment.

In this work, we introduce Unsupervised Real-world Skill Acquisition (URSA), a framework for unsupervised QD that operates directly on physical robots. URSA extends the Quality-Diversity Actor-Critic (QDAC) framework [13] for unsupervised learning and leverages the DayDreamer [14] world model as its backbone, enabling efficient skill acquisition directly on hardware without relying on simulation or sim-to-real transfer. We evaluate URSA on a Unitree A1 quadruped and show it outperforms baselines in 5 out of 9 simulated and 3 out of 5 real-world damage scenarios. This demonstrates its potential for enabling adaptable, resilient skill discovery directly on physical robots. Our key contributions are:

- We propose URSA, a framework for unsupervised QD that extends QDAC with a learned latent skill space, safety-aware optimization, and efficient sampling.
- We show that URSA can learn diverse locomotion behaviors directly in the real world using imagination-based training with world models.
- We demonstrate that the learned skill repertoire enables adaptation to physical damage by supporting selective skill reuse in downstream tasks.

2 Problem Statement: A New Perspective on Skill Discovery

We model the environment as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, r, p)$ [15], where at each timestep t, the agent in state $\boldsymbol{s}_t \in \mathcal{S}$ takes an action $\boldsymbol{a}_t \in \mathcal{A}$, transitions to $\boldsymbol{s}_{t+1} \sim p(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$, and receives reward $r_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t)$. We assume access to a feature function $\phi : \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$, either learned or user-defined, that captures instantaneous behavioral properties, such as velocity or foot contact patterns in a robot.

To summarize behavior over time, we define a *skill* as the expected feature vector under a policy's stationary distribution.

Definition 2.1 (Skill)**.** The skill $\boldsymbol{z} \in \mathcal{Z}$ of a policy π is defined as $\boldsymbol{z} = \mathbb{E}_\pi\left[\phi(\boldsymbol{s}, \boldsymbol{a})\right]$.

This expectation captures the characteristic behavior induced by a policy. For example, in a quadrupedal robot where $\boldsymbol{\phi}_t[i] = 1$ if the i-th foot is on the ground and 0 otherwise, the skill $\boldsymbol{z}$ represents the episodic proportion of contact time per foot. A vector like $\boldsymbol{z} = [0.8 \quad 0.3 \quad 0.8 \quad 0.3]^\intercal$ describes a gait where the left feet are used 80% of the time and the right feet 30%, potentially corresponding to a limping motion.

Within the skill space $\mathcal{Z}$, we distinguish two subspaces: the reachable skill space of all theoretically attainable skills, and the achieved skill space of skills actually mastered by our policy.

Definition 2.2 (Reachable Skill Space)**.** The reachable skill space $\mathcal{Z}_p \subseteq \mathcal{Z}$ is the set of all skills $\boldsymbol{z} \in \mathcal{Z}$ for which some policy π can achieve them: $\mathcal{Z}_p = \{\boldsymbol{z} \in \mathcal{Z} \mid \exists \pi, \mathbb{E}_\pi\left[\phi(\boldsymbol{s}, \boldsymbol{a})\right] = \boldsymbol{z}\}$

Definition 2.3 (Achieved Skill Space of a Policy)**.** A skill $\boldsymbol{z} \in \mathcal{Z}$ is said to be *achieved* by π if and only if $\mathbb{E}_{\pi_{\boldsymbol{z}}}\left[\phi(\boldsymbol{s}, \boldsymbol{a})\right] = \boldsymbol{z}$, where $\pi_{\boldsymbol{z}}$ represents the skill-conditioned policy $\pi(\cdot|\cdot, \boldsymbol{z})$. The achieved skill space $\mathcal{Z}_\pi \subseteq \mathcal{Z}$ is the set of all achieved skills: $\mathcal{Z}_\pi = \{\boldsymbol{z} \in \mathcal{Z} \mid \mathbb{E}_{\pi_{\boldsymbol{z}}}\left[\phi(\boldsymbol{s}, \boldsymbol{a})\right] = \boldsymbol{z}\}$

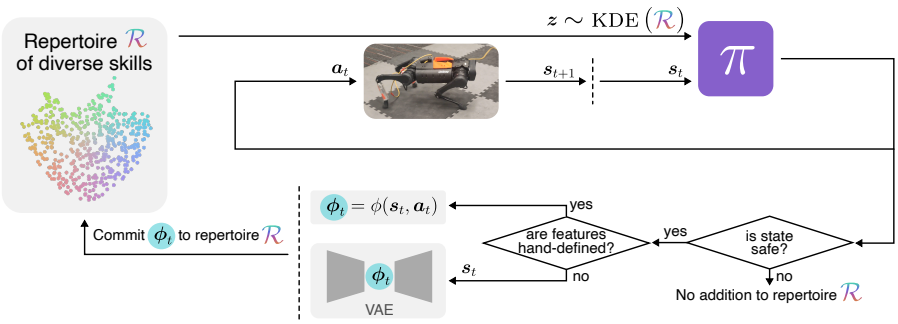

Figure 2: Overview of URSA: The system checks if the state $\boldsymbol{s}_t$ is safe, if so, encodes it into features $\boldsymbol{\phi}_t$, and builds a diverse skill repertoire $\mathcal{R}$. New skills are sampled using a Kernel Density Estimator on the repertoire from the safe, reachable skill space. The skill-conditioned policy $\pi_{\boldsymbol{z}}$ maximizes its expected return while matching the sampled skill $\boldsymbol{z}$.

By construction, it follows that $\mathcal{Z}_\pi \subseteq \mathcal{Z}_p \subseteq \mathcal{Z}$. In this work, we aim to learn these spaces, with the ultimate goal of (1) maximizing the volume of the achieved skill space $\mathrm{vol}\left(\mathcal{Z}_\pi\right)$, thereby encouraging behavioral diversity, and (2) ensuring that each skill is performed with high expected return. However, jointly optimizing for both diversity and performance across a potentially high-dimensional skill space poses significant challenges.

To make this problem tractable, we introduce a surrogate probability distribution $q(\cdot)$ over $\mathcal{Z}$, designed to approximate a uniform distribution over $\mathcal{Z}_p$ and decompose the objective into two components:

- **Diversity:** Maximize the entropy of the skill distribution q over the reachable skill space $\mathcal{Z}_p$.
- **Performance:** Learn a skill-conditioned policy $\pi_{\boldsymbol{z}}$ that maximizes the expected return while achieving each sampled skill $\boldsymbol{z} \sim q$:

$$\text{maximize} \quad \mathbb{E}_{\pi_{\boldsymbol{z}}}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}\right] \quad \text{subject to} \quad \mathbb{E}_{\pi_{\boldsymbol{z}}}\left[\phi(\boldsymbol{s}, \boldsymbol{a})\right] = \boldsymbol{z} \tag{P1}$$

3 Background

3.1 Quality-Diversity Actor-Critic (QDAC)

Our proposed method builds upon the QDAC algorithm [13], and extends it to unsupervised QD in a real-world setting. QDAC aims at finding a skill-conditioned policy $\pi_{\boldsymbol{z}}$ that maximizes the reward while following a given skill $\boldsymbol{z}$, effectively tackling problem P1. It does this by making two approximations: (1) the expected state is approximated by the discounted average $(1-\gamma)\boldsymbol{\psi}(\boldsymbol{s}, \boldsymbol{z})$, and (2) the strict equality constraint from problem P1 is replaced by an inequality constraint forcing the policy to stay close to the target skill $\boldsymbol{z}$. This defines a new problem:

$$\forall \boldsymbol{z} \sim \mathcal{U}\left(\mathcal{Z}_p\right), \quad \text{maximize } V(\boldsymbol{s}, \boldsymbol{z}) \quad \text{subject to } \left\|(1-\gamma)\boldsymbol{\psi}(\boldsymbol{s}, \boldsymbol{z}) - \boldsymbol{z}\right\|_2 \leq \delta \tag{P2}$$

where $V(\boldsymbol{s}, \boldsymbol{z}) = \mathbb{E}_{\pi_{\boldsymbol{z}}}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} \middle| \boldsymbol{s}_t = \boldsymbol{s}\right]$ is the value function [15], $\boldsymbol{\psi}(\boldsymbol{s}, \boldsymbol{z}) = \mathbb{E}_{\pi_{\boldsymbol{z}}}\left[\sum_{i=0}^{\infty} \gamma^i \boldsymbol{\phi}_{t+i} \middle| \boldsymbol{s}_t = \boldsymbol{s}\right]$ successor features [16] and δ is a hyperparameter that determines the maximal acceptable distance between the expected state and the skill.

Our approach addresses two core limitations of QDAC. First, rather than assuming a predefined reachable skill space $\mathcal{Z}_p$ and a fixed constraint threshold δ, we learn both the structure of the skill space and how to efficiently sample from it during training, while adaptively tuning δ. Second, instead of relying on manually defined features, we extract them from unsupervised encodings of the agent's state, enabling skill discovery directly from raw observations.

3.2 Real-world Robot Learning with DayDreamer

DayDreamer [14] is a model-based RL algorithm that enables efficient real-world robot learning by predicting environment dynamics in a latent space using a recurrent state-space model (RSSM). Its

Algorithm 1 URSA – Data Collection

input Parameters θ_π ▷ Initial params. for π
$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay buffer
repeat until convergence
  $\boldsymbol{z} \sim \text{KDE}(\mathcal{R})$
  for T steps **do**
    $\boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t, \boldsymbol{z})$
    $\boldsymbol{s}_{t+1} \sim p(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$
    $r_t \leftarrow r(\boldsymbol{s}_t, \boldsymbol{a}_t)$
    $\boldsymbol{\phi}_t \leftarrow \phi(\boldsymbol{s}_t, \boldsymbol{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{\phi}_t, \boldsymbol{s}_{t+1})\}$
    if $\boldsymbol{s}_t \in \mathcal{S}_{\text{safe}}$ **then**
      commit $\phi(\boldsymbol{s}_t, \boldsymbol{a}_t)$ to repertoire $\mathcal{R}$
  Fetch θ_π and θ_ϕ (if unsupervised) from Algo. 2

All highlighted components are specific to URSA

Algorithm 2 URSA – Training

input Parameters $\theta_{(\cdot)}$ ▷ Initial params. for π, V, $\boldsymbol{\psi}$, C, λ_i, $\mathcal{W}$, and ϕ (if unsupervised)
repeat until convergence
  Fetch $\mathcal{D}$ and $\mathcal{R}$ from Algo. 1
  $\widetilde{\boldsymbol{z}}_i \sim \text{KDE}(\mathcal{R})$ for $i \in \{1, \ldots, N\}$
  $\theta_{\mathcal{W}} \leftarrow \theta_{\mathcal{W}} - \alpha_{\mathcal{W}} \nabla J_{\mathcal{W}}(\theta_{\mathcal{W}})$
  ▷Training from rollouts in $\mathcal{W}$ with skills $\widetilde{\boldsymbol{z}}_i$ (Fig. 8)
  $\theta_{\lambda_i} \leftarrow \theta_{\lambda_i} - \alpha_\lambda \nabla J_{\lambda_i}(\theta_{\lambda_i})$ for $i \in \{1, 2\}$
  $\theta_V \leftarrow \theta_V - \alpha_V \nabla J_V(\theta_V)$
  $\theta_{\boldsymbol{\psi}} \leftarrow \theta_{\boldsymbol{\psi}} - \alpha_{\boldsymbol{\psi}} \nabla J_{\boldsymbol{\psi}}(\theta_{\boldsymbol{\psi}})$
  $\theta_C \leftarrow \theta_C - \alpha_C \nabla J_C(\theta_C)$
  $\theta_\pi \leftarrow \theta_\pi + \alpha_\pi \nabla J_\pi(\theta_\pi)$
  if unsupervised **then**
    $\theta_\phi \leftarrow \theta_\phi - \alpha_\phi \nabla J_\phi(\theta_\phi)$ with $\mathcal{R}$

asynchronous architecture comprises a learner thread for model and policy updates and an actor thread for data collection, allowing continuous training and effective use of limited real-world interactions through imagined rollouts. We build on this framework by integrating unsupervised QD into both threads, enabling robots to autonomously explore and master diverse behaviors.

4 Unsupervised Real-world Skill Acquisition

We propose Unsupervised Real-world Skill Acquisition (URSA), an RL framework for autonomously discovering diverse skills in real-world environments without prior knowledge of the skill space. Built on QDAC [13], URSA introduces three extensions: (1) safety constraints to prevent unsafe behaviors, (2) a skill repertoire $\mathcal{R}$ to store skills, and (3) an optional learnable feature function $\phi(\cdot)$ for compact skill representations from observations. URSA leverages the DayDreamer architecture [14], running two asynchronous threads: one for collecting real-world data (Algorithm 1), and another for updating the world model and policy via imagined rollouts (Algorithm 2, see Figure 8 in appendix B.4).

4.1 Ensuring Safe Skill Discovery

In real-world robotics, some skills, such as falling, can damage the robot and should be avoided. To handle this, we use a mechanism following the constrained RL [17] framework. More specifically, we define a user-provided cost function $c : \mathcal{S} \to \mathbb{R}$ such that $c(\boldsymbol{s}) \leq 0$ if and only if $\boldsymbol{s}$ is in a hand-defined set of safe states $\mathcal{S}_{\text{safe}}$. We ensure that the associated critic $C(\boldsymbol{s}, \boldsymbol{z}) = \mathbb{E}_{\pi_{\boldsymbol{z}}}\left[\sum_{i=0}^{\infty} \gamma^i c_{t+i} | \boldsymbol{s}_t = \boldsymbol{s}\right]$ remains non-positive for every skill $\boldsymbol{z}$ learned by URSA. Incorporating safety into skill discovery yields the constrained optimization problem (see Appendix B.1 for further implementation details):

$$\text{maximize } V(\boldsymbol{s}, \boldsymbol{z}) \quad \text{subject to } \|(1-\gamma)\boldsymbol{\psi}(\boldsymbol{s}, \boldsymbol{z}) - \boldsymbol{z}\|_2 \leq \delta \quad \text{and } C(\boldsymbol{s}, \boldsymbol{z}) \leq 0 \tag{P3}$$

4.2 Efficient Skill Sampling from the Reachable Space

To maximize the diversity of achieved skills, URSA must efficiently explore the reachable skill space $\mathcal{Z}_p$ while ensuring safe execution. URSA addresses this by maintaining a diversity-aware repertoire $\mathcal{R}$ and sampling skills from a flexible surrogate distribution $q(\cdot)$ learned from experience. Figure 2 and Algorithm 1 provide high-level overviews of the skill collection and sampling process.

Adaptive Skill Distribution via Non-Parametric Density Estimation We model q as a Gaussian Kernel Density Estimator (KDE) [18, 19], which provides a flexible, non-parametric way to capture the structure of the reachable skill space for sampling physically achievable skills. We fit this distribution on the repertoire $\mathcal{R} = \{\boldsymbol{z}_i\}_{i=1}^{N_{\mathcal{R}}}$ as $q = \text{KDE}(\mathcal{R}) = \frac{1}{N_{\mathcal{R}}} \sum_{i=1}^{N_{\mathcal{R}}} \mathcal{N}(\boldsymbol{z}_i, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$

is the repertoire's skill covariance matrix, with a scaling factor of $N_{\mathcal{R}}^{-\frac{1}{D+4}}$ (where D denotes the dimensionality of the skill space), following the approach of Scott [20].

Maximizing Entropy for Uniform Skill Sampling To approximate a uniform sampling of $\mathcal{Z}_p$, we maximize the entropy of the sampling distribution q by maintaining a diverse repertoire. This entropy can be approximated via Monte-Carlo sampling with the skills from the repertoire: $\mathrm{H}(q) = -\int q(x)\log q(x)\mathrm{d}x \approx -\frac{1}{N_{\mathcal{R}}}\sum_{i=1}^{N_{\mathcal{R}}} q(\boldsymbol{z}_i)\log q(\boldsymbol{z}_i)$. This estimated entropy, written $\widehat{\mathrm{H}}(q)$, has a tractable lower bound based on nearest-neighbor Mahalanobis distances $d_{\boldsymbol{\Sigma}}$, derived in Appendix C:

$$\widehat{\mathrm{H}}(q) \geq -\frac{1}{N_{\mathcal{R}}}\sum_{i=1}^{N_{\mathcal{R}}} \log\left(1 + (N_{\mathcal{R}} - 1)e^{-\frac{1}{2}d_{\boldsymbol{\Sigma}}(\boldsymbol{z}_i, \boldsymbol{z}_i^{\mathrm{nn}})^2}\right) + \frac{1}{2}\log\left(\det(\boldsymbol{\Sigma})\right) + k \tag{1}$$

where k is a constant and $\boldsymbol{z}_i^{\mathrm{nn}}$ is the nearest-neighbor of skill $\boldsymbol{z}_i$ in the repertoire. To maximize this bound, we continuously update the repertoire by replacing existing skills with newly discovered ones that increase the distribution's spread.

Filling the Repertoire with Safe and Reachable Skills We maximize diversity within the reachable space by adding features that result in *safe* states to the repertoire. When a new feature $\boldsymbol{\phi}_t$ is committed, we compute distances $d_{\boldsymbol{\Sigma}}(\boldsymbol{z}, \boldsymbol{z}^{\mathrm{nn}})$ for all $\boldsymbol{z} \in \mathcal{R} \cup \{\boldsymbol{\phi}_t\}$, and remove the skill with the smallest distance, thereby keeping the repertoire size fixed. Assuming the impact of those updates on $\boldsymbol{\Sigma}$ is negligible, URSA maximizes the lower bound on the approximated entropy of the skill-sampling distribution q (see Equation 1). This process maintains a uniform coverage of the safe and reachable skill space, while continuously increasing diversity through the discovery of new skills.

Dynamic Threshold As the repertoire expands, the typical distance between skills increases, requiring adaptive tuning of the constraint threshold δ in Problem P3. To that end, we introduce a hyperparameter $N_{\boldsymbol{z}}$, which controls the number of distinct skills that the robot can execute. We compute δ as the mean nearest-neighbor distance, if the repertoire contained $N_{\boldsymbol{z}}$ uniformly distributed skills (see equation in Appendix B.5).

This ensures that executing one skill does not inadvertently satisfy multiple constraints.

4.3 Unsupervised Quality-Diversity

When a feature function is not provided, URSA learns one automatically from raw state observations. We implement $\phi(\cdot)$ using a variational autoencoder (VAE) [21], which encodes zeroth-order kinematics (e.g., joint angles, body height) from $\mathcal{S}$ into a compact latent representation.

Following Definition 2.1, a skill $\boldsymbol{z}$ is defined as the expected latent encoding under the policy's stationary distribution. The VAE is trained on the diverse set of states collected in the repertoire $\mathcal{R}$, enabling unsupervised discovery of meaningful skills without human supervision.

5 Results

5.1 Experimental Setup and Baselines

We evaluate URSA on the Unitree A1 quadruped in both PyBullet simulation and real-world environments. The robot state includes joint angles and velocities for its 12 joints; actions specify target joint positions executed via a 20 Hz PD controller. Skills are sampled every 250 steps from a fixed repertoire of $N_R = 4096$. Training in simulation runs for 1 million timesteps, while in the real world, we collect 5 hours of data with asynchronous data collection and training on a single NVIDIA RTX 6000 Ada GPU. To match real-world conditions, episodes in simulation proceed continuously without manual resets to predetermined states. Additional training details and all hyperparameters are provided in Appendix D. Demonstration videos are available at adaptive-intelligent-robotics.github.io/URSA.

We compare against two baselines: **DayDreamer** [14], a single-skill learner corresponding to URSA without the successor features constraint in P3; and **DOMiNiC** [22], which incorporates successor

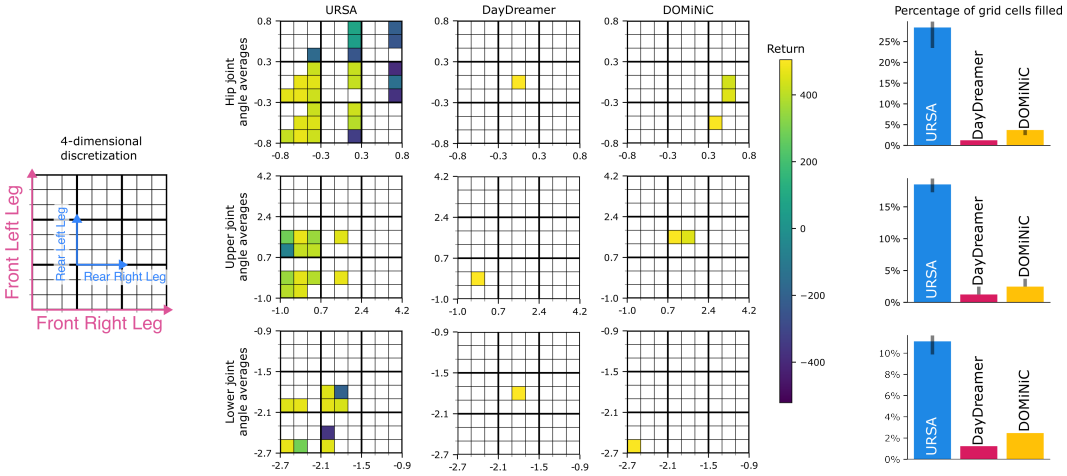

Figure 3: Average joint angles across the skill repertoire in URSA, DayDreamer, and DOMiNiC. Each cell represents average joint angles (hip, upper, and lower) for all leg combinations. Cells are colored if at least one skill's average falls within that region.

features and enforces a near-optimality constraint, discovering multiple policies only after achieving a high reward. Following the original implementation, we set the number of skills in DOMiNiC to 8. To ensure a fair comparison, we implemented DOMiNiC using the same DayDreamer backbone as our method, maintaining architectural consistency across all compared approaches.

Our evaluation addresses three key questions: **(RQ1)** Can URSA learn diverse, high-performing and unsupervised locomotion skills? **(RQ2)** Can these skills be reused for downstream tasks like damage adaptation? **(RQ3)** Can URSA discover useful, controllable skills for target-driven tasks?

5.2 RQ1: Unsupervised Discovery of Diverse and High-Performing Locomotion Skills

To evaluate URSA's ability to learn diverse locomotion behaviors in an unsupervised setting, we train it to acquire forward movement skills without predefined behavioral objectives. Features ϕ_t are represented as 2-dimensional latent vectors encoded from raw states using a VAE. The reward function encourages forward velocity while maintaining stability, and the hand-defined set of safe states $\mathcal{S}_{\text{safe}}$ ensures upright posture (see Appendix D.1 for details).

Figure 3 illustrates the distribution of average joint angles, evaluated in simulation, across different leg combinations. URSA discovers a wide variety of motion patterns through varied use of the leg joints. This is further supported by Figure 1, which visualizes four example behaviors from the repertoire, ranging from gaits that keep the torso near the ground to more agile, coordinated walking motions. By contrast, DayDreamer converges on a single locomotion strategy that maximizes return, while DOMiNiC—though able to learn multiple skills—enforces near-optimality, resulting in less joint-level diversity across its 8 discrete behaviors. Quantitatively, when projecting the joint angle trajectories onto discretized spaces of average angles, URSA achieves 4 times greater coverage compared to the baselines, demonstrating significantly more behavioral diversity (Fig. 3).

5.3 RQ2: Downstream Use of Skills for Damage Adaptation

To evaluate the utility of URSA's learned repertoire for downstream tasks, we study its effectiveness in adapting to physical damage across simulation (Figure 4) and real-world (Figure 5) scenarios. We apply damage scenarios including single joint failures and full failures of one or two legs[1].

In Figure 4, we report an upper bound for URSA by executing all skills in the repertoire and recording the highest return. Given the large number of distinct skills (see N_z in Appendix D.4), we also

[1]Joint notation: F = Front, B = Back, R = Right, L = Left (e.g., FR = Front Right, BL = Back Left).

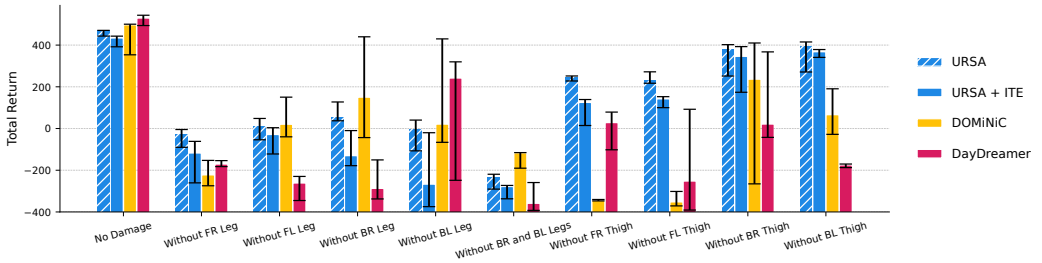

Figure 4: Comparison of returns across joint damage scenarios in simulation. The best return for URSA is shown in hatched bars, compared to a version using ITE for adaptation. Results display the median return and interquartile range (IQR) across 5 independent runs.

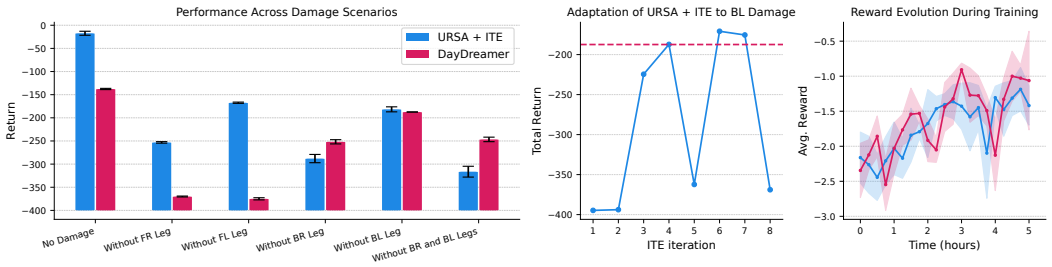

Figure 5: **Left**: Comparison of returns across damage scenarios in the real world, showing median return and IQR across 2 independent runs. **Middle**: Evolution of attempted skills for a single run of ITE with FL leg damage. **Right**: Average reward during training (shaded area as standard deviation of 15-minute segments).

evaluate URSA combined with Iterative Trial and Error (ITE) [3] (URSA + ITE), which uses Bayesian optimization to find the best-performing skill in a few trials (see Appendix D.2). URSA + ITE outperforms both DOMiNiC and DayDreamer in upper-leg joint damage. However, with more severe damage (e.g., full leg failures), the near-optimality constraint of DOMiNiC proves more beneficial than URSA's diversity-aware mechanism.

In Figure 5, we evaluate only URSA + ITE due to the impracticality of executing all skills. URSA outperforms DayDreamer in all damage scenarios except for back-right leg failures (left panel). Notably, URSA demonstrates reduced overall performance degradation under damage, leveraging its diverse skill repertoire to compensate for broken joints. This is evident in Figure 5 (middle), where URSA + ITE progressively improves total return after consecutive iterations, recovering within 8 iterations during back-left leg damage. The underlying skill repertoire's quality is shown in Figure 5 (right), where consistent improvement in average rewards indicates that URSA builds a diverse set of high-performing behaviors. In contrast, DayDreamer tends to overfit to forward locomotion strategies, leading to significant performance drops when the front legs are damaged. Additionally, URSA achieves higher returns in the undamaged setting, suggesting that behavioral diversity not only improves robustness but also acts as a stepping stone toward discovering high-performing locomotion strategies, as opposed to single-skill optimization.

5.4 RQ3: Heuristic-Based Skill Discovery for Velocity Tracking

To assess whether URSA can discover useful and controllable behaviors, we set $\phi(s_t) = [v_{x,t}, \omega_{z,t}]$ to capture forward and angular velocities. The reward encourages stable posture and smooth movement, and a hand-designed safe state set maintains upright poses (see Appendix D.3).

We assess the usefulness of the discovered skills by measuring how well they track target velocity commands. Figure 6 reports the tracking error across a grid of desired forward and angular velocities. URSA learns a repertoire that spans a broad range of these velocities, as confirmed by the plotted

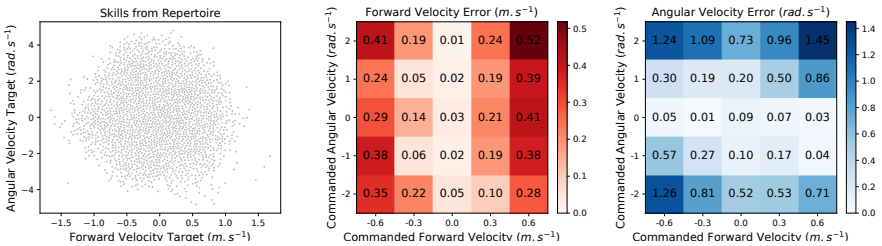

Figure 6: Velocity tracking errors during skill execution, evaluating the robot's accuracy in following target velocity commands across the reachable space discovered by URSA. Lower values indicate better control.

skill distribution (Fig. 6, left). However, we observe that tracking performance degrades as target velocities deviate further from zero, particularly in the case of forward velocity. This discrepancy likely arises because some target velocities, while instantaneously achievable, cannot be consistently sustained across the entire duration of a skill. These findings highlight that URSA is capable not only of discovering diverse behaviors, but also of learning skill representations that are suitable for real-world, target-driven control.

6 Related Work

Unsupervised skill discovery Recent methods aim to discover diverse behaviors without explicit rewards. Unsupervised RL approaches [23, 24, 25, 26] aim at maximizing the mutual information between abstract skills and trajectories to induce distinct behaviors, while SMERL [27] and DOMiNO [28] improve robustness by learning multiple solutions per task. In the Quality-Diversity (QD) domain, methods such as TAXONS [29], STAX [30], and AURORA [10] learn behavioral descriptors directly from data, removing the need for hand-crafted behavioral descriptors. However, these approaches heavily rely on simulation and face challenges in real-world applications. Our work builds on these methods by introducing safety constraints and efficient skill sampling for real-world learning, as well as imagination-based training to reduce physical interactions.

Real-world robot learning Real-world learning faces challenges like including limited data, safety concerns, and the lack of reset mechanisms. Reset-free QD [31, 7] selects behaviors that self-reset. Laversanne-Finot et al. [32] demonstrate diverse skill acquisition in robotic arms via goal-directed exploration, but still rely on episodic resets. off-DADS [6] enables skill discovery on real quadrupeds with off-policy RL, but does not consider any extrinsic reward. DayDreamer [14] sets a standard for sample-efficient learning using world models, and similarly, SERL [1], A Walk in the Park [2] and CrossQ [33] improve sample efficiency through careful system and algorithmic innovations. Our approach, like DayDreamer, leverages world models for unsupervised multi-skill discovery, adding safety constraints, and sophisticated skill selection mechanisms.

7 Discussion and Future Work

In this work, we introduce URSA, showcasing its ability to perform safe and efficient unsupervised QD in the real world. We show that URSA successfully adapts to damage, leveraging its learned repertoire for resilience across diverse real-world damage scenarios. Although our results are promising, several challenges remain. One such challenge is the potential benefit of learning safety boundaries directly from data, rather than relying on predefined constraints. This approach could enhance the system's robustness in dynamic and changing environments. Furthermore, developing more advanced sampling strategies that focus on the agent's zone of proximal development—where the potential for learning progress is maximized—could improve the efficiency of skill acquisition. These advancements would represent steps toward more autonomous and adaptable robotic systems capable of continuously learning and evolving in real-world conditions.

8 Limitations

Although our approach demonstrates promising results, there are several important limitations to address. While URSA uses a skill-conditioned policy network that enables the discovery of diverse skills, some discovered skills naturally share similar gaits that rely on specific limbs, making them more vulnerable to targeted damage scenarios affecting those particular limbs. This architectural choice, while beneficial for training efficiency, can lead to performance degradation in specific damage conditions.

Additionally, URSA's performance is critically dependent on the selection of input state variables provided to the VAE, which requires careful engineering and domain knowledge. The choice of which state dimensions to include or exclude can significantly impact downstream task performance, particularly in damage adaptation scenarios. For instance, omitting rear leg joint information would likely result in less diverse rear leg behaviors and degraded performance when rear legs are damaged. However, predicting the optimal set of input variables remains challenging, as the relationship between VAE input features and the policy's capability to solve specific tasks is often difficult to establish [34].

Our method also lacks an explicit mechanism, such as a dedicated reward function, to actively encourage the discovery of diverse skills. Instead, the system is limited to exploring skills that have been previously encountered during training, potentially missing novel and useful behaviors.

Moreover, our implementation is not fully reset-free. When the robot moves outside the designated arena, human intervention is still required to return it to the training area. This limitation could potentially be addressed by developing or providing a simple return-to-arena policy [35], enabling the robot to autonomously reset its position when needed.

In tasks involving forward movement, our current reward function does not sufficiently penalize lateral and angular velocities. This results in behaviors that deviate from straight-line motion, indicating a need to refine the reward structure to better align with desired movement patterns.

Finally, while our system trains to maximize rewards across all discovered skills, it does not discriminate between high-potential skills, with effective forward movement, and less useful ones, with minimal movement (see Fig. 1). Future work could explore methods to prioritize the development and diversification of more promising skills, potentially leading to a more efficient and practical skill repertoire.

References

[1] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine. Serl: A software suite for sample-efficient robotic reinforcement learning. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16961–16969, 2024. URL https://api.semanticscholar.org/CorpusID:267311834.

[2] I. Kostrikov, L. M. Smith, and S. Levine. Demonstrating A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. In K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi:10.15607/RSS.2023.XIX.056. URL https://doi.org/10.15607/RSS.2023.XIX.056.

[3] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, May 2015. ISSN 1476-4687. doi:10.1038/nature14422. URL https://doi.org/10.1038/nature14422.

[4] K. Chatzilygeroudis, V. Vassiliades, and J.-B. Mouret. Reset-free trial-and-error learning for robot damage recovery. *Robotics and Autonomous Systems*, 100:236–250, 2018.

[5] R. Kaushik, P. Desreumaux, and J.-B. Mouret. Adaptive prior selection for repertoire-based online adaptation in robotics. *Frontiers in Robotics and AI*, 6:151, 2020.

[6] A. Sharma, M. Ahn, S. Levine, V. Kumar, K. Hausman, and S. Gu. Emergent Real-World Robotic Skills via Unsupervised Off-Policy Reinforcement Learning. In *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation, July 2020. ISBN 978-0-9923747-6-1. doi:10.15607/RSS.2020.XVI.053. URL http://www.roboticsproceedings.org/rss16/p053.pdf.

[7] S. C. Smith, B. Lim, H. Janmohamed, and A. Cully. Quality-diversity optimisation on a physical robot through dynamics-aware and reset-free learning. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, page 171–174, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701207. doi:10.1145/3583133.3590625. URL https://doi.org/10.1145/3583133.3590625.

[8] J. K. Pugh, L. B. Soros, and K. O. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

[9] A. Cully and Y. Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2017.

[10] A. Cully. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 81–89, 2019.

[11] L. Grillotti and A. Cully. Relevance-guided unsupervised discovery of abilities with quality-diversity algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 77–85, 2022.

[12] L. Coiffard, P. Templier, and A. Cully. *Overcoming Deceptiveness in Fitness Optimization with Unsupervised Quality-Diversity*, page 122–130. Association for Computing Machinery, New York, NY, USA, 2025. ISBN 9798400714658. URL https://doi.org/10.1145/3712256.3726314.

[13] L. Grillotti, M. Faldor, B. González León, and A. Cully. Quality-diversity actor-critic: Learning high-performing and diverse behaviors via value and successor features critics. In *International Conference on Machine Learning*. PMLR, 2024.

[14] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel. DayDreamer: World Models for Physical Robot Learning, June 2022. URL http://arxiv.org/abs/2206.14176. arXiv:2206.14176 [cs].

[15] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018. ISBN 978-0-262-03924-6.

[16] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. van Hasselt, and D. Silver. Successor features for transfer in reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 4058–4068, Red Hook, NY, USA, Dec. 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4.

[17] E. Altman. *Constrained Markov decision processes*. Routledge, 1999.

[18] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[19] M. Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837, 1956. doi:10.1214/aoms/1177728190. URL https://doi.org/10.1214/aoms/1177728190.

[20] D. W. Scott. *Multivariate Density Estimation: Theory, Practice and Visualization*. John Wiley & Sons, Inc., New York, 1992.

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

[22] J. Cheng, M. Vlastelica, P. Kolev, C. Li, and G. Martius. Learning diverse skills for local navigation under multi-constraint optimality. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5083–5089. IEEE, 2024.

[23] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

[24] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgLZR4KvH.

[25] S. Park, K. Lee, Y. Lee, and P. Abbeel. Controllability-aware unsupervised skill discovery. *arXiv preprint arXiv:2302.05103*, 2023.

[26] S. Park, O. Rybkin, and S. Levine. Metra: Scalable unsupervised rl with metric-aware abstraction. *arXiv preprint arXiv:2310.08887*, 2023.

[27] S. Kumar, A. Kumar, S. Levine, and C. Finn. One solution is not all you need: few-shot extrapolation via structured MaxEnt RL. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, pages 8198–8210, Red Hook, NY, USA, Dec. 2020. Curran Associates Inc. ISBN 978-1-71382-954-6.

[28] T. Zahavy, Y. Schroecker, F. Behbahani, K. Baumli, S. Flennerhag, S. Hou, and S. Singh. Discovering policies with domino: Diversity optimization maintaining near optimality. *arXiv preprint arXiv:2205.13521*, 2022.

[29] G. Paolo, A. Laflaquiere, A. Coninx, and S. Doncieux. Unsupervised Learning and Exploration of Reachable Outcome Space. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2379–2385, Paris, France, May 2020. IEEE. ISBN 978-1-72817-395-5. doi:10.1109/ICRA40945.2020.9196819. URL https://ieeexplore.ieee.org/document/9196819/.

[30] G. Paolo, M. Coninx, A. Laflaquière, and S. Doncieux. Discovering and Exploiting Sparse Rewards in a Learned Behavior Space. *Evolutionary Computation*, 32(3):275–305, Sept. 2024. ISSN 1063-6560. doi:10.1162/evco_a_00343. URL https://doi.org/10.1162/evco_a_00343.

[31] B. Lim, A. Reichenbach, and A. Cully. Learning to walk autonomously via reset-free quality-diversity. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, pages 86–94, New York, NY, USA, July 2022. Association for Computing Machinery. ISBN 978-1-4503-9237-2. doi:10.1145/3512290.3528715. URL https://doi.org/10.1145/3512290.3528715.

[32] A. Laversanne-Finot, A. Péré, and P.-Y. Oudeyer. Intrinsically motivated exploration of learned goal spaces. *Frontiers in neurorobotics*, 14:555271, 2021.

[33] N. Bohlinger, J. Kinzel, D. Palenicek, L. Antczak, and J. Peters. Gait in eight: Efficient on-robot learning for omnidirectional quadruped locomotion. *arXiv preprint arXiv:2503.08375*, 2025.

[34] D. Tarapore, J. Clune, A. Cully, and J.-B. Mouret. How do different encodings influence the performance of the map-elites algorithm? In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 173–180, 2016.

[35] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan. Learning to walk in the real world with minimal human effort. In J. Kober, F. Ramos, and C. Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 1110–1120. PMLR, 16–18 Nov 2021. URL https://proceedings.mlr.press/v155/ha21c.html.

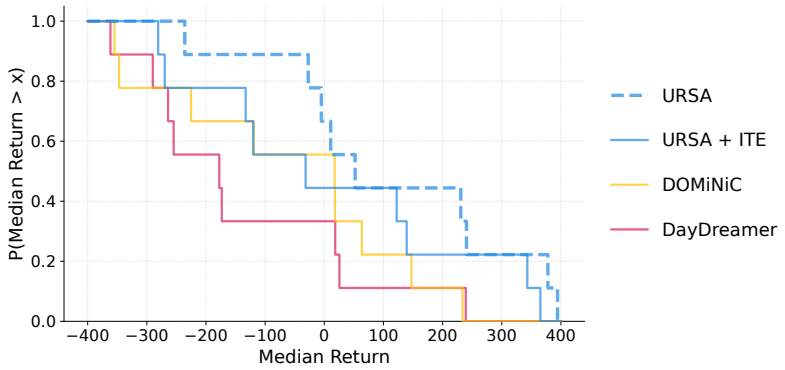

Figure 7: Complementary cumulative distribution functions (CCDF) of the median performance across the simulated damage scenarios from Figure 4.

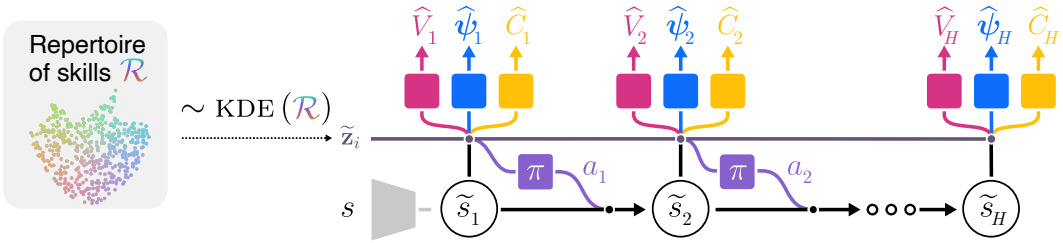

Figure 8: Illustration of URSA's imagination-based training loop in the world model $\mathcal{W}$. Given a sampled skill $\boldsymbol{z}$, the world model generates imagined trajectories used to update networks parameterizing the value function V, successor features $\boldsymbol{\psi}$, cost function C, and the policy π.

A Supplementary Analysis

URSA does not consistently outperform all baselines across individual damage scenarios (Fig. 4). Nevertheless, URSA and URSA + ITE exhibit less performance degradation overall, indicating greater robustness. This is illustrated by the complementary cumulative distribution functions (CCDF) of the median performance across our 9 damage scenarios considered (Fig. 7).

The seemingly weaker results from Figure 4 stem partly from URSA's network architecture. As URSA uses a skill-conditioned policy network, some discovered skills naturally share similar gaits that rely on specific limbs, making them more vulnerable to targeted damage scenarios.

B Method Details

Here we provide further details on our optimization objective (Problem P3), skill sampling strategy (Section 4.2), and adaptive thresholding mechanism (Section 4.2).

B.1 Lagrangian Optimization for Safe and Targeted Skill Execution

We solve the constrained optimization problem in Eq. P3 using min-max Lagrangian optimization:

$$\max_{\pi} \min_{\lambda_1,\lambda_2 \geq 0} V(\boldsymbol{s},\boldsymbol{z}) - \lambda_1(\|(1-\gamma)\boldsymbol{\psi}(\boldsymbol{s},\boldsymbol{z}) - \boldsymbol{z}\|_2 - \delta) - \lambda_2 C(\boldsymbol{s},\boldsymbol{z})$$

where λ_1 and λ_2 are the Lagrange multipliers associated with the distance to skill and safety constraints, respectively.

Algorithm 3 URSA – Repertoire Update

input state $\boldsymbol{s}_t$, current repertoire $\mathcal{R}$, maximum size $N_{\mathcal{R}}$
output Updated repertoire $\mathcal{R}$

$\boldsymbol{\phi}_t = \boldsymbol{\phi}(\boldsymbol{s}_t)$

if $\boldsymbol{\phi}_t \notin \mathcal{S}_{\text{safe}}$ **then**
  return ▷ Never add unsafe features to skill repertoire
else if $|\mathcal{R}| < N_{\mathcal{R}}$ **then**
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\boldsymbol{\phi}_t\}$ ▷ Add to repertoire if it is not full yet
else
  $\mathcal{R}' \leftarrow \mathcal{R} \cup \{\boldsymbol{\phi}_t\}$ ▷ Create temporary extended repertoire
  for each skill $\boldsymbol{z} \in \mathcal{R}'$ **do**
    $d_{\boldsymbol{z}} \leftarrow \min_{\boldsymbol{z}' \in \mathcal{R}' \setminus \{\boldsymbol{z}\}} d_{\boldsymbol{\Sigma}}(\boldsymbol{z}, \boldsymbol{z}')$ ▷ Compute nearest neighbor distance for each skill
  $\boldsymbol{z}_{\text{remove}} \leftarrow \arg\min_{\boldsymbol{z} \in \mathcal{R}'} d_{\boldsymbol{z}}$ ▷ Find skill with smallest nearest neighbor distance
  $\mathcal{R} \leftarrow \mathcal{R}' \setminus \{\boldsymbol{z}_{\text{remove}}\}$ ▷ Remove to improve diversity and maintain $N_{\mathcal{R}}$ elements

In practice, we found that optimizing the following smoothed actor objective with clipped multipliers $0 \leq \lambda_1, \lambda_2 \leq 1$ improves stability:

$$J_\pi = (1-\lambda_1)(1-\lambda_2)V(\boldsymbol{s}, \boldsymbol{z}) - \lambda_1(1-\lambda_2)(\|(1-\gamma)\boldsymbol{\psi}(\boldsymbol{s}, \boldsymbol{z}) - \boldsymbol{z}\|_2 - \delta) - \lambda_2 C(\boldsymbol{s}, \boldsymbol{z})$$

The Lagrange multipliers λ_1 and λ_2 are parameterized as neural networks that take $(\boldsymbol{s}, \boldsymbol{z})$ as input. This allows dynamic trade-offs across states and skills. As λ_2 increases, the agent prioritizes safety, suppressing both reward maximization and skill execution. Once safety constraints are reliably satisfied (i.e., λ_2 is low), the agent shifts focus to skill tracking by increasing λ_1. When both constraints are met, the agent emphasizes reward maximization. During training, they are updated via gradient descent.

B.2 Kernel Density Estimation Sampling

Initialization Initially, the repertoire $\mathcal{R}$ is empty, so we sample skills from a standard normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. These initial random skills do not compromise learning since sampled skills are only used for action selection and policy updates, and are never stored in the dataset $\mathcal{D}$. Once the repertoire contains enough skills for a non-degenerate covariance matrix, we transition to sampling from the KDE fitted on $\mathcal{R}$.

KDE Sampling To sample a skill $\boldsymbol{z}_{\text{target}}$ from the KDE distribution $q = \text{KDE}(\mathcal{R}) = \frac{1}{N_{\mathcal{R}}} \sum_{i=1}^{N_{\mathcal{R}}} \mathcal{N}(\boldsymbol{z}_i, \boldsymbol{\Sigma})$, we use the standard mixture sampling procedure:

1. **Component Selection:** Uniformly select an index $i \sim \text{Uniform}(1, \ldots, N_{\mathcal{R}})$ from the repertoire.
2. **Gaussian Sampling:** Sample $\boldsymbol{z}_{\text{target}} \sim \mathcal{N}(\boldsymbol{z}_i, \boldsymbol{\Sigma})$ using the selected repertoire skill $\boldsymbol{z}_i$ as the mean.

The covariance matrix $\boldsymbol{\Sigma}$ is the empirical covariance of all skills in the repertoire scaled by Scott's rule bandwidth $h = N_{\mathcal{R}}^{-\frac{1}{D+4}}$, such that the final covariance is $\boldsymbol{\Sigma} = h^2 \boldsymbol{\Sigma}_{\text{empirical}}$. This sampling procedure naturally concentrates samples around regions of high skill density while allowing exploration of nearby skill variations through the Gaussian kernels.

B.3 Repertoire Update

Algorithm 3 details the repertoire update mechanism from Section 4.2.

B.4 Imagination-Based Skill Learning

Figure 8 illustrates the imagination-based training loop. The world model generates imagined rollouts conditioned on a sampled skill $\boldsymbol{z}$, from an initial state $\boldsymbol{s}$, over a fixed horizon H. These trajectories are used to train the value function V, successor features $\boldsymbol{\psi}$, cost critic C, and policy π, enabling efficient skill acquisition.

B.5 Adaptive Threshold Computation

To avoid satisfying multiple constraints with a single skill, we define an adaptive constraint threshold δ that scales with the repertoire's density and the target number of distinct skills $N_{\boldsymbol{z}}$:

$$\delta = \left(\frac{N_{\mathcal{R}}}{N_{\boldsymbol{z}}}\right)^{1/D} \cdot \frac{1}{2N_{\mathcal{R}}} \sum_{i=1}^{N_{\mathcal{R}}} d_{\boldsymbol{\Sigma}}(\boldsymbol{z}_i, \boldsymbol{z}_i^{\mathrm{nn}})$$

Here, $\boldsymbol{z}_i^{\mathrm{nn}}$ is the nearest neighbor of $\boldsymbol{z}_i$ under the Mahalanobis distance. This formulation ensures that the learned skills are well-separated in the latent space.

C Mathematical Derivations

We provide here the technical details behind the lower bound on the approximate entropy of the KDE:

$$\begin{aligned}
\widehat{\mathrm{H}}(\mathrm{KDE}\,(\mathcal{R})) &= -\frac{1}{N_{\mathcal{R}}} \sum_{i=1}^{N_{\mathcal{R}}} \log \left(\frac{1}{N_{\mathcal{R}}} \sum_{j=1}^{N_{\mathcal{R}}} \mathcal{N}(\boldsymbol{z}_i | \boldsymbol{z}_j, \boldsymbol{\Sigma}) \right) \\
&\geq -\frac{1}{N_{\mathcal{R}}} \sum_{i=1}^{N_{\mathcal{R}}} \log \left(1 + (N_{\mathcal{R}} - 1) e^{-\frac{1}{2} d_{\boldsymbol{\Sigma}}(\boldsymbol{z}_i, \boldsymbol{z}_i^{\mathrm{nn}})^2} \right) + \frac{1}{2} \log \left(\det(\boldsymbol{\Sigma}) \right) + \frac{D}{2} \log(2\pi) + \log N_{\mathcal{R}}
\end{aligned}$$

where $d_{\boldsymbol{\Sigma}}(\cdot, \cdot)$ is the Mahalanobis distance with respect to the covariance matrix $\boldsymbol{\Sigma}$.

Proof. For all $i \in \{1, 2, \ldots, N_{\mathcal{R}}\}$:

$$\begin{aligned}
\log \sum_{j=1}^{N_{\mathcal{R}}} \mathcal{N}(\boldsymbol{z}_i | \boldsymbol{z}_j, \boldsymbol{\Sigma}) &= \log \left(\frac{1}{\sqrt{(2\pi)^D \det(\boldsymbol{\Sigma})}} \sum_{j=1}^{N_{\mathcal{R}}} \exp \left(-\frac{1}{2} (\boldsymbol{z}_i - \boldsymbol{z}_j)^\intercal \boldsymbol{\Sigma}^{-1} (\boldsymbol{z}_i - \boldsymbol{z}_j) \right) \right) \\
&= \log \left(\sum_{j=1}^{N_{\mathcal{R}}} \exp \left(-\frac{1}{2} (\boldsymbol{z}_i - \boldsymbol{z}_j)^\intercal \boldsymbol{\Sigma}^{-1} (\boldsymbol{z}_i - \boldsymbol{z}_j) \right) \right) - \frac{1}{2} \log \left(\det(\boldsymbol{\Sigma}) \right) - \frac{D}{2} \log(2\pi) \\
&= \log \left(\sum_{j=1}^{N_{\mathcal{R}}} \exp \left(-\frac{1}{2} d_{\boldsymbol{\Sigma}}(\boldsymbol{z}_i, \boldsymbol{z}_j)^2 \right) \right) - \frac{1}{2} \log \left(\det(\boldsymbol{\Sigma}) \right) - \frac{D}{2} \log(2\pi) \\
&= \mathrm{LSE} \left(-\frac{1}{2} d_{\boldsymbol{\Sigma}}(\boldsymbol{z}_i, \boldsymbol{z}_1)^2, \ldots, -\frac{1}{2} d_{\boldsymbol{\Sigma}}(\boldsymbol{z}_i, \boldsymbol{z}_{N_{\mathcal{R}}})^2 \right) - \frac{1}{2} \log \left(\det(\boldsymbol{\Sigma}) \right) - \frac{D}{2} \log(2\pi) \\
&\leq \log \left(1 + (N_{\mathcal{R}} - 1) \exp \left(-\frac{1}{2} d_{\boldsymbol{\Sigma}}(\boldsymbol{z}_i, \boldsymbol{z}_i^{\mathrm{nn}})^2 \right) \right) - \frac{1}{2} \log \left(\det(\boldsymbol{\Sigma}) \right) - \frac{D}{2} \log(2\pi)
\end{aligned}$$

Then we have:

$$\begin{aligned}
\widehat{\mathrm{H}}(\mathrm{KDE}\,(\mathcal{R})) &= -\frac{1}{N_{\mathcal{R}}} \sum_{i=1}^{N_{\mathcal{R}}} \left(\log \sum_{j=1}^{N_{\mathcal{R}}} \mathcal{N}(\boldsymbol{z}_i | \boldsymbol{z}_j, \boldsymbol{\Sigma}) \right) + \log N_{\mathcal{R}} \\
&\geq -\frac{1}{N_{\mathcal{R}}} \sum_{i=1}^{N_{\mathcal{R}}} \log \left(1 + (N_{\mathcal{R}} - 1) e^{-\frac{1}{2} d_{\boldsymbol{\Sigma}}(\boldsymbol{z}_i, \boldsymbol{z}_i^{\mathrm{nn}})^2} \right) + \frac{1}{2} \log \left(\det(\boldsymbol{\Sigma}) \right) + \frac{D}{2} \log(2\pi) + \log N_{\mathcal{R}}
\end{aligned}$$

□

D Training Details

This section provides implementation details for the reward, feature, and cost functions used in both unsupervised and velocity-conditioned experiments, as referenced in Section 5. We also report details on the URSA + ITE implementation used for damage adaptation tasks in Section 5.3. Additional hyperparameters are listed in Table 1.

D.1 Unsupervised Quality-Diversity

For the unsupervised setting, our objective is to learn a repertoire of diverse behaviors for forward locomotion. The feature function $\phi(\cdot)$ is implemented as a VAE encoder network that maps the robot's joint angles and torso height at each timestep t into a 2D latent skill space $\mathcal{Z}$.

The reward function promotes forward movement while ensuring stability:

$$r(s_t, a_t) = r_{\text{upr}} + \mathbb{1}_{r_{\text{upr}}>0.7} \cdot (5r_{\text{vel}_x} - 0.5r_{\text{vel}_y} - 0.5r_{\text{yaw}}) - 0.001(r_{\text{speed}} + r_{\text{work}} + r_{\text{smooth}})$$

Here, r_{upr} encourages upright posture, while forward velocity r_{vel_x} is rewarded. Lateral and yaw motion $(r_{\text{vel}_y}, r_{\text{yaw}})$ are penalized to promote forward-aligned behavior. Smoothness penalties $(r_{\text{speed}}, r_{\text{work}}, r_{\text{smooth}})$ encourage efficient movement. Following Wu et al. [14], velocity-based rewards are only active when the robot is upright ($r_{\text{upr}} > 0.7$).

The safe state set includes all states with $r_{\text{upr}} > 0.7$, and the cost function is defined as:

$$c(s_t, a_t) = 0.7 - r_{\text{upr}}(s_t, a_t)$$

This penalizes instability and unsafe motion when the robot deviates from an upright posture. We aim to learn a repertoire of $N_{\boldsymbol{z}} = 512$ distinct skills in this setting.

D.2 ITE for Damage Adaptation

In this section, we provide the implementation details of the iterative trial and error (ITE) algorithm, adapted for use with URSA's skill repertoire. The approach is based on the methodology introduced in Cully et al. [3].

The ITE process is run for 8 iterations to match the number of skills in the DOMiNiC baseline. We set the length scale parameter to 0.1 in the Gaussian process model to control the smoothness of the predictions and the noise level to 1×10^{-3} to account for observation noise during optimization.

We build a behavior map that serves as our prior, constructed by evaluating value function returns across 512 rollouts in our repertoire. This matches the number of distinct skills used during training.

For the kernel used in the Gaussian process, we employ a Matern52 kernel, which has proven effective in many real-world scenarios due to its flexibility and smoothness properties. In terms of the acquisition function, we adopt the Upper Confidence Bound (UCB) approach, following the original implementation in [3]. This acquisition function balances exploration and exploitation by selecting the point with the highest expected return, considering both the predicted mean and uncertainty of the model.

These steps ensure that the ITE algorithm efficiently adapts to damage while leveraging the learned skill repertoire for robust performance in real-world environments.

D.3 Velocity-Conditioned Skill Discovery

For the heuristic-based skill discovery setting, the reward function encourages smooth and well-postured motion:

$$r(s_t, a_t) = r_{\text{upr}} + r_{\text{hip}} + r_{\text{upper}} + r_{\text{lower}} - 0.001(r_{\text{speed}} + r_{\text{work}} + r_{\text{smooth}})$$

Each term encourages proper posture at a different joint level (torso, hip, upper leg, lower leg), activated sequentially through dependency on earlier posture terms (e.g., r_{hip} activates only if $r_{\text{upr}} > 0.7$). This encourages the robot to build stable poses from the ground up.

Safe states are defined by the conditions:

$$r_{\text{upr}},\ r_{\text{hip}},\ r_{\text{upper}},\ r_{\text{lower}} > 0.7$$

The cost function penalizes deviations below this threshold:

$$c(s_t, a_t) = \sum_{j \in \{\text{upr,hip,upper,lower}\}} \max(0.7 - r_j,\ 0)$$

D.4 Hyperparameters

A full list of rollout parameters and training constants used in our experiments is provided in Table 1.

Table 1: Hyperparameters

Parameter	Value
Imagination batch size N	1024
Real environment exploration batch size	32
Total timesteps (in simulation)	1×10^6
Optimizer	Adam
Learning rate	1×10^{-4}
Replay buffer size	1×10^6
Discount factor γ	0.995
Imagination horizon H	15
Target smoothing coefficient τ	0.02
Sampling period T	250
Lambda Return λ	0.95
Repertoire size $N_{\mathcal{R}}$	4096
Target distinct skills $N_{\boldsymbol{z}}$	512
Control frequency	20 Hz
DOMiNiC [22]	
Number of skills n_{dominic}	8
Near-optimality constraint	$\geq 0.9 \times$ value optimal policy
Target distance between skills l_0	1.0