Divide, Discover, Deploy: Factorized Skill Learning with Symmetry and Style Priors

Rafael Cathomen
ETH Zurich
carafael@ethz.ch

Mayank Mittal
ETH Zurich & NVIDIA
mittalma@ethz.ch

Marin Vlastelica
ETH Zurich
mvlastelica@ethz.ch

Marco Hutter
ETH Zurich
mahutter@ethz.ch

Abstract:
Unsupervised Skill Discovery (USD) allows agents to autonomously learn diverse behaviors without task-specific rewards. While recent USD methods have shown promise, their application to real-world robotics remains underexplored. In this paper, we propose a modular USD framework to address the challenges in the safety, interpretability, and deployability of the learned skills. Our approach employs user-defined factorization of the state space to learn disentangled skill representations. It assigns different skill discovery algorithms to each factor based on the desired intrinsic reward function. To encourage structured morphology-aware skills, we introduce symmetry-based inductive biases tailored to individual factors. We also incorporate a style factor and regularization penalties to promote safe and robust behaviors. We evaluate our framework in simulation using a quadrupedal robot and demonstrate zero-shot transfer of the learned skills to real hardware. Our results show that factorization and symmetry lead to the discovery of structured human-interpretable behaviors, while the style factor and penalties enhance safety and diversity. Additionally, we show that the learned skills can be used for downstream tasks and perform on par with oracle policies trained with hand-crafted rewards. For code and videos, please check: https://leggedrobotics.github.io/d3-skill-discovery/.

Keywords: unsupervised skill discovery, reinforcement learning, legged robots

1 Introduction

Reinforcement learning (RL) has achieved remarkable success across a range of real-world robotics applications [1, 2, 3]. However, these successes typically depend on carefully prespecified reward functions. Designing such rewards for a large number of tasks demands significant engineering effort and often becomes increasingly complex as task difficulty grows. Unsupervised Skill Discovery (USD) seeks to address these challenges by training agents to autonomously acquire a diverse repertoire of behaviors, or *skills*, without relying on handcrafted rewards. These skills can then be reused or fine-tuned to solve downstream tasks more efficiently.

Current USD approaches use an intrinsic reward function to generate training signals to acquire task-agnostic behaviors. These intrinsic rewards are typically formulated as different variants of mutual information (MI) between the agent's state $\boldsymbol{s}$ and its latent skill representation $\boldsymbol{z}$. For instance, DIAYN [4] optimizes a variational lower bound on the MI, while METRA [5] uses a Wasserstein variant of the MI. Furthermore, Hu et al. [6] show that factorizing $(\boldsymbol{s}, \boldsymbol{z})$ facilitates more interpretable and controllable skills. Despite these advances, most USD research remains confined to simulation, with limited demonstrations on real-world robotic systems.

A core limitation in USD lies in the exclusive reliance on intrinsic rewards: while these encourage exploration and behavioral diversity, they offer no feedback on whether the learned behaviors are safe, stable, or physically feasible on real hardware. As a result, the behaviors learned by many USD approaches tend to be overly aggressive or unsafe. Although some works [7, 8] have demonstrated

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

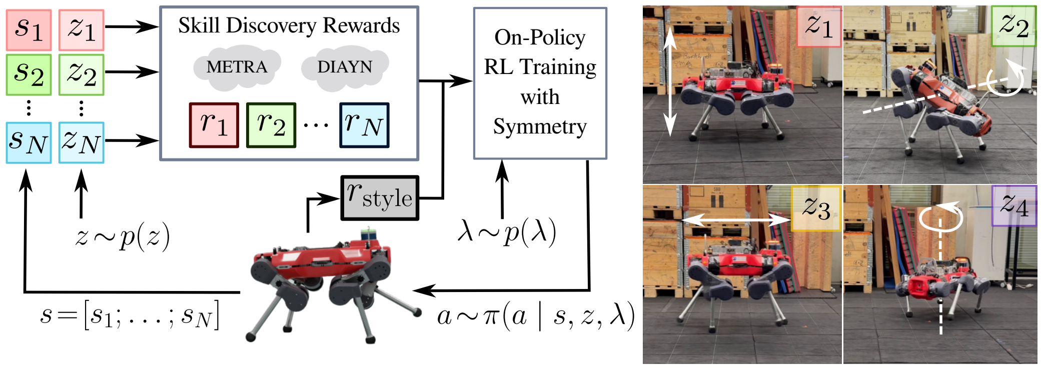


Figure 1: **Approach overview.** The agent's state $\boldsymbol{s}$ is factorized by the user into N components, each paired with a latent skill z_i and an intrinsic reward r_i, selected from METRA or DIAYN objectives. An extrinsic reward r_{style} promotes safe behaviors. The factor weights λ allow the agent to prioritize certain factors during training. The policy π is trained using on-policy RL with symmetry augmentation to discover structured, morphology-aware skills. The resulting skills are interpretable, robust, and can be commanded by a human operator.

unsupervised skill deployment on hardware, they usually focus on constrained or simplified scenarios. Efforts to improve safety in USD [9, 10] have also made progress, but often trade off between safety, skill diversity, and scalability. Overcoming these challenges is essential for advancing USD from an exploratory paradigm to a practical tool for developing robotic systems.

In this work, we present a factorized skill discovery framework that selectively applies USD algorithms across different state dimensions defined by the user (Fig. 1). The core idea is that the desired form of diversity often depends on the specific subset of the state space and the chosen USD algorithm. For instance, in our experiments, we observe that METRA excels at improving the state coverage on unbounded state factors such as planar position, while DIAYN is better suited for bounded state factors, such as the robot's orientation, where continuous drift is impossible. Our design effectively takes advantage of these individual benefits while also exploiting the symmetry in the robot's morphology. By extending this symmetry to the skill space, the framework encourages the discovery of more structured skills. To address the critical issue of deployability, we propose two additional mechanisms. First, we introduce an additional *style* factor, which is an extrinsic signal that shapes the agent's behavior toward safe and stable actions. Second, we develop a skill weighting mechanism that facilitates the handling of conflicting skills and allows their balanced adjustment during deployment. Using this framework, we demonstrate the discovery of diverse quadrupedal skills that are learned entirely in simulation and can be safely deployed on real hardware.

2 Related Work

Unsupervised Skill Discovery. The goal of USD is to extract task-agnostic behaviors from intrinsic rewards. Eysenbach et al. [4] maximize the lower bound on MI between skills and states via a learned discriminator, while Sharma et al. [11] add transition dynamics to encourage more kinetic skills. Optimistic exploration through discriminator ensembles [12] further enhances the state coverage. This is complementary to the problem of exploration, where the goal is to maximize coverage, often regularized by task reward [13, 14, 15, 16, 17], or maximize information gain [18]. An alternate line of work replaces the MI objective with a Wasserstein dependency measure (WDM). METRA [5] and its variants [19, 20, 21] maximize the directed distance in a learned latent space, resulting in highly dynamic state-covering skills. To increase interpretability of learned skills, DUSDi [6] factorizes the state and skill spaces and applies DIAYN per factor with an entanglement penalty. Subsequent work [22] extends this by using inter-factor dependency graphs to discover interaction-focused skills. Our proposed framework generalizes the factorization idea in DUSDi by allowing different USD objectives per state factor, letting each dimension exploit the most suitable notion of diversity. Additionally, we inject robot morphology-based symmetry priors [23, 24, 25] into the latent skills and introduce factor weights to coordinate potentially conflicting skills.

Deployment of USD. Most USD studies remain mainly in simulation; only a few consider real robots. Kim et al. [10] bias METRA with labeled desirable and undesirable trajectories, while Atanassov et al. [8] combine a norm-matching objective with hand-crafted rewards to transfer discovered locomotion skills to a quadruped. Cheng et al. [26] utilize DOMiNO [27] to learn diverse solutions for navigation tasks, while still relying on explicit task rewards. Further efforts [28, 29] have been made in constructing offline task-regularized USD algorithms by leveraging the Fenchel duality. Sharma et al. [7] show that basic locomotion skills can be learned directly on hardware with an off-policy version of DADS. However, the learned skills are relatively simple (planar locomotion) and contain undesirable motion artifacts. Unlike prior work relying on supervision or task rewards, we deploy intrinsically learned skills by combining a style factor, global regularization, and per-factor weighting, which balances safety with skill diversity.

3 Background

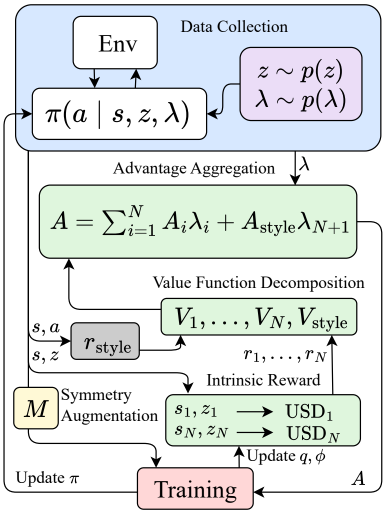
Symmetric Factored MDP. In this work, we consider a symmetric factored MDP. A factored MDP [6] is defined as the tuple $\mathcal{M}(\mathcal{S}, \mathcal{A}, \mathcal{T}, R)$, where the state space $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_N$ is factorized into N factors. Each state $\boldsymbol{s} \in \mathcal{S}$ consists of N state factors: $\boldsymbol{s} = [\boldsymbol{s}_1; \ldots; \boldsymbol{s}_N], \boldsymbol{s}_i \in \mathcal{S}_i$. The action space and transition kernel are denoted by $\mathcal{A}$ and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ respectively, where $\Delta(\cdot)$ denotes the probability simplex. The goal of USD is to learn a skill-conditioned policy $\pi_\theta : \mathcal{S} \times \mathcal{Z} \to \Delta(\mathcal{A})$ that results in diverse, useful, and distinguishable behaviors (*i.e.*, *skills*). This is typically achieved by maximizing a certain information objective, such as the mutual information (MI) between states and latent skills. Following the factored MDP, the skill space $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_N$ is also factorized, with the disentangled skill component $\boldsymbol{z}_i$ only affecting the state factor $\boldsymbol{s}_i$. The skills are sampled from a prior distribution $\boldsymbol{z} \sim p(\boldsymbol{z}) = \Pi_{i=1}^N p(\boldsymbol{z}_i)$. The MI objective results in a reward function $R : \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$, which can be maximized using standard RL [30, 31].

Intuitively, a symmetric MDP [32] means the dynamics and rewards are preserved under a set of transformations over the state and action spaces, such as a left-right reflection. An MDP has a K-fold symmetry if a set of K distinct transformations exist under which the transition dynamics is equivariant and the reward model is invariant. Extending this definition to USD, the transformations also need to be defined over the skill space. Let the functions $M_s^k : \mathcal{S} \to \mathcal{S}$, $M_a^k : \mathcal{A} \to \mathcal{A}$ and $M_z^k : \mathcal{Z} \to \mathcal{Z}$ define the k-th transformation functions for states, actions and skills, respectively. The MDP $\mathcal{M}$ is symmetric if $\forall k \in 1, \ldots, K$, $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{S}$, $\boldsymbol{a} \in \mathcal{A}$ and $z \in \mathcal{Z}$, the transition model $\mathcal{T}(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}) = \mathcal{T}(M_s^k(\boldsymbol{s}') \mid M_s^k(\boldsymbol{s}), M_a^k(\boldsymbol{a}))$ is equivariant, and the reward model $R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) = R(M_s^k(\boldsymbol{s}), M_a^k(\boldsymbol{a}), M_z^k(\boldsymbol{z}))$ and the skill prior $p(\boldsymbol{z}) = p(M_z^k(\boldsymbol{z}))$ are invariant. It is important to note that the mirror functions M_s^k and M_a^k are determined solely based on the transition model (*i.e.*, the robot's morphology), while the mirror function for the skills M_z^k must be defined in a way that the symmetry condition holds. These choices are discussed in Sec. 4.

MI-based USD Rewards. Our framework utilizes two algorithms: DIAYN [4] and METRA [5]. DIAYN maximizes MI, $I(\boldsymbol{S}; \boldsymbol{Z}) \triangleq D_{\mathrm{KL}}(p(\boldsymbol{s}, \boldsymbol{z}) \| p(\boldsymbol{s})p(\boldsymbol{z}))$, using a learned discriminator $q_\phi(\boldsymbol{z}|\boldsymbol{s})$ that approximates the posterior $p(\boldsymbol{z}|\boldsymbol{s})$, yielding the reward $r_{\mathrm{DIAYN}}(\boldsymbol{s}, \boldsymbol{z}) = \log q_\phi(\boldsymbol{z}|\boldsymbol{s}) - \log p(\boldsymbol{z})$. METRA replaces MI with WDM: $I_{\mathcal{W}}(\boldsymbol{S}; \boldsymbol{Z}) \triangleq \mathcal{W}(p(\boldsymbol{s}, \boldsymbol{z}), p(\boldsymbol{s})p(\boldsymbol{z}))$ under a temporal distance metric. It trains $\phi(\boldsymbol{s})$ and uses the reward $r_{\mathrm{METRA}}(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{s}') = (\phi(\boldsymbol{s}') - \phi(\boldsymbol{s}))^\top \boldsymbol{z}$ to align latent state transitions with the skill vector. Additional details are in App. A.1.

4 Method

Our approach builds on the idea of factorizing the latent skill vector $\boldsymbol{z}$ to create independent and disentangled skill components. Building on the notion of factored MDPs, we extend the framework from Hu et al. [6] to support different intrinsic objectives and include a *style* objective to promote deployable behaviors. This flexibility enables behavior-specific inductive biases by applying the most suitable USD algorithm per factor. To further enhance control and coordination between skill components, we introduce a scalar weight for each factor, which allows prioritizing specific components during training and effectively resolving conflicts between simultaneously learned skills.


Env
Data Collection
$z \sim p(z)$
$\lambda \sim p(\lambda)$
$\pi(a \mid s, z, \lambda)$
Advantage Aggregation
λ
$A = \sum_{i=1}^{N} A_i\lambda_i + A_{\text{style}}\lambda_{N+1}$
Value Function Decomposition
$V_1, \ldots, V_N, V_{\text{style}}$
s, a
r_{style}
s, z
$r_1, \ldots, r_N$
Intrinsic Reward
M
Symmetry Augmentation
$s_1, z_1 \longrightarrow \text{USD}_1$
$s_N, z_N \longrightarrow \text{USD}_N$
Update q, ϕ
Update π
Training
A

```
Algorithm 1: Skill Discovery Procedure
  Factorize state space $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_N$
  Factorize skill space $\mathcal{Z} = \mathcal{Z}_1 \times \cdots \times \mathcal{Z}_N$
  Define skill prior $p(\boldsymbol{z})$ and weighting prior $p(\lambda)$
  Define style reward $r_{\text{style}}$ and regularization reward $r_{\text{reg}}$
  Define symmetry mirroring functions $M = (M_s, M_a, M_z)$
  Select USD algorithm per factor $\text{USD}_i \in \{\text{METRA}, \text{DIAYN}\}$

  Initialize $\pi_\theta$, $\{V_i\}_{i=1}^{N}$, $V_{\text{style}}$
  while not converged do
      Sample skill $\boldsymbol{z} \sim p(\boldsymbol{z})$ and weights $\lambda \sim p(\lambda)$ every $k$ steps
      Sample action $\boldsymbol{a} \sim \pi_\theta(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z}, \lambda)$
      Collect on-policy samples $(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{a}, \boldsymbol{z}, \lambda, r)$
      Augment the dataset by mirroring each sample
      Update all $\text{USD}_i$ and $V_i$ with the augmented samples
      Compute advantages $A_i$ for each factorized value function
      Compute weighted sum of advantages $A = \sum_{i=1}^{N+1} \lambda_i A_i$
      Update $\pi_\theta$ based on $A$ using any on-policy RL algorithm
  end while
```

Figure 2: **Proposed algorithm for skill discovery.** The agent π_θ, conditioned on a sampled skill $\boldsymbol{z}$ and factor weights λ, collects transitions and receives a total reward combining per-factor intrinsic rewards and a style reward. The transitions are then augmented via symmetry-based mirroring, after which the intrinsic reward models, factorized value functions, and policy are updated using on-policy RL.

More formally, the objective for the skill-conditioned policy π_θ is to maximize

$$\mathcal{J}(\theta) = \sum_{i=1}^{N} \lambda_i I_{\text{USD}_i}(\boldsymbol{S}_i, \boldsymbol{Z}_i) + \lambda_{N+1} J_{\text{style}}(\boldsymbol{S}, \boldsymbol{A}), \tag{1}$$

where $\lambda = [\lambda_i]_{i=1}^{N+1}$ is the factor weighting vector, which assigns relative importance to individual objectives. The per-factor objective I_{USD_i} depends on the selected USD algorithm for that factor. In this work, we consider this objective based on DIAYN with disentanglement penalty [6, 4] or METRA [5]. The objective J_{style} includes extrinsic rewards for neutral skills, such as standing stationary. In the remainder of the section, we provide further details on these individual components.

Factor Weighting. When we disentangle the skill space through factorization, each skill dimension (or factor) is intended to control a distinct and ideally independent aspect of the agent's behavior. However, in practice, state dependencies between factors can lead to behavioral conflicts. For example, skill factors for standing still and moving forward cannot be executed simultaneously without interference. This issue leads to poorer coverage of the learned skill factors [6]. To manage potential conflicts between skill factors, we introduce *factor weights* $\lambda \in \mathbb{R}^{N+1}$, where each $\lambda_i \geq 0$ and $\|\lambda\|_2 = 1$. These weights modulate the relative importance of each intrinsic reward, arising from I_{USD_i}, as well as any extrinsic rewards. By conditioning the policy π_θ on λ, the agent can dynamically prioritize the skill factors. During training, we sample λ by normalizing a vector of i.i.d. positive values from a truncated Gaussian, ensuring the norm constraint is satisfied.

Style Factor and Regularization Penalties. To improve the deployability of unsupervised skills, prior work [8] incorporates two types of extrinsic rewards, one for smoothness and one for aesthetic behavior. However, applying both rewards uniformly can overconstrain skill discovery and limit diversity. To address this, we separate these signals. We introduce an additional factor that provides a neutral "style" reward that depends on the robot's configuration. For a quadruped, this reward may encourage the robot to maintain a stable posture, such as standing still. Treating this as a separate factor allows the agent to learn a safe fallback skill and benefit from a soft inductive bias during learning. Since the style factor is included in the policy input, its influence can be modulated dynamically via the weighting mechanism, enabling the agent to stay near safe behaviors when needed, while still exploring meaningfully under intrinsic objectives.

We apply global regularization penalties to reduce joint torques and velocities and enforce physical constraints such as joint limits. Unlike the style factor, these are applied uniformly across all skills and are not part of the policy input. They promote safety and support hardware deployment.

The resulting reward is defined as: $r(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) = \sum_{i=1}^{N} \lambda_i r_{\text{USD}_i}(\boldsymbol{s}_i, \boldsymbol{z}_i) + \lambda_{N+1} r_{\text{style}}(\boldsymbol{s}, \boldsymbol{a}) + r_{\text{reg}}(\boldsymbol{s}, \boldsymbol{a})$, where r_{style} and r_{reg} correspond to the two types of extrinsic rewards. In practice, because the magnitudes of the individual USD and style rewards can differ, we apply exponential moving average (EMA) normalization to them to stabilize the training. Following [22], we adopt value function decomposition, training a separate value function for each reward term. These are then combined using the factor weights to compute the overall advantage function, as explained in Fig. 2.

Symmetry Augmentation. Following Mittal et al. [25] and Sec. 3, we promote symmetry by augmenting the collected transitions by mirroring: $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}, r) \rightarrow \{(M_s^k(\boldsymbol{s}), M_a^k(\boldsymbol{a}), M_z^k(\boldsymbol{z}), r)\}_{k=1}^{K}$. Since the USD rewards r_{USD_i} are outputs of neural networks, they may not inherently respect the symmetry invariance. One way to enforce symmetry is by averaging the reward over mirrored samples: $\frac{1}{K} \sum_{k=1}^{K} r_{\text{USD}_i}(M_s^k(\boldsymbol{s}), M_a^k(\boldsymbol{a}), M_z^k(\boldsymbol{z}))$. However, in practice, we found that it suffices to train all networks (actor, critics, discriminators, and encoders) on symmetry-augmented data to induce approximate symmetry in both the learned reward signal and the agent's behavior.

The central challenge in combining USD with symmetry augmentation lies in defining M_z such that invariance in the prior distribution and composition are respected. For instance, if in the state space $M_s^1 = M_s^2 \circ M_s^3$ (where $\circ$ denotes function composition), then in the skill space M_z^1 should also be equal to $M_z^2 \circ M_z^3$. DIAYN priors are typically invariant to coordinate permutations, while METRA's isotropic priors are direction-invariant. Thus, the mirroring function M_z can be realized as coordinate permutations for DIAYN and optional sign flips for METRA. More details are in App. A.2.

As a concrete example of a permutation-based skill mirroring function, consider a factored MDP with K symmetries. For each state factor i, we assign a corresponding skill factor of dimension $\dim(\mathcal{Z}_i) = n \cdot K$, where $n \in \mathbb{N}^+$ is a hyperparameter. The factor skill is partitioned as $\boldsymbol{z}_i = [\boldsymbol{z}_{i,1}; \cdots; \boldsymbol{z}_{i,K}]$, with $\dim(\boldsymbol{z}_{i,k}) = n$. We define M_z as permuting these K sub-skills. A convenient choice is to let the permutations realize a Latin square [33], where every sub-skill cycles through every position exactly once in a way that respects composition. For a robotic agent with its four-fold symmetries, we can define the symmetry transformations as:

$$M_z^1(\boldsymbol{z}_i) = [\boldsymbol{z}_{i,1}; \boldsymbol{z}_{i,2}; \boldsymbol{z}_{i,3}; \boldsymbol{z}_{i,4}], \qquad M_z^2(\boldsymbol{z}_i) = [\boldsymbol{z}_{i,3}; \boldsymbol{z}_{i,4}; \boldsymbol{z}_{i,1}; \boldsymbol{z}_{i,2}],$$
$$M_z^3(\boldsymbol{z}_i) = [\boldsymbol{z}_{i,2}; \boldsymbol{z}_{i,1}; \boldsymbol{z}_{i,4}; \boldsymbol{z}_{i,3}], \qquad M_z^4(\boldsymbol{z}_i) = [\boldsymbol{z}_{i,4}; \boldsymbol{z}_{i,3}; \boldsymbol{z}_{i,2}; \boldsymbol{z}_{i,1}],$$

which leaves the prior distribution unchanged and satisfies the desired composition rule.

Importantly, in METRA, each skill $\boldsymbol{z}_i$ is interpreted as a direction in the learned projected state space. Shuffling coordinates or adding redundant dimensions undermines the geometric meaning and continuity of the skills. To preserve this structure, we use a low-dimensional representation (at most three coordinates $d \leq 3$) per factor and define the mirroring operation to match that of the corresponding state factor. This ensures that symmetry transformations in the skill space remain consistent with those in the state space, preserving the directional semantics critical to METRA.

Skill Prior and Curriculum. For factors trained with METRA, skills are initially sampled uniformly from the unit hypersphere $\boldsymbol{z}_i \sim \mathrm{U}(\mathbb{S}^{d-1})$. Training begins with the default METRA alignment objective and gradually transitions to the norm-matching objective proposed by Atanassov et al. [8]. The alignment objective provides a more interpretable and stable learning signal early on, facilitating initial skill acquisition. As training progresses, switching to the norm-matching objective increases the expressiveness of the skill space by allowing the skill norm to influence execution speed. During this transition, we update the skill prior by sampling variable-norm skills, enabling finer control over behavior dynamics. Additional details are in App. A.1. For DIAYN, we sample the skills from a symmetric Dirichlet prior $\boldsymbol{z}_i \sim \mathrm{Dir}(\alpha)$. To emulate the separation of a categorical latent while retaining continuity, we start with a sparse prior $(\alpha_k = 0.05)$, concentrating the probability mass on a single coordinate. When the discriminator's accuracy surpasses a preset threshold, every component is annealed linearly to 1.0, yielding a maximum-entropy Dirichlet (uniform over the $nK - 1$ dimensional probability simplex) and enabling smooth skill interpolation.

Skill Switching. Most USD methods fix the latent skill by sampling it once at the start of each episode. In contrast, we resample the skill multiple times within an episode. Without this resampling,

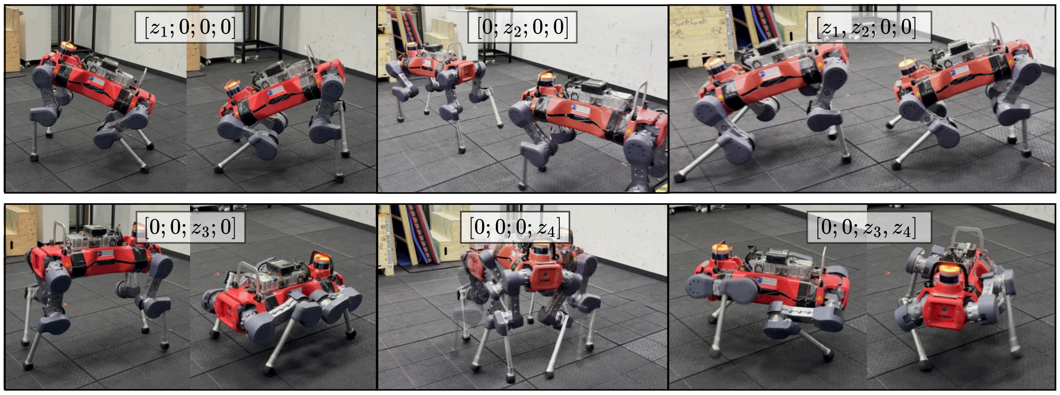


Figure 3: **Deployment of learned skills on the real robot**. The learned structured skill space enables intuitive and composable control. Each behavior corresponds to a manually commanded skill $\boldsymbol{z}$, set by adjusting individual skill factors $\boldsymbol{z}_i$. This results in diverse behaviors: pitching, walking, ducking, rotating and their combinations. Here we show walking while pitching, and ducking while rotating in the top and bottom rows respectively.

we observe that agents tend to ”lock in” to the initial skill, *i.e.*, even when the skill input is changed during deployment, the behavior remains unchanged. We hypothesize that this happens because the agent infers the skill from its state and, upon reaching a rewarding configuration, it chooses to stay there to maximize the USD reward. Resampling skills during training encourages the agent to remain responsive to the skill input, resulting in smoother skill switching at test time.

5 Experiments

We consider skill discovery for the quadrupedal robotic platform, ANYmal-D, which has four symmetries [25]. We train policies on a rough terrain environment with a difficulty-based curriculum that depends on state coverage. The state space is factorized into base position, linear velocity, heading rate, base height, and base roll and pitch. Depending on the evaluation setup, we use different subsets of these factors and assign varying algorithm combinations to evaluate their effects. All training is carried out in simulation using Isaac Lab [34] with 2048 environments. On an NVIDIA RTX 3090, the training converges within a day. For additional training details, please check App. A.3.

Deployment of Learned Skills. We demonstrate the structured nature of the learned skill space by manually commanding individual skill dimensions on the real robot. As shown in Fig. 3, each skill aligns with a specific state factor and can be composed intuitively. The framework captures symmetries in behavior, for example, forward and backward walking, or tilting in opposite directions, while the style factor enables stable behaviors like standing still. Combinations of skills, such as walking while pitching or rotating while crouched, further highlight the composability and expressiveness of the learned skill space.

Factor Weights. We evaluate the effect of per-factor weights λ in a setup with four DIAYN-trained factors (roll-pitch, heading, planar velocity, and height) and the style factor. Fig. 4 shows the skill discriminability and the scaled style reward. When using weights, each rollout contributes to the per-factor metrics proportionally to its assigned factor weights, normalized to avoid numerical bias. The weighted setup achieves substantially higher scores, showing that the agent learns to prioritize relevant factors. This effect is strongest with DIAYN; METRA-based variants showed smaller gains (App. A.5).

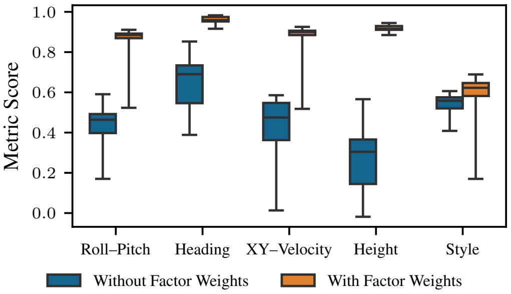


Figure 4: **Effect of factor weighting on skill metrics.** The metric score (details in App. A.4) reflects either discriminator classification accuracy (state factors) or style reward (style factor). Incorporating per-factor weights λ enables the agent to prioritize relevant factors, yielding consistently higher scores across all dimensions.

Table 1: **Effect of the style factor on skill metrics and safety.** The factor metrics report classification accuracy for DIAYN and cosine similarity for METRA (with mean ± std over 5 seeds). Illegal contacts show the percentage of illegal contacts per step for different body parts.

	Factor Metrics ↑			% Illegal Contacts per Step ↓		
Style	Extrinsic	Position	Heading	Base	Shank	Thigh
Without Style Factor	0.19 ± 0.11	0.27 ± 0.14	0.56 ± 0.04	0.46 ± 0.37	0.75 ± 0.23	4.04 ± 0.06
With Style Factor	**0.59 ± 0.04**	**0.68 ± 0.10**	**0.72 ± 0.04**	**0.00 ± 0.00**	**0.12 ± 0.07**	**0.03 ± 0.00**

Table 2: **Comparison against different USD approaches across state factors.** Diversity is measured as state coverage (details in App. A.4) with mean $\pm$ std over 5 seeds. Higher values indicate broader skill coverage.

	Algorithm Chosen per Factor		Diversity per Factor ↑	
Approach	Position	Heading	Position	Heading
DIAYN	DIAYN ($\dim(\boldsymbol{z}) = 8$)		0.389 ± 0.183	1.067 ± 0.532
METRA	METRA ($\dim(\boldsymbol{z}) = 3$)		9.832 ± 0.808	0.212 ± 0.018
DUSDi	DIAYN ($\dim(\boldsymbol{z}_1) = 4$)	DIAYN ($\dim(\boldsymbol{z}_2) = 2$)	1.363 ± 0.333	1.811 ± 0.084
2xMETRA	METRA ($\dim(\boldsymbol{z}_1) = 2$)	METRA ($\dim(\boldsymbol{z}_2) = 1$)	8.836 ± 1.411	0.271 ± 0.078
Mixed (Ours)	METRA ($\dim(\boldsymbol{z}_1) = 2$)	DIAYN ($\dim(\boldsymbol{z}_2) = 2$)	8.776 ± 0.667	1.031 ± 0.476

Safety and Extrinsic Rewards. To evaluate the impact of the style factor, we conduct experiments using METRA for base position and DIAYN for heading rate factor. Two sets of experiments are run, each with five different random seeds: one with the style factor active, and one with the style factor disabled by setting its weight to zero. From Tab. 1, we observe that the style factor significantly reduces undesirable contacts and improves discriminability for both position and heading. This suggests it promotes safer behaviors while regularizing skill learning toward more consistent, interpretable skills.

Comparing Different USD Objectives. To evaluate the flexibility and effect of assigning different USD algorithms to individual state factors, we compare diversity across various algorithm configurations. We factorize the state into base position and heading rate. In our setup, we use METRA for position, favoring broad state-space coverage, and DIAYN for heading, encouraging skill separability. We compare this mixed setup against several baselines: both factors trained with DIAYN (similar to DUSDi [6] but with a style factor and regularization, denoted as “DUSDi” in the tables), both with METRA (“2×METRA”), and single-objective baselines where the two factors are combined into one (i.e., no factorization) and trained with either DIAYN [4] or METRA [5], both with style and regularization. For evaluating diversity, we follow Zahavy et al. [27] and compute Monte Carlo estimates of successor representations for each skill and report their standard deviation in Tab. 2. Higher values indicate broader diversity in the corresponding factor. We observe that METRA significantly improves diversity in the position factor, while DIAYN achieves higher diversity for the heading factor. The mix of algorithms per factor outperforms using a single algorithm for all factors.

Symmetric Skill Discovery. To evaluate the effect of symmetry augmentation, we train policies with and without symmetry bias. We observe that symmetry augmentation does not result in faster convergence or higher evaluation metrics. Additionally, policies trained with symmetry augmentation often obtain lower metric scores for factors trained with METRA-style rewards and perform similarly with DIAYN-style rewards (see App. A.6). This could be due to METRA-based factors being harder to symmetrize effectively due to the geometric interpretation of the skill, or due to the augmentation technique being suboptimal compared to more structured methods like Latin Square symmetry. Nevertheless, symmetry augmentation leads to more interpretable and structured skill-to-state mappings. For example, when controlling base heading, policies with symmetry learn to rotate uniformly in both directions, whereas for those without symmetry, the behavior often tends to be biased. In Fig. 5, we visualize the effect of symmetry augmentation on state space coverage, showing how learned skills become more symmetrically distributed.

Downstream Task. We assess the utility of the learned skill libraries on a rough-terrain waypoint–navigation task with random position goals (up to 15 m away) and random target headings. We compare three control schemes: **(i)** *Direct*, a basic PPO policy that outputs joint position commands,

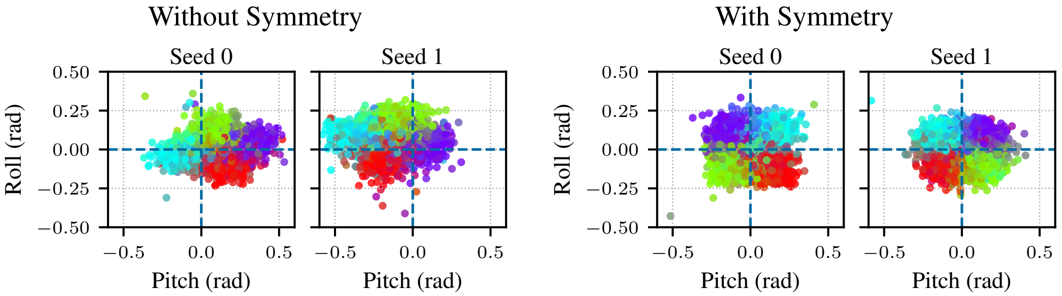


Figure 5: **Impact of symmetry augmentation on skill-to-state mappings.** Each point shows roll and pitch angles (in radians) reached by the policy, colored by the commanded skill. Without symmetry augmentation, the mapping is arbitrary and less structured. With symmetry augmentation, skills align symmetrically across the factor space, leading to more interpretable and balanced behaviors.

Table 3: **Performance on downstream navigation task.** Metrics include average reward, tracking errors, and episode termination ratios: timeouts (exceeding 30s), base collisions, or successful goal-reaching.

Approach	Reward ↑	Tracking Error		Termination Ratio		
		Heading ↓	Position ↓	Goal Reached ↑	Base Collision ↓	Time Out
Direct	1.85 ± 0.48	1.56 ± 0.03	11.00 ± 0.12	0.004 ± 0.000	0.996 ± 0.106	0.000 ± 0.000
DIAYN	27.87 ± 2.42	1.34 ± 0.05	7.50 ± 0.26	0.034 ± 0.000	0.000 ± 0.000	0.966 ± 0.000
DUSDi	35.26 ± 3.65	1.36 ± 0.05	6.74 ± 0.31	0.039 ± 0.005	0.001 ± 0.001	0.960 ± 0.004
2xMETRA	32.05 ± 7.73	1.54 ± 0.04	7.05 ± 0.32	0.090 ± 0.052	0.593 ± 0.103	0.317 ± 0.032
METRA	81.62 ± 50.20	1.33 ± 0.14	4.68 ± 2.78	0.300 ± 0.234	0.378 ± 0.532	0.322 ± 0.037
Mixed (Ours)	148.55 ± 29.24	1.03 ± 0.14	1.33 ± 0.27	0.797 ± 0.424	0.012 ± 0.014	0.191 ± 0.185
Oracle	164.37 ± 21.42	1.07 ± 0.17	1.66 ± 0.65	0.871 ± 0.427	0.052 ± 0.031	0.078 ± 0.111

(ii) *Oracle*, a hierarchical PPO policy with a hand-tuned velocity-tracking controller [35] as the low-level policy, and **(iii)** Skill-based, the same hierarchy as the oracle but using pre-trained skill-conditioned policies (from Tab. 2) as the low-level controller. As shown in Tab. 3, our *mixed* setup (using METRA for position, DIAYN for heading) closely matches the oracle, achieving low tracking error and high success rate. In contrast, direct control fails entirely because of poor structure in the action space. Mismatched or single-objective USD skill libraries also underperform, underscoring the importance of appropriate factor–algorithm combinations for downstream performance. Additional implementation details are in App. A.3.

6 Conclusion

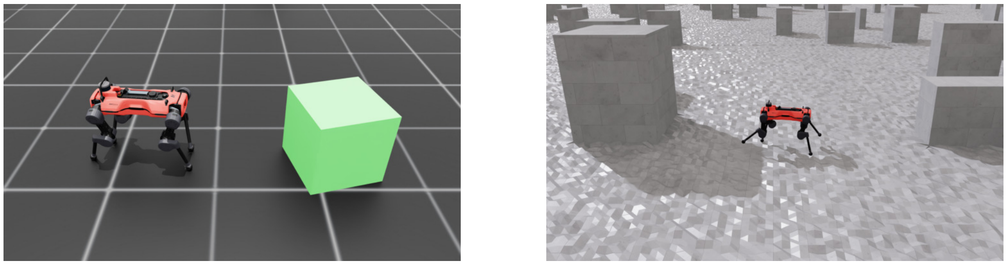
We presented a modular framework for unsupervised skill discovery (USD) that employs user-defined factorization of the state space and allows assigning different algorithms to each factor. This design leverages the complementary strengths of USD objectives: METRA excels at exploring unbounded dimensions like position through latent-space traversal, while DIAYN produces more distinguishable behaviors on bounded dimensions like heading or orientation. To support real-world deployment, our framework introduces several key components: a style factor and regularization terms that encourage safe and stable behaviors; symmetry augmentation that induces morphology-aware structure; and a factor-weighting mechanism that prioritizes relevant behaviors and resolves conflicts across active skills. We showed that these components individually contribute to skill quality, and their combination enables a smooth zero-shot transfer from simulation to hardware as well as intuitive control through direct skill commands. On downstream navigation tasks, our approach achieves near-oracle performance and significantly outperforms single or mismatched USD setups, demonstrating improved sample efficiency. The framework integrates seamlessly with scalable simulation and on-policy RL, and remains compatible with any USD method with intrinsic rewards. We open-source the code for future research in this direction: https://leggedrobotics.github.io/d3-skill-discovery/.

Future directions include scaling to more complex behaviors such as loco-manipulation and climbing boxes, which may require stronger exploration or curricula. Additionally, extending the framework to different robots with varying morphologies and symmetries is another promising avenue.

7 Limitations

Our results show that factorized, symmetry-aware USD can produce safe, deployable skills, but several gaps remain.

Discovering More Complex Skills. While the framework handled rough-terrain locomotion, extending it to complex, interaction-rich tasks was considerably more challenging. For instance, when adding a "box pose" factor to encourage loco-manipulation (Fig. 6a), the agent seldom learned the multi-stage behavior of walking to the box before the interaction. The discovered pushing skills often relied on unsafe, forceful collisions. Similarly, for obstacle-rich navigation without task-rewards (Fig. 6b), the agent failed to acquire obstacle avoidance behaviors. These observations indicate that additional guidance, such as task-aware curricula or alternative intrinsic objectives, is required to unlock more complex loco-manipulation and locomotion skills. More details are provided in App. A.6.



(a) Flat terrain with a $0.5\,\mathrm{m}$ cubic box weighing $10\,\mathrm{kg}$.

(b) Terrain with randomly placed static obstacles.

Figure 6: Environments for more complex skill discovery. (a) Adding a box pose factor encourages pushing, but the agent relies on unsafe, forceful collisions rather than controlled manipulation. (b) In an obstacle-rich environment, the agent explores but fails to discover safe avoidance behaviors without explicit rewards.

Cost of Symmetry in Skill Emergence. To enable the robot to learn pedipulation-like behaviors [36], we added factors for the position of each foot of the robot. However, we observed that the symmetry-mirroring suppressed the emergence of lifting of individual feet. Disabling symmetry produced a lift for only one of the legs, but this behavior did not extend to the other legs of the robot. These results suggest that fine-scale skills may benefit from softer symmetry biases or an explicit per-leg curriculum for skill discovery.

User-defined Design Decisions. Our proposed method still requires user decisions about factorization, algorithm selection, hyperparameters, and safety shaping. Automating these design choices and adding hard safety guarantees would make the framework more plug-and-play, increasing its usability across diverse robotic platforms and downstream tasks.

Acknowledgments

We thank Philip Arm and David Hoeller for helpful discussions at the start of the project. This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Automation (NCCR automation). Additionally, it has received funding from the European Union's Horizon Europe Framework Programme under grant agreement No 101121321.

References

[1] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 2022.

[2] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 2023.

[3] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[4] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning diverse skills without a reward function. In *International Conference on Learning Representations (ICLR)*, 2018.

[5] S. Park, O. Rybkin, and S. Levine. METRA: Scalable unsupervised rl with metric-aware abstraction. In *International Conference on Learning Representations (ICLR)*, 2024.

[6] J. Hu, Z. Wang, P. Stone, and R. Martín-Martín. Disentangled unsupervised skill discovery for efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[7] 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 (RSS)*, 2020.

[8] V. Atanassov, W. Yu, A. L. Mitchell, M. N. Finean, and I. Havoutis. Constrained skill discovery: Quadruped locomotion with unsupervised reinforcement learning. *arXiv preprint arXiv:2410.07877*, 2024.

[9] S. Kim, J. Kwon, T. Lee, Y. Park, and J. Perez. Safety-aware unsupervised skill discovery. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2023.

[10] H. Kim, B. Lee, H. Lee, D. Hwang, D. Kim, and J. Choo. Do's and don'ts: Learning desirable skills with instruction videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[11] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations (ICLR)*, 2020.

[12] D. J. Strouse, K. Baumli, D. Warde-Farley, V. Mnih, and S. Hansen. Learning more skills through optimistic exploration. In *International Conference on Learning Representations (ICLR)*, 2022.

[13] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *International Conference on Learning Representations (ICLR)*, 2018.

[14] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman. Ucb exploration via q-ensembles. *arXiv preprint arXiv:1706.01502*, 2017.

[15] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[16] I. Osband, D. Russo, and B. Van Roy. More efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[17] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[18] B. Sukhija, S. Coros, A. Krause, P. Abbeel, and C. Sferrazza. Maxinforl: Boosting exploration in reinforcement learning through information gain maximization. *arXiv preprint arXiv:2412.12098*, 2024.

[19] S. Park, J. Choi, J. Kim, H. Lee, and G. Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations (ICLR)*, 2022.

[20] S. Park, K. Lee, Y. Lee, and P. Abbeel. Controllability-aware unsupervised skill discovery. In *International Conference on Machine Learning (ICML)*, 2023.

[21] S. Rho, L. Smith, T. Li, S. Levine, X. B. Peng, and S. Ha. Language guided skill discovery. In *International Conference on Learning Representations (ICLR)*, 2024.

[22] Z. Wang, J. Hu, C. Chuck, S. Chen, R. Martín-Martín, A. Zhang, S. Niekum, and P. Stone. Skild: Unsupervised skill discovery guided by factor interactions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[23] D. O. Apraez, G. Turrisi, V. Kostic, M. Martin, A. Agudo, F. Moreno-Noguer, M. Pontil, C. Semini, and C. Mastalli. Morphological symmetries in robotics. *International Journal of Robotics Research (IJRR)*, 2025.

[24] Z. Su, X. Huang, D. Ordoñez-Apraez, Y. Li, Z. Li, Q. Liao, G. Turrisi, M. Pontil, C. Semini, Y. Wu, and K. Sreenath. Leveraging symmetry in rl-based legged locomotion control. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2024.

[25] M. Mittal, N. Rudin, V. Klemm, A. Allshire, and M. Hutter. Symmetry considerations for learning task symmetric robot policies. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2024.

[26] J. Cheng, M. Vlastelica, P. Kolev, C. Li, and G. Martius. Learning diverse skills for local navigation under multi-constraint optimality. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2024.

[27] T. Zahavy, Y. Schroecker, F. Behbahani, K. Baumli, S. Flennerhag, S. Hou, and S. Singh. Discovering policies with domino: Diversity optimization maintaining near optimality. In *International Conference on Learning Representations (ICLR)*, 2023.

[28] M. Vlastelica, J. Cheng, G. Martius, and P. Kolev. Offline diversity maximization under imitation constraints. In *Reinforcement Learning Conference*, 2024.

[29] P. Kolev, M. Vlastelica, and G. Martius. Dual-force: Enhanced offline diversity maximization under imitation constraints. In *Seventeenth European Workshop on Reinforcement Learning*, 2025.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

[32] B. Ravindran and A. G. Barto. Symmetries and model minimization in markov decision processes. Technical report, University of Massachusetts, USA, 2001.

[33] J. Dénes and A. Keedwell. *Latin Squares and Their Applications*. Academic Press, 1974. URL https://books.google.ch/books?id=W2IPAQAAMAAJ.

[34] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[35] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2021.

[36] P. Arm, M. Mittal, H. Kolvenbach, and M. Hutter. Pedipulate: Enabling manipulation skills using a quadruped robot's leg. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2024.

[37] T. Imagawa, T. Hiraoka, and Y. Tsuruoka. Unsupervised discovery of continuous skills on a sphere. *arXiv preprint arXiv:2305.14377*, 2023.

[38] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced skills by learning locomotion and local navigation end-to-end. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

A Appendix

A.1 Unsupervised Skill Discovery Algorithms

DIAYN: Diversity Is All You Need. DIAYN [4] aims to learn a skill-conditioned policy $\pi_\theta(\boldsymbol{a} \mid \boldsymbol{s}, \boldsymbol{z})$ by maximizing the mutual information (MI) between states and skills $I(\boldsymbol{S}; \boldsymbol{Z}) \triangleq D_{\mathrm{KL}}(p(\boldsymbol{s}, \boldsymbol{z}) \| p(\boldsymbol{s})p(\boldsymbol{z})) \equiv \mathcal{H}(\boldsymbol{Z}) - \mathcal{H}(\boldsymbol{Z} \mid \boldsymbol{S})$, where $\mathcal{H}(\cdot)$ denotes the Shannon or differential entropy. Intuitively, minimizing $\mathcal{H}(\boldsymbol{Z} \mid \boldsymbol{S})$ means that the skill $\boldsymbol{z}$ should be easy to infer given the state $\boldsymbol{s}$. This is implemented by learning a discriminator $q_\phi(\boldsymbol{s} \mid \boldsymbol{z})$ that approximates the posterior $p(\boldsymbol{s} \mid \boldsymbol{z})$. The policy and discriminator form a cooperative game: the discriminator predicts the skill that led to the policy visiting certain states, while the policy seeks to visit states that make it easy for the discriminator to identify the skill. The resulting reward is:

$$r_{\text{DIAYN}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) = \log q_\phi(\boldsymbol{z} \mid \boldsymbol{s}) - \log p(\boldsymbol{z}) \tag{2}$$

Note, $\log p(\boldsymbol{z})$ only needs to be included in the reward term if $p(\boldsymbol{z})$ is not uniform. The discriminator is trained by maximizing the log-likelihood of the posterior $\mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z}), \boldsymbol{s} \sim \pi_\theta(\boldsymbol{z})} \log q_\phi(\boldsymbol{z} \mid \boldsymbol{s})$. In practice, the discriminator predicts parameters of a distribution over the skill space, which depends on the selection of the prior $p(\boldsymbol{z})$.

The authors of the original paper use a categorical distribution for the prior $p(\boldsymbol{z})$. and noted that learning with continuous skill distributions such as uniform or Gaussian distribution degrades performance. Imagawa et al. [37] show that to learn more skills, using a continuous distribution yields better results than using a large number of discrete skills. Depending on the selection of the skill prior, the parameterization of the discriminator needs adjustments. Importantly, the support of the posterior has to contain the support of the prior. In Tab. 4 we list different combinations of priors and posteriors we tested. We found that Dirichlet-distributed skills offer a good trade-off between continuous skill expressiveness and discriminability accuracy.

Table 4: **Possible choices for skill distributions.** Depending on the choice of the prior distribution for the skills, we choose the posterior according to this table.

Prior $p(\boldsymbol{z})$	Posterior $q(\boldsymbol{z} \mid \boldsymbol{s})$
Uniform categorical	Categorical
Uniform continuous	Gaussian
Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$	Gaussian
Uniform on sphere	Von Mises-Fisher
Symmetric Dirichlet	Dirichlet

METRA: Metric-Aware Abstraction. METRA [5] (as well as LSD [19] and CSD [8]) aims to learn a skill-conditioned policy by learning an encoder ϕ that maps states into a latent space of the same dimensionality as the skill space. The skill discovery objective is to maximize the alignment of latent transitions $\phi(\boldsymbol{s}') - \phi(\boldsymbol{s})$ with the skill z under a constraint $\|\phi(\boldsymbol{s}) - \phi(\boldsymbol{s}')\|_2 \leq d(\boldsymbol{s}', \boldsymbol{s})$, with distance metric $d(\cdot, \cdot)$. LSD proposes to use the Euclidean distance between the states, $d(\boldsymbol{s}', \boldsymbol{s}) = \|\boldsymbol{s}' - \boldsymbol{s}\|$; CSD [20] proposes to use controllability-aware distance metric; while METRA [5] proposes to use temporal distance, *i.e.*, the minimum number of episodic steps to reach $\boldsymbol{s}'$ from $\boldsymbol{s}$, which in their setup is simply $d(\boldsymbol{s}', \boldsymbol{s}) = 1$.

The objective of the encoder in METRA is defined as, for all $(\boldsymbol{s}, \boldsymbol{s}') \in \mathcal{S}_{\text{adj}}$:

$$\mathcal{J}_{\text{METRA}}(\theta, \phi) = \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z}), \boldsymbol{s} \sim \pi_\theta(\boldsymbol{z})} \left[(\phi(\boldsymbol{s}') - \phi(\boldsymbol{s}))^\top \boldsymbol{z} \right] \text{s.t.} \|\phi(\boldsymbol{s}) - \phi(\boldsymbol{s}')\|_2 \leq d(\boldsymbol{s}', \boldsymbol{s}). \tag{3}$$

In practice, this constrained obejctive is optimized via dual-gradient descent. It simplifies into a reward function that rewards the agent if its actions result in state transitions that align with the skill in the latent state space:

$$r_{\text{METRA}}(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{s}') = (\phi(\boldsymbol{s}') - \phi(\boldsymbol{s}))^\top \boldsymbol{z}. \tag{4}$$

However, this objective leads to skills that move maximally fast through the state space, which might not be desirable. Instead, Atanassov et al. [8] propose a norm-matching objective to also control the execution speed of a skill, where for all $(\boldsymbol{s}, \boldsymbol{s}') \in \mathcal{S}_{\text{adj}}$:

$$\mathcal{J}_{\text{NM}}(\theta, \phi) = \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z}), \boldsymbol{s}\sim\pi_\theta(\boldsymbol{z})} \left[\|(\phi(\boldsymbol{s}') - \phi(\boldsymbol{s}) - \boldsymbol{z}\|^2 \right] \text{s.t.} \|\phi(\boldsymbol{s}) - \phi(\boldsymbol{s}')\| \leq d(\boldsymbol{s}', \boldsymbol{s}). \tag{5}$$

The resulting reward becomes a function of the error between the skill and the latent transition:

$$r_{\text{METRA,nm}}(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{s}') = (1 + \sigma \|(\phi(\boldsymbol{s}_{t+1}) - \phi(\boldsymbol{s}_t)) - \boldsymbol{z}\|_2^2)^{-1} \tag{6}$$

where $\sigma \in \mathbb{R}$ is a scaling factor.

In practice, we found that the norm-matching objective has a considerably weaker alignment component, making it difficult to train from scratch, particularly when the initial alignment between latent state transitions and skills is poor. The original alignment objective is often more stable in the early stages of training. To combine the strengths of both objectives, we implement a curriculum that starts with the original alignment objective and smoothly transitions to the norm-matching objective as alignment performance improves, *i.e.*, the final objective is a weighted sum:

$$\mathcal{J}_{\text{METRA mix}}(\theta, \phi) = (1 - \alpha_{\text{mix}}) \mathcal{J}_{\text{METRA}}(\theta, \phi) + \alpha_{\text{mix}} \mathcal{J}_{\text{NM}}(\theta, \phi). \tag{7}$$

The transition is controlled by the interpolation parameter $\alpha_{\text{mix}} \in [0, 1]$, which is dynamically calculated based on the cosine similarity between the latent state transition $(\phi(\boldsymbol{s}') - \phi(\boldsymbol{s}))$ and the skill $\boldsymbol{z}$. Specifically, α_{mix} linearly ramps from 0 to 1 as the cosine similarity score increases over a predefined range. For our experiments, this objective switching range is set to $[0.5, 0.7]$, as detailed in Tab. 7. This means that when the cosine similarity is below 0.5, the agent is trained purely on the alignment objective ($\alpha_{\text{mix}} = 0$), and when it exceeds 0.7, the training switches completely to the norm-matching objective ($\alpha_{\text{mix}} = 1$). In between these values, the objectives are mixed linearly. The weight α_{mix} is used for both the agent's reward calculation and the encoder's loss function.

DUSDi: Disentangled Unsupervised Skill Discovery. To learn disentangled skills, Hu et al. [6] propose learning two discriminators per factor, based on DIAYN. The first discriminator predicts the skill factor from the respective state factor $q_i(\boldsymbol{z}_i \mid \boldsymbol{s}_i)$, while the second discriminator predicts the skill from every other state factor: $q_{\neg i}(\boldsymbol{z}_i \mid \boldsymbol{s}_{\neg i})$, where $\boldsymbol{s}_{\neg i} \in \mathcal{S}_{\neg i} = \mathcal{S}_1 \times \ldots \mathcal{S}_{i-1} \times \mathcal{S}_{i+1} \times \ldots \mathcal{S}_N$. This results in a reward function defined as:

$$r_{\text{DUSDI}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) \triangleq \sum_{i=1}^{N} q_i(\boldsymbol{z}_i \mid \boldsymbol{s}_i) - \gamma q_{\neg i}(\boldsymbol{z}_i \mid \boldsymbol{s}_{\neg i}), \tag{8}$$

where $\gamma < 1$ is a hyperparameter that controls the importance of the entanglement penalty relative to the skill-factor association (typically γ = 0.1). The first reward component is the standard DIAYN reward, while the second one has the same formulation but is used as a penalty. The harder it is to infer a skill factor given other state factors, the more disentangled the learned skills are.

A.2 Symmetry Augmentation

Symmetry augmentation can help boost sample efficiency and learn smoother behaviors. Mittal et al. [25] propose to simply augment the collected data instead of introducing an extra symmetry objective, or enforcing symmetry in the network architecture.

So far, symmetry biases have not been used as part of unsupervised skill discovery. However, it might be useful to learn symmetric skills and boost exploration. To utilize symmetry augmentation, the MDP needs to have symmetries, which requires the reward to be invariant to symmetry transformations. In general, this is not the case in skill discovery. One way to enforce symmetry in the reward is computing it as an average over all symmetries:

$$r_{\text{sym}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{z}) = \frac{1}{K} \sum_{i=1}^{K} r(M_s^i(\boldsymbol{s}), M_a^i(\boldsymbol{a}), M_z^i(\boldsymbol{z})), \tag{9}$$

where r_{sym} is a reward that is guaranteed to be invariant over all symmetries. However, in practice, we found that it suffices to train all networks on symmetry-augmented data.

Skill Mirroring Function Properties. The function that mirrors skills, M_z, can be chosen freely as long as it preserves: (i) the invariance of the skill prior and (ii) the group composition of the underlying symmetries. The choice of a valid mirroring function M_z depends on the skill prior $p(\boldsymbol{z})$. For METRA, we use an isotropic prior, where all skills with the same norm $\|\boldsymbol{z}\|_2$ have the same probability, regardless of their direction. This means any mirroring function M_z is valid as long as it is norm-preserving (e.g., a reflection or rotation). For DIAYN, we use a symmetric Dirichlet distribution, $\mathrm{Dir}(\boldsymbol{z} \mid \alpha)$, where all entries of the concentration parameter $\alpha \in \mathbb{R}^d_+$ are equal. The resulting probability distribution is invariant to any permutation of the entries of $\boldsymbol{z}$. Therefore, for DIAYN, any mirroring function M_z is valid as long as it only permutes the entries of the skill vector.

The set of mirroring functions must also respect the composition of the physical symmetries. For example, mirroring a state *left–right* followed by *front–back* should yield the same physical transformation as a 180° rotation about the z-axis. The same composition must hold in the skill space. Otherwise, symmetry augmentation introduces contradictory training signals. Concretely, if $M_s^1\big(M_s^2(\boldsymbol{s}_i)\big) = M_s^3(\boldsymbol{s}_i) \quad \forall\, \boldsymbol{s}_i \in \mathcal{S}_i$, but there exists a skill $\boldsymbol{z}_i \in \mathcal{Z}_i \quad$ s.t. $\quad M_z^1\big(M_z^2(\boldsymbol{z}_i)\big) \neq M_z^3(\boldsymbol{z}_i)$, then symmetry augmentation can produce tuples with the *same* state but *different* mirrored skills:

$$\big(M_s^3(\boldsymbol{s}_i),\ \ldots,\ M_z^1\big(M_z^2(\boldsymbol{z}_i)\big)\big) \text{ and } \big(M_s^3(\boldsymbol{s}_i),\ \ldots,\ M_z^3(\boldsymbol{z}_i)\big).$$

Since states are now paired with ambiguous skills, any skill-conditioned reward or discriminator cannot remain consistent, yielding irreconcilable gradients and hindering learning. Therefore, M_z must satisfy $M_z^1 \circ M_z^2 = M_z^3$, to ensure coherent symmetry-augmented training.

The quadruped ANYmal-D is left-right and front-back symmetric, resulting in four symmetry transformations: identity, left-right reflection, front-back reflection, and their composition, a 180° rotation about the z-axis.

Skill Mirroring Function Implementation. For **DIAYN**, we mirror skills such that subskills form a Latin square. We use the following skill permutations:

$$\begin{aligned}
M_z^1(\boldsymbol{z}_i) &= [\boldsymbol{z}_i^1; \boldsymbol{z}_i^2; \boldsymbol{z}_i^3; \boldsymbol{z}_i^4] \\
M_z^2(\boldsymbol{z}_i) &= [\boldsymbol{z}_i^3; \boldsymbol{z}_i^4; \boldsymbol{z}_i^1; \boldsymbol{z}_i^2] \\
M_z^3(\boldsymbol{z}_i) &= [\boldsymbol{z}_i^2; \boldsymbol{z}_i^1; \boldsymbol{z}_i^4; \boldsymbol{z}_i^3] \\
M_z^4(\boldsymbol{z}_i) &= [\boldsymbol{z}_i^4; \boldsymbol{z}_i^3; \boldsymbol{z}_i^2; \boldsymbol{z}_i^1]
\end{aligned}$$

Permuting sub-skills gives the latent space room for states that are invariant to certain symmetries. Let $\mathcal{S}_{\text{sym}(i,j)} = \big\{\boldsymbol{s} \in \mathcal{S} \mid \ M_s^i(\boldsymbol{s}) = M_s^j(\boldsymbol{s})\big\}$ be such states (e.g., forward/backward velocity is invariant to a left–right flip). Whenever $\|\mathcal{S}_{\text{sym}(i,j)}\| > 1$ is, we require matching skills $\mathcal{Z}_{\text{sym}(i,j)} = \big\{\boldsymbol{z} \in \mathcal{Z} \mid \ M_z^i(\boldsymbol{z}) = M_z^j(\boldsymbol{z})\big\}$, which our permutation-based mirroring provides automatically:

$\mathcal{Z}_{\text{sym}(i,j)}$	$j=1$	$j=2$	$j=3$	$j=4$
$i=1$	$\mathcal{Z}$	$[a;b;a;b]$	$[a;a;b;b]$	$[a;b;b;a]$
$i=2$	$[a;b;a;b]$	$\mathcal{Z}$	$[a;b;b;a]$	$[a;a;b;b]$
$i=3$	$[a;a;b;b]$	$[a;b;b;a]$	$\mathcal{Z}$	$[a;b;a;b]$
$i=4$	$[b;a;a;b]$	$[a;a;b;b]$	$[a;b;a;b]$	$\mathcal{Z}$

where $a, b \in \mathbb{R}^n$ denote subskills. This pattern guarantees that a state symmetric under i and j can always be paired with a unique skill lying in $\mathcal{Z}_{\text{sym}(i,j)}$.

If $\mathcal{S}_{\text{sym}(i,j)} = \mathcal{S}$, i.e., every state of a factor is invariant under a pair (i,j), then the skill map must satisfy $\mathcal{Z}_{\text{sym}(i,j)} = \mathcal{Z}$ as well. For the heading-rate factor, a scalar that flips sign under left-right or front-back reflections but is unchanged by their composition, both reflections are merged into a single "flip", resulting in the symmetries $\{1,2\}$ = (identity, flip). With two subskills we set $M_z^1(\boldsymbol{z}_i) = [\boldsymbol{z}_i^1; \boldsymbol{z}_i^2]$ and $M_z^2(\boldsymbol{z}_i) = [\boldsymbol{z}_i^2; \boldsymbol{z}_i^1]$. For states with no symmetries, e.g., the base height, we omit symmetry augmentation for the skills completely.

For **METRA**, we treat the skill vector $\boldsymbol{z}_i$ as a directional proxy for its associated state factor and mirror it accordingly. This approach is grounded in the geometric nature of METRA's alignment objective. Specifically, since the skill $\boldsymbol{z}_i$ represents a direction, the mirroring function M_z^k applied to the skill is defined to be the same geometric transformation as the function M_s^k applied to the state. For instance, if a state factor like the robot's base velocity is reflected across a plane for a left-right symmetry transformation, its corresponding skill vector is also reflected across that same plane. Applying identical symmetry transformations to both states and skills introduces a strong inductive bias that aligns the learned skill behaviors with the physical symmetries of the robot. This consistency is critical for the METRA objective, which directly rewards the alignment between the skill vector and the change in the latent state representation.

In our experiments, we also found it crucial to limit the dimensionality of these directional skill vectors to $d \leq 3$. This is an empirical finding. When we experimented with higher-dimensional skill vectors ($d > 3$), the policy consistently learned to ignore the additional, non-mirrored dimensions. This behavior effectively caused the learned skill space to collapse back into a 3D geometric subspace, and attempts to define more complex, higher-dimensional mirroring functions did not prevent this instability. Therefore, constraining the skill dimensionality to match the 3D nature of the physical transformations proved to be the most stable and effective approach.

A.3 Implementation Details

Policy Network. For the policy, we use a 3-layer MLP [512, 256, 128] with elu activations. The action space is 12-dimensional, corresponding to the robot's joint position targets. For each action dimension, the policy predicts the mean and log std of a Gaussian distribution, of which we clamp the standard deviation to the range $[e^{-5}, e^{2}]$. During training, actions are sampled from the predicted distribution. During deployment, the action is the predicted mean.

Hyperparameters. We list hyperparameters for PPO in Tab. 5, for METRA in Tab. 7, and for DIAYN in Tab. 8. The rewards for the style factor are listed in Tab. 9 and the regularization penalties in Tab. 10. The policy observations can be found in Tab. 6.

Table 5: PPO Hyperparameters

Hyperparameter	Value
PPO clip ratio	0.2
Value clip ratio	0.2
Num env steps before update	24
Num learning epochs	5
Num minibatches	4
Learning rate	1.0e-3
Discount factor	0.99
GAE lambda	0.95
KL target	0.01
Max grad norm	1.0

Table 6: Policy Observations

Name	Dim
Base xy-position in world frame	2
Base linear velocity	3
Base angular velocity	3
Projected gravity	3
Previous action	12
Joint position	12
Joint velocity	12
Height scan ($1.6\,\mathrm{m} \times 1.0\,\mathrm{m}$)	231

Table 7: METRA Hyperparameters

Hyperparameter	Value
Learning rate	1.0e-4
Initial Lagrange multiplier	30.0
Lagrange multiplier lr	1e-4
Lagrange multiplier slack	1e-5
Objective switching range	(0.5, 0.7)
Network	MLP: [256, 256]
Norm-matching σ	10.0

Table 8: DIAYN Hyperparameters

Hyperparameter	Value
Skill distribution	Dirichlet
Disentanglement λ	0.1
Network	MLP: [256, 256]
Learning rate	1e-4
Dirichlet param range	(0.05, 1.0)

Critic Decomposition. In DUSDi [6], the authors also propose decomposing the Q function as a sum of Q values over individual factors. We do the same, but with value functions. Additionally, instead of having one value function per factor, we may have an ensemble of value functions per factor

Table 9: Style Factor Rewards

Name	Objective	Weight
Joint torques	$\lVert\boldsymbol{\tau}\rVert_2^2$	-1.0e-3
Joint acceleration	$\lVert\ddot{\mathbf{q}}\rVert_2^2$	-1.0e-5
Action rate	$\lVert\boldsymbol{a}_t - \boldsymbol{a}_{t-1}\rVert_2^2$	-0.2
Action norm	$\lVert\boldsymbol{a}\rVert_2^2$	-0.4
Undesired Contacts	$\sum_{b\in\{\text{Thighs, Shanks, Base}\}} \mathbf{1}\left[\text{contact}(b)\right]$	-30.0
Base height	$\lVert p_z - 0.55\rVert^2$	-10.0
Flat orientation	$\lVert\mathbf{g}_{b,\text{xy}}\rVert_2^2$	-10.0

Table 10: Regularization Rewards

Name	Objective	Weight
Joint torques	$\lVert\boldsymbol{\tau}\rVert_2^2$	-1.0e-3
Joint acceleration	$\lVert\ddot{\mathbf{q}}\rVert_2^2$	-2.5e-7
Action rate	$\lVert\boldsymbol{a}_t - \boldsymbol{a}_{t-1}\rVert_2^2$	-0..05
Torque limits	$\lVert \max\left(\boldsymbol{\tau} - \tau_{\max},\ 0\right)\rVert_1 + \lVert \min\left(\boldsymbol{\tau} - \tau_{\min},\ 0\right)\rVert_1$	-15.0
Torque ratio limits	$\lVert \max\left(\boldsymbol{\tau} - 0.75\tau_{\max},\ 0\right)\rVert_1 + \lVert \min\left(\boldsymbol{\tau} - 0.75\tau_{\min},\ 0\right)\rVert_1$	-15.0
Joint vel limits	$\lVert\min\left(\max\left(\lvert\dot{\mathbf{q}}\rvert - \dot{\mathbf{q}}_{\text{lim}},\ 0\right),\ 1.0\right)\rVert_1$	-10.0
Joint pos limits	$\left\lVert\max\left(\mathbf{q} - \mathbf{q}_{\text{lim}}^{\text{upper,soft}},\ 0\right) + \min\left(\mathbf{q} - \mathbf{q}_{\text{lim}}^{\text{lower,soft}},\ 0\right)\right\rVert_1$	-10.0
Upside down termination	$\mathbf{1}$ [flipped termination]	-5000

due to UCB exploration [14]. To do so, we need to store the individual factor rewards separately, as the aggregated rewards are only required for the policy update. As a result, we do weighted aggregation over the advantage estimates:

$$A = \lambda_{N+1} A_{\text{style}} + \sum_{i=1}^{N} \lambda_i (A_{i,\mu} + \lambda_{\text{UCB}} A_{i,\sigma}). \tag{10}$$

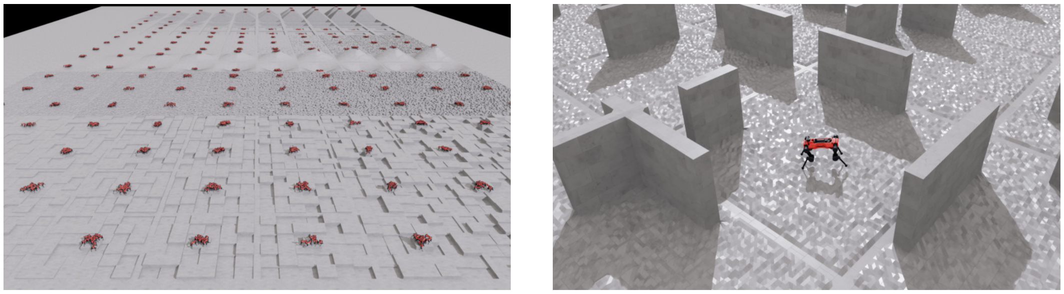
where A_{style} is the advantage of the style factor and $A_{i,\mu}$, $A_{i,\sigma}$ are the mean and standard deviation of the advantage ensembles per factor. The policy is updated with aggregated advantage A.

Environments. The simulation environments are implemented in NVIDIA Isaac Lab [34]. Fig. 7a shows the different types of terrain used in the experiments. The environment comprises flat, randomly rough, and pyramidal sloped and stair terrains. The robot is placed at the center of these sub-terrains and no external task-specific objectives are provided from the environment. The robot needs to learn diverse skills through its intrinsic USD objectives.

Similarly to previous work [38], we randomize the physical properties of the robot (such as friction and base mass) and introduce external pushes for robustness. An episode end if the robot base rotates more than $100\,\text{deg}$ or the duration of the episode reaches $30\,\text{s}$.

Inspired by Rudin et al. [38], we design a game-inspired terrain curriculum where the robot encounters increasingly difficult sub-terrains as training progresses. Our curriculum is not driven by a task-specific reward, but rather by a task-agnostic measure of skill capability: state coverage. Since increasingly difficult terrain primarily challenges the agent's ability to explore its position state space, we use the coverage of this specific factor to control the curriculum's progression. We quantify this coverage by the total distance traversed, and adjust the terrain difficulty based on performance: if an agent travels more than $10.0\,\text{m}$, it advances to a more difficult level, whereas if it travels less than $5.0\,\text{m}$, it is moved to an easier one.

The environments shown in Figs. 6a, 6b and 7b are used for additional experiments on discovering loco-manipulation and high-level navigation skills. These are discussed in App. A.6.



(a) Game-based curriculum terrain design.

(b) Rough terrain environment with random walls.

Figure 7: **Environments used for learning skills.** These environments are generated procedurally using the same mechanism as in Rudin et al. [38]. The policy receives the height-scan for perceiving the different terrains.

A.4 Evaluation Metrics

Metric Score. The metric score, a value in $[-1, 1]$, quantifies the performance of each skill factor. The definition of the score varies by the type of factor.

- For METRA factors, the score is the cosine similarity between the latent state transition $(\phi(\boldsymbol{s}') - \phi(\boldsymbol{s}))$ and the commanded skill $\boldsymbol{z}$. This corresponds directly to the METRA reward signal, $r_{\text{METRA}}(\boldsymbol{s}, \boldsymbol{z}, \boldsymbol{s}')$.
- For DIAYN factors, the score is the cosine similarity between the commanded skill $\boldsymbol{z}$ and the expectation of the predicted posterior, $\mathbb{E}[q_\phi(\boldsymbol{z} \mid \boldsymbol{s})]$. Note: For a Dirichlet posterior, this value is always positive, as its support is the probability simplex.
- For the style factor, we directly use the scaled extrinsic reward as the metric score.

Due to these different definitions, metric scores are not comparable across different factor types and should only be compared for the same factor across different experimental runs.

Diversity. To quantify the diversity of learned behaviors, we measure the breadth of the state space that the policy can reach. We calculate this metric as follows:

1. Sample a large number of skills, $n > 10,000$, from the prior $p(\boldsymbol{z})$.
2. For each skill, execute a full rollout with the policy to collect a trajectory of states.
3. Calculate the mean state for each of the n trajectories.
4. Calculate the standard deviation over these n mean states.

This final standard deviation serves as our diversity metric, where higher values indicate broader state coverage.

A.5 Additional Results and Discussion

In this section, we provide additional details and insights for the experiments in the main paper.

Mixing USD Algorithms for Diverse Factor Types. The results in Tab. 2 highlight a key insight beyond the performance trade-off mentioned in the main text: single-algorithm baselines tend to over-specialize. We observe that when a method struggles with one type of factor (e.g., unbounded position), its measured success on another (e.g., bounded heading) can be inflated. Our mixed approach avoids this issue by leveraging each algorithm's strengths. This principle also guides our choice of skill dimensions, where we use a 2D skill for METRA to match the xy-plane's geometry and a 4D skill for DIAYN to represent discrete-like heading directions.

Investigating Reward Weighting for Conflicting Factors. We expected factor weighting to alleviate the issue in factorized skill learning when certain factors cannot be interacted with simultaneously. To evaluate this, we factorized the position factor further into four quadrants (NE ($x > 0, y > 0$), SE ($x > 0, y < 0$), SW ($x < 0, y < 0$), and NW ($x < 0, y > 0$)). The robot cannot be in multiple quadrants simultaneously, which conflicts with the skills commanded for each factor in each quadrant. We trained all factors with METRA and without symmetry augmentation. In Fig. 8 we visualize the cosine similarity between the latent state transition and the commanded skills, and the scaled style reward for setups with and without weighting. Weighting does not change the performance significantly. We hypothesize that factor weights did not help because while they affect the policy's reward, the underlying USD networks (discriminators/encoders) are still trained on all collected data. This means data from trajectories where a factor's weight was near zero is still used to train that factor's USD network, creating conflicting gradients. This suggests a valuable future direction: incorporating the factor weights into the USD network loss, such that the collected rollouts are weighted by their relevance to each factor during the network update.

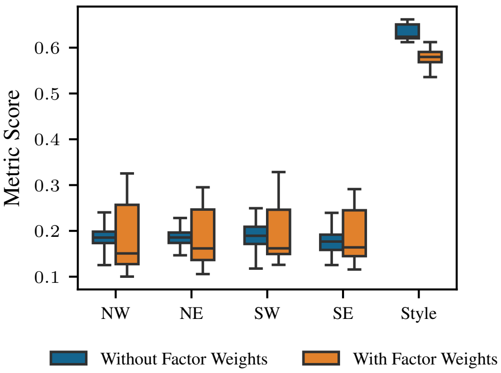


Figure 8: Effect of factor weighting on conflicting factors. Weighting does not improve the metrics

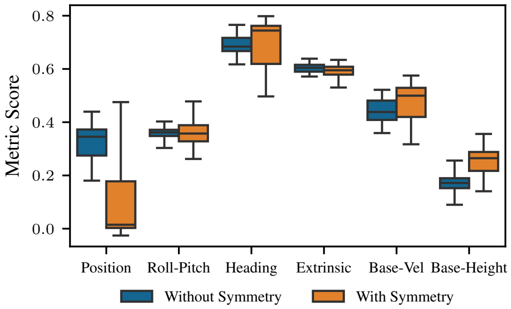


Figure 9: Effect of symmetry augmentation on skill learning metrics.

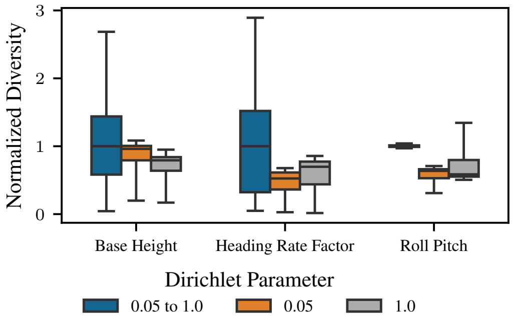


Figure 10: Effect of Dirichlet parameter on diversity on three factors.

Symmetry Augmentation. In Fig. 9, we show how symmetry augmentation affects skill discovery performance across different state factors. METRA is used for the position factor, while DIAYN is used for the others. Notably, symmetry augmentation reduces the encoder accuracy in METRA, while it has little effect on the discriminator accuracy in DIAYN. We hypothesize that this is because METRA relies on a more direct, directional interpretation of the skill vector, making it more sensitive to the constraints introduced by symmetry augmentation.

DIAYN Distribution Type. We evaluate the impact of different Dirichlet priors on skill diversity for DIAYN-trained factors. We compare three setups: a fixed high-concentration prior ($\alpha = 0.05$, a fixed low-concentration prior ($\alpha = 1.0$), and a curriculum that linearly increases α from 0.05 to 1.0 based on discriminator accuracy. As shown in Fig. 10, the curriculum setup results in greater variability and often higher diversity scores for the base height and heading rate factors. This suggests that starting with sparse skill sampling helps early specialization, while gradually broadening the support encourages later diversity. In contrast, the Roll-Pitch factor appears less sensitive to the choice of prior, likely due to its lower inherent diversity. Overall, the curriculum provides a trade-off between diversity and training stability.

A.6 Additional Tasks

Loco-manipulation. We attempted to learn loco-manipulation skills by placing a movable box in the environment (shown in Fig. 6a) and adding the box pose as an additional state factor. However, this setup alone failed to produce meaningful skills, as interactions with the box were rare and the resulting intrinsic rewards from the box factor were weak. To address this, we incorporated exploration guidance using RND [13] and UCB [14]. Neither method led to significant improvements. With RND, the prediction error rapidly decreased before the agent could discover interactions with the box, providing little incentive to explore it. With UCB, the agent received persistently high intrinsic rewards due to high ensemble disagreement, caused by low-quality value estimates, without corresponding learning progress, leading to unstructured and unproductive behavior.

Strouse et al. [12] showed that DIAYN suffers from poor exploration. To still encourage high diversity, we can add an exploration bonus on top of the pure skill discovery reward. A simple form of intrinsic motivation is random network distillation (RND) [13], which encourages exploration by rewarding states that are rarely visited. Another method to encourage exploration is to use ensemble disagreement as an exploration reward. One way to implement this, proposed by Strouse et al. [12], is by defining multiple discriminators $q_{\phi_i}(\boldsymbol{z} \mid \boldsymbol{s})$ and then rewarding the agent for high entropy of the mixture compared to the mean entropy.

$$r_{\text{DISDAIN}} = \mathcal{H}\left(\frac{1}{N}\sum_{i=0}^{N} q_{\phi_i}(\boldsymbol{z} \mid \boldsymbol{s})\right) - \frac{1}{N}\sum_{i=0}^{N} \mathcal{H}\left(q_{\phi_i}(\boldsymbol{z} \mid \boldsymbol{s})\right) \tag{11}$$

Depending on the distribution, this may not be easy to implement. A simpler approach based on ensemble disagreement uses the variance of the rewards as an exploration bonus:

$$r_{\text{EXPLORE}} = \text{Var}([\log q_{\phi_1}(\boldsymbol{z} \mid \boldsymbol{s}), \ldots, \log q_{\phi_N}(\boldsymbol{z} \mid \boldsymbol{s})]) \tag{12}$$

This is similar to the method proposed by Chen et al. [14], which used a Bayesian learning approach by updating the policy based on an upper confidence bound (UCB) for the value estimate by defining an ensemble of value functions and adding a disagreement bonus. However, this method also did not help to discover meaningful loco-manipulation skills.

High-level Navigation. We investigated whether our framework could be applied hierarchically to learn complex navigation behaviors without direct supervision. For this, we used one of our pre-trained USD policies as a fixed, low-level controller that provides a library of basic skills. We then trained a high-level policy on top, again using our USD objective, which learns to sequence these skills by outputting skill vectors for the low-level policy. We evaluated this in environments with randomly placed obstacles (shown in Figs. 6b and 7b), applying the METRA objective on the base position factor to train the high-level policy. While the agent learned to cover the space, it struggled to avoid obstacles, even with access to a 2D planar distance scan. We hypothesize that this is due to the lack of an extrinsic signal that encourages obstacle avoidance.

We also explored using this setup to learn box-pushing behaviors. To simplify the task, the box could be moved by simple collisions. Using METRA on the box position factor, the agent learned to push the box effectively. However, since the low-level policy was not trained in the presence of the box, it lacked any meaningful manipulation capabilities. As a result, the agent relied on forceful collisions to move the box, an approach that is unsafe for real-world deployment.