arXiv:2603.21268v1 [cs.RO] 22 Mar 2026

Evaluating Factor-Wise Auxiliary Dynamics Supervision for Latent Structure and Robustness in Simulated Humanoid Locomotion

Chayanin Chamachot*
Department of Computer Engineering, Faculty of Engineering,
Chulalongkorn University, Bangkok 10330, Thailand

Abstract

We evaluate whether factor-wise auxiliary dynamics supervision produces useful latent structure or improved robustness in simulated humanoid locomotion. DynaMITE—a transformer encoder with a factored 24-d latent trained by per-factor auxiliary losses during proximal policy optimization (PPO)—is compared against Long Short-Term Memory (LSTM), plain Transformer, and Multilayer Perceptron (MLP) baselines on a Unitree G1 humanoid across four Isaac Lab tasks. The supervised latent shows no evidence of decodable or functionally separable factor structure: probe $R^2 \approx 0$ for all five dynamics factors, clamping any subspace changes reward by < 0.05, and standard disentanglement metrics (MIG, DCI, SAP) are near zero. An unsupervised LSTM hidden state achieves higher probe R^2 (up to 0.10). A 2×2 factorial ablation ($n = 10$ seeds) isolates the contributions of the tanh bottleneck and auxiliary losses: the auxiliary losses show no measurable effect on either in-distribution (ID) reward ($+0.03$, $p = 0.732$) or severe out-of-distribution (OOD) reward ($+0.03$, $p = 0.669$), while the bottleneck shows a small, consistent advantage in both regimes (ID: $+0.16$, $p = 0.207$; OOD: $+0.10$, $p = 0.208$). The bottleneck advantage persists under severe combined perturbation but does not amplify, indicating a training-time representation benefit rather than a robustness mechanism. LSTM achieves the best nominal reward on all four tasks ($p < 0.03$); DynaMITE degrades less under combined-shift stress (2.3% vs. 16.7%), but this difference is attributable to the bottleneck compression, not the auxiliary supervision. For locomotion practitioners: auxiliary dynamics supervision does not produce an interpretable estimator and does not measurably improve reward or robustness beyond what the bottleneck alone provides; recurrent baselines remain the stronger choice for nominal performance.

Keywords: reinforcement learning; humanoid locomotion; domain randomization; latent dynamics; auxiliary losses; sim-to-real; disentanglement; representation analysis

1 Introduction

Reinforcement learning (RL) policies for humanoid locomotion are typically trained in simulation with domain randomization (DR) [6, 7, 10] and must transfer to conditions that differ from the training distribution. One strategy for handling dynamics variation is latent dynamics identification: training an encoder to infer a compact latent representation of the environment's physical parameters from an observation–action history, then conditioning the policy on this estimate. Rapid Motor Adaptation (RMA) [1] demonstrated this approach with a two-phase training pipeline, achieving sim-to-real transfer on a quadruped. Subsequent work [2, 3] has extended the paradigm, generally reporting improved robustness.

A natural refinement is to supervise the latent

*Correspondence: 6633041021@student.chula.ac.th
ORCID: 0009-0009-1605-7988

representation per factor—training dedicated subvectors to predict individual dynamics parameters (friction, mass, motor strength, etc.) rather than a monolithic estimate. The intuition is appealing: a factored, decodable latent should enable both better generalization and post-hoc interpretability, allowing practitioners to inspect *what* the policy believes about each physical quantity.

We test this intuition rigorously. Our architecture, DynaMITE (Dynamics-Matching Inference via Transformer Encoding), maps a short history through a transformer encoder to a 24-dimensional factored latent with five dedicated subspaces trained by per-factor auxiliary losses during PPO. We compare it against LSTM, plain Transformer, and MLP baselines on a Unitree G1 humanoid across four locomotion tasks in NVIDIA Isaac Lab, using 5 seeds for the main comparison, 10 seeds for ablations, and over 7,000 evaluation episodes for push recovery.

The central finding is negative: factor-wise auxiliary dynamics supervision does not produce a decodable or functionally separable latent representation in this setting. Probe $R^2 \approx 0$ for all five factors (linear and MLP probes); clamping any factor subspace changes reward by less than 0.05; and standard disentanglement metrics (MIG, DCI, SAP) are near zero. An unsupervised LSTM hidden state achieves higher probe R^2 (up to 0.10). A 2×2 factorial ablation isolates the bottleneck and auxiliary-loss contributions: the auxiliary losses show no measurable effect ($p > 0.66$), while the tanh bottleneck shows a small, consistent advantage ($p \approx 0.2$). DynaMITE degrades less than LSTM under severe combined-shift perturbation (2.3% vs. 16.7%), but the factorial confirms this reflects the bottleneck compression, not the auxiliary supervision.

Our analysis uses six complementary tools: probes (decodability), interventions (functional separability), gradient flow (training dynamics), SVD geometry (representation structure), KNN-based MI (information content), and standard disentanglement metrics (MIG, DCI, SAP). Across all six, results for dynamics identification are null or weak. We report these negative findings in full, following calls for more complete reporting of null results in RL [40].

2 Related Work

2.1 Domain Randomization and Sim-to-Real Transfer

Domain randomization (DR) trains policies over distributions of simulator parameters to improve robustness at deployment [6, 7]. Successes span quadrupeds [10, 11, 12], bipeds [13, 14, 15], and full humanoids [16, 17]. However, excessively wide randomization can degrade nominal performance [18, 19, 20], motivating adaptive methods that infer dynamics online rather than relying solely on broad randomization.

2.2 History-Conditioned Policies

LSTM-based policies implicitly adapt via hidden state dynamics [8, 21, 22, 23]. Transformers process a fixed-length history window with attention [9, 24, 16], enabling parallel training and explicit context length control. Decision Transformer [25] and Algorithm Distillation [26] frame RL as sequence modeling but target offline and meta-RL respectively; our setting is online PPO with DR.

2.3 Latent Dynamics Identification

Classical system identification recovers physics parameters from input–output data [30, 31]. RMA [1] introduced a two-phase approach: train a privileged teacher with access to ground-truth dynamics, then distill an adaptation module that infers a latent dynamics estimate from history. World models [32, 33] learn latent dynamics for planning; context-conditioned policies [34, 35] achieve similar goals in a meta-RL framework. DynaMITE differs by integrating per-factor auxiliary supervision directly into the PPO training loop via a factored bottleneck, avoiding two-phase distillation.

2.4 Representation Analysis in RL

Probing classifiers assess what information is decodable from learned representations [4, 5]. Disentanglement metrics (MIG [36], DCI [37],

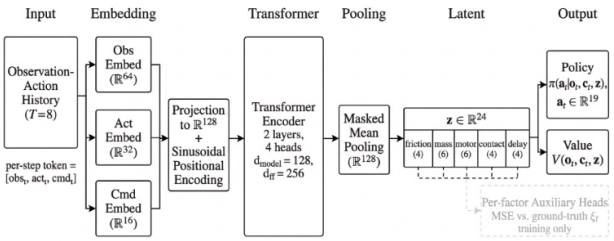

Figure 1: Overview of the DynaMITE architecture. A two-layer transformer encoder processes an 8-step (160 ms) observation–action history to produce a 24-dimensional factored latent vector $\boldsymbol{z} \in \mathbb{R}^{24}$, decomposed into five factor subspaces (friction, mass, motor strength, contact stiffness, action delay). Each subspace is trained with a dedicated auxiliary dynamics-prediction loss during PPO training. The latent is concatenated with the current observation and fed to the policy $\pi(a \mid s, \boldsymbol{z})$ and value $V(s, \boldsymbol{z})$ heads. Auxiliary losses are active only during training.

SAP [38]) quantify factor alignment in generative models but have seen limited application to RL latent spaces. Recent work in RL representation analysis [39] studies zero-shot transfer via disentangled features. Our analysis applies six complementary tools to evaluate whether auxiliary supervision produces the intended latent structure.

3 Method

The DynaMITE architecture (Figure 1) maps a short observation–action history to a factored latent vector concatenated with the current observation before being fed to the policy and value heads.

3.1 History Buffer and Transformer Encoder

At each control step t, the agent maintains a history buffer of the $H = 8$ most recent observation–action pairs $\{(o_{t-H+1}, a_{t-H+1}), \ldots, (o_t, a_t)\}$, corresponding to a 160 ms context window at the 50 Hz control rate. Each pair is embedded via learned linear projections and summed with sinusoidal positional encodings. The resulting sequence of tokens is processed by a transformer encoder with 2 layers, 4 attention heads, and model dimension $d_{\text{model}} = 128$. The encoder output is mean-pooled across the sequence dimension to yield a fixed-size context vector $\boldsymbol{h} \in \mathbb{R}^{128}$.

3.2 Factored Latent Head

A linear projection followed by a tanh nonlinearity maps $\boldsymbol{h}$ to a 24-dimensional latent vector $\boldsymbol{z} \in \mathbb{R}^{24}$, which is explicitly partitioned into five contiguous factor subspaces: friction (dims 0–3), mass (dims 4–9), motor strength (dims 10–15), contact stiffness (dims 16–19), and action delay (dims 20–23). The subspace sizes are chosen proportional to the expected complexity of each factor. During training, each subspace $\boldsymbol{z}_f$ receives a dedicated auxiliary loss that trains it to predict the corresponding ground-truth dynamics parameter θ_f:

$$\mathcal{L}_{\text{aux},f} = \|g_f(\boldsymbol{z}_f) - \theta_f\|^2, \tag{1}$$

where g_f is a small MLP head. The total training loss is:

$$\mathcal{L} = \mathcal{L}_{\text{PPO}} + c_v \mathcal{L}_{\text{value}} + 0.1 \sum_f \mathcal{L}_{\text{aux},f}, \tag{2}$$

with auxiliary loss weight fixed at 0.1. Auxiliary losses are active only during training; at evaluation time, the auxiliary heads are discarded.

3.3 Policy and Value Heads

The latent $\boldsymbol{z}$ is concatenated with the current observation o_t and fed to shared-architecture policy and value MLPs. The policy head outputs a Gaussian distribution $\pi(a \mid o_t, \boldsymbol{z})$ from which actions are sampled during training; at evaluation, actions are taken as the distribution mean (deterministic mode). The value head outputs $V(o_t, \boldsymbol{z})$ for advantage estimation.

3.4 Baseline Architectures

All four model architectures share the same observation embedding, action embedding, policy

MLP, and value MLP. They differ only in the history encoding mechanism:

Table 1: Model architecture summary. Parameter counts vary by task due to different observation dimensions.

Model	History	Latent	Aux Loss	Params
MLP	None	No	No	266–362k
LSTM	Hidden state	No	No	176–215k
Transformer	8 steps	No	No	330–342k
DynaMITE	8 steps	24-d factored	Yes	342–392k

The **MLP** baseline receives only the current observation with no history. The **LSTM** baseline processes observations sequentially and conditions the policy on its hidden state. The **Transformer** baseline uses the same encoder as DynaMITE but without the factored latent head or auxiliary losses—the mean-pooled context vector is passed directly to the policy and value heads. This baseline controls for the transformer encoder architecture, but does not isolate the auxiliary losses from the 24-d tanh bottleneck: DynaMITE adds both the bottleneck and the auxiliary supervision simultaneously, so any behavioral difference between Transformer and DynaMITE may reflect either component or their interaction. The 2×2 factorial ablation (Section 5.7.1) addresses this confound by independently varying bottleneck usage and auxiliary losses; all factorial conditions share identical architecture, training protocol, and evaluation.

4 Evaluation Protocol

To reduce analytic flexibility, we fixed the evaluation protocol—including metrics, sweep levels, and checkpoint-selection criteria—before the main experiment campaign.

4.1 Training Protocol

All models are trained with PPO (clipped objective, generalized advantage estimation (GAE)) [7] using 512 parallel Isaac Lab environments. Each training run spans 10 million timesteps at a control frequency of 50 Hz ($\Delta t = 20$ ms), requiring approximately 14 minutes on a single NVIDIA RTX 4060 Laptop GPU. Checkpoints are saved every 614,400 steps (∼60 s), and the best checkpoint is selected by training-time stochastic evaluation reward. All reported results then use deterministic evaluation (Section 4.2); the stochastic-to-deterministic gap is a potential source of noise but is consistent across all models.

4.2 Deterministic Evaluation

All reported numbers use **deterministic evaluation**: actions are taken as the distribution mean with no sampling. Each evaluation consists of 100 episodes (main comparison, ablations) or 50 episodes (OOD sweeps, push recovery). Episodes use fixed-length rollouts with no early termination. The evaluation environment seed is fixed at 42 for all models within a task, ensuring identical initial conditions. Environment resets fully randomize initial joint positions and domain parameters.

The main comparison spans 5 training seeds (42–46) $\times$ 4 tasks $\times$ 4 models = 80 evaluations. OOD sweeps cover 5 training seeds $\times$ 4 models $\times$ 5 sweep types $\times$ 3 tasks = 140 evaluations. Push recovery evaluates 5 seeds $\times$ 4 models $\times$ 7 magnitudes $\times$ 50 episodes = 7,000 episodes. Mechanistic analyses use 3 seeds for gradient flow (10M steps each) and 5 seeds for geometry and mutual information estimation (∼36,000 representations per model/seed).

4.3 Metrics

Reward. Penalty-based (always negative); higher (less negative) values indicate better performance. A method achieving -4.18 vs. -4.48 accumulates approximately 6% less penalty per step.

OOD sensitivity (custom metric). Defined as $\max(\bar{r}) - \min(\bar{r})$ across sweep levels, where

$\bar{r}$ is the mean reward at each level. Lower values indicate a flatter reward curve across perturbation levels. *Caveat:* this metric does not distinguish models with uniformly poor performance from models that genuinely resist degradation; a model that performs badly everywhere will also have low sensitivity. It should therefore be interpreted alongside absolute reward levels (severe-level mean, worst-case reward). We also report the severe-level mean (average of the two highest-severity levels), worst-case reward (most negative across all sweep levels), and degradation (absolute difference between ID reward and severe-level mean, expressed also as a percentage of |ID Reward|).

Recovery time (custom metric). Steps from push onset until the velocity tracking error drops below a threshold of 1.5. Measured in the push-recovery protocol with controlled push magnitudes.

Factor alignment (custom metric). A within-factor Pearson correlation ratio, not a standard disentanglement benchmark such as MIG, DCI, or SAP. Chance level is 0.20 for five factors. Not comparable to metrics from prior disentanglement work.

4.4 Statistical Reporting

For the main comparison ($n = 5$ seeds), we report mean $\pm$ standard deviation, 95% confidence intervals (t-distribution), and paired t-tests. For ablations ($n = 10$ seeds), paired t-tests are computed against the full DynaMITE model. For OOD sweeps, Holm–Bonferroni corrected p-values are used. Only 3 of 42 pairwise OOD comparisons survive correction at $p_{\text{adj}} < 0.05$; most ranking claims are therefore directional rather than statistically confirmed.

5 Results

5.1 Nominal In-Distribution Comparison

Table 2 presents the mean $\pm$ standard deviation of cumulative reward across five training seeds and four tasks under deterministic 100-episode evaluation. LSTM achieves the best reward on all four tasks with the lowest variance ($p < 0.03$, paired t-test; Table 3). DynaMITE ranks second on three of four tasks (Figure 2).

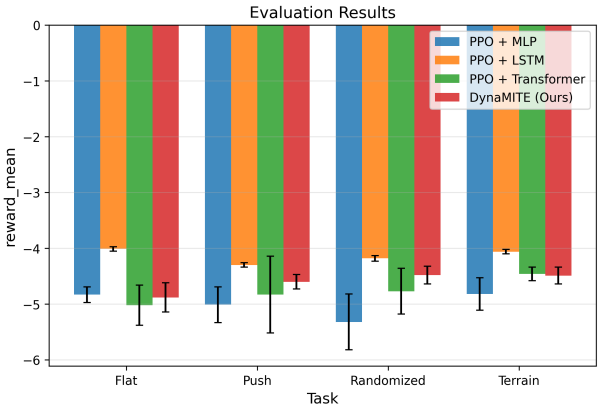

Figure 2: In-distribution reward (5 seeds, deterministic evaluation). LSTM achieves the best reward on all tasks; DynaMITE ranks second on three of four tasks but is significantly worse than LSTM on all.

Table 2: In-distribution reward comparison (5 seeds, deterministic eval). Bold indicates best per task. All LSTM vs. DynaMITE paired t-tests are significant ($p < 0.03$).

Method	Flat	Push	Rand.	Terrain
MLP	−4.83	−5.01	−5.32	−4.82
LSTM	**−4.01**	**−4.30**	**−4.18**	**−4.06**
Transf.	−5.02	−4.83	−4.77	−4.46
DynaMITE	−4.88	−4.60	−4.48	−4.49

Table 3: Paired t-tests: LSTM vs. DynaMITE (matched training seeds, $n = 5$).

Task	Mean Diff	t	p
Flat	+0.870	8.49	0.0011
Push	+0.305	4.69	0.0094
Randomized	+0.303	3.34	0.029
Terrain	+0.435	8.55	0.0010

5.2 Combined-Shift Stress Test

To simulate the compounding dynamics mismatch characteristic of sim-to-real transfer, we simultaneously shift friction, push magnitude, and action delay across five severity levels. Table 4 reports reward; Table 5 reports tracking error.

Table 4: Combined-shift reward across severity levels (randomized task, 5 seeds, 50-episode deterministic eval per level). Bold indicates best at each level.

Method	L0 (ID)	L1	L2	L4
DynaMITE	−4.38	−4.47	−4.50	**−4.63**
LSTM	**−3.56**	**−4.23**	**−4.40**	−5.12
Transf.	−4.67	−4.73	−4.72	−4.86
MLP	−5.68	−5.61	−5.55	−5.64

Table 5: Combined-shift tracking error across severity levels (randomized task, 5 seeds, 50-episode deterministic eval per level).

Method	L0	L1	L2	L3	L4
DynaMITE	2.15	2.56	3.18	4.23	6.13
LSTM	**1.31**	**2.19**	**3.01**	4.59	6.94
Transf.	2.08	2.58	3.17	4.35	6.48
MLP	2.24	2.76	3.40	4.47	6.30

Figure 3 and Table 6 quantify severe OOD degradation. Among the two highest-performing ID models, LSTM degrades substantially more than DynaMITE under combined shift (16.7% vs. 2.3% relative to ID reward). DynaMITE's mean reward matches or exceeds LSTM's from severity level 3 onward; at level 4, LSTM's tracking error (6.94) is the highest among all models, reversing its low-perturbation advantage. Note, however, that the Transformer and MLP baselines also show low sensitivity values (0.20 and 0.13, respectively)—in MLP's case because it performs poorly across all levels rather than because it resists degradation.

Note that the combined-shift Level 0 rewards (e.g., LSTM −3.56) differ from the main-comparison ID rewards in Table 2 (LSTM −4.18): Level 0 pins all dynamics to nominal values (friction = 1.0, no push, no delay), while the main comparison evaluates under the full domain randomization distribution. LSTM benefits disproportionately from nominal conditions ($\Delta = 0.62$) compared with DynaMITE ($\Delta = 0.10$), which itself is consistent with higher sensitivity to dynamics variation.

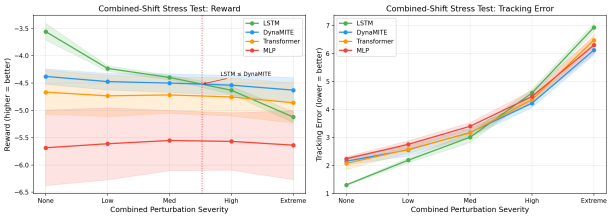

Figure 3: Combined-shift stress test (randomized task, 5 seeds). LSTM achieves the best reward at low severity but degrades steeply; DynaMITE's reward is lower at baseline but more stable. The crossover occurs at severity level 3. Neither model dominates across all levels.

Table 6: Severe OOD degradation (combined-shift sweep, randomized task, 5 seeds, deterministic eval). Degradation = |Severe − ID|; percentages relative to |ID|.

Model	ID	Severe	Worst	Degrad.
DynaMITE	−4.48	−4.58	−4.63	**2.3%**
LSTM	−4.18	−4.88	−5.12	16.7%
Transf.	−4.77	−4.81	−4.86	0.8%
MLP	−5.32	−5.66	−5.68	6.4%

5.3 Pareto Analysis

Aggregating across randomized-task OOD sweeps (combined-shift, push magnitude, friction), Table 7 and Figure 4 show that DynaMITE has the highest mean severe OOD reward in our sample (−4.58) and best worst-case (−4.78), while LSTM achieves the best ID reward (−4.14). No model dominates both axes under our protocol.

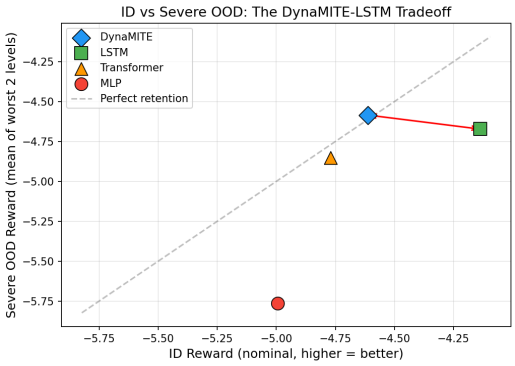

Figure 4: Pareto front: in-distribution reward vs. severe OOD reward. No model dominates both axes. LSTM achieves the best ID reward; DynaMITE has the highest mean severe OOD reward.

Table 7: Pareto analysis (5 seeds, deterministic eval): ID reward = 4-task mean; Severe OOD Mean = average across three randomized-task sweeps.

Model	ID	Severe OOD	Worst
DynaMITE	−4.61	**−4.58**	**−4.78**
LSTM	**−4.14**	−4.67	−5.12
Transf.	−4.77	−4.85	−5.08
MLP	−5.00	−5.76	−5.97

5.4 Push Recovery

In a controlled push-recovery protocol on flat terrain, the robot walks for 30 steps (steady-state), receives an exact-magnitude push in a random direction, and recovers over 40 steps. Recovery is defined as the tracking error returning below 1.5. Each condition uses 50 episodes per magnitude per seed across 5 seeds. Table 8 reports recovery time; Table 9 reports peak tracking error.

Table 8: Steps to recover command tracking after push (flat task, 5 seeds × 50 episodes per magnitude). Bold indicates fastest.

Push	DynaMITE	LSTM	Transf.	MLP
1.0	**5.6**	9.2	6.7	6.3
2.0	**5.9**	15.9	8.1	7.6
3.0	**6.0**	20.6	8.3	7.8
4.0	**6.0**	19.7	8.3	8.1
5.0	**6.1**	19.8	8.0	8.2
6.0	**6.1**	19.5	8.1	8.7
8.0	**6.2**	19.6	8.3	8.3

DynaMITE returns below the tracking-error threshold (1.5) in approximately 6 steps regardless of push magnitude (5.6–6.2, nearly constant). LSTM requires 9 to 20 steps to reach the same threshold as push magnitude grows, then plateaus; it is the slowest of all four models under this metric. At pushes ≥ 3 m/s, DynaMITE crosses the threshold 3.4× faster than LSTM.

Table 9: Peak tracking error after push (flat task, 5 seeds × 50 episodes). Bold indicates lower peak error.

Push (m/s)	DynaMITE	LSTM
1.0	3.96 ± 0.53	$\mathbf{3.05 \pm 0.14}$
2.0	4.03 ± 0.37	$\mathbf{3.32 \pm 0.08}$
3.0	4.36 ± 0.39	$\mathbf{3.78 \pm 0.08}$
4.0	5.01 ± 0.26	$\mathbf{4.62 \pm 0.07}$
5.0	5.76 ± 0.22	$\mathbf{5.50 \pm 0.11}$
6.0	6.74 ± 0.17	$\mathbf{6.41 \pm 0.08}$
8.0	8.45 ± 0.15	$\mathbf{8.22 \pm 0.12}$

However, LSTM achieves lower peak tracking error at all magnitudes (Table 9). Post-push reward is also substantially better for LSTM (−3.2 to −4.8 vs. DynaMITE's −9.5 to −9.8). The result is narrower than "better recovery": under our protocol and adopted threshold, DynaMITE returns below the tracking criterion fastest but with worse peak displacement and worse overall locomotion quality; LSTM achieves the best gait but crosses the threshold slowest.

5.5 Single-Axis OOD Sweeps

Table 10 presents the push magnitude sweep on the randomized task. DynaMITE matches or exceeds LSTM from push level 4 (3–5 m/s). Among these two models, LSTM degrades more steeply with increasing push magnitude (sensitivity 1.52 vs. 0.41).

Table 10: Push magnitude sweep (randomized task, 5 seeds, 50-ep eval per level). Bold indicates best.

Method	0–0.5	0.5–1	1–2	2–3	3–5	5–8	S.
DynaMITE	−4.37	−4.44	−4.50	−4.56	**−4.63**	**−4.78**	0.41
LSTM	**−3.58**	**−4.05**	**−4.23**	**−4.45**	−4.70	−5.09	1.52
Transf.	−4.65	−4.72	−4.76	−4.81	−4.94	−5.08	0.42
MLP	−5.65	−5.64	−5.79	−5.75	−5.89	−5.97	**0.33**

Table 11 summarizes severe OOD degradation across all sweeps and tasks.

Table 11: Severe OOD degradation across all sweep types and tasks (5 seeds). DynaMITE degrades less than LSTM on all push-related sweeps; crossover = level where DynaMITE first matches or exceeds LSTM.

Sweep	Task	DynaMITE	LSTM	Cross.
Combined	rand.	**2.3**%	16.7%	L3
Push mag.	rand.	**5.1**%	17.1%	L4
Push mag.	push	**4.4**%	12.2%	L5
Push mag.	terrain	**6.6**%	21.5%	L5
Friction	rand.	−0.4%*	1.5%	None

* Negative degradation indicates slight improvement under OOD conditions (within noise).

Figure 5 presents the full OOD sweep comparison. LSTM degrades more steeply than DynaMITE across all three tasks for push magnitude sweeps: sensitivity 1.52 (randomized), 1.41 (push), 1.61 (terrain) vs. DynaMITE's 0.39–0.46.

Only 3 of 42 pairwise OOD comparisons survive

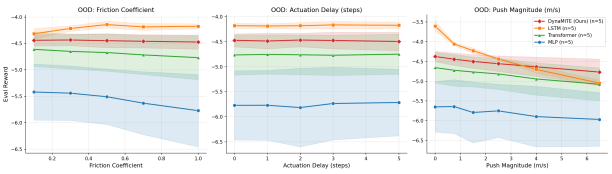

Figure 5: OOD sweep comparison across four models and five seeds. LSTM degrades more steeply than DynaMITE under push magnitude perturbation across all three tasks.

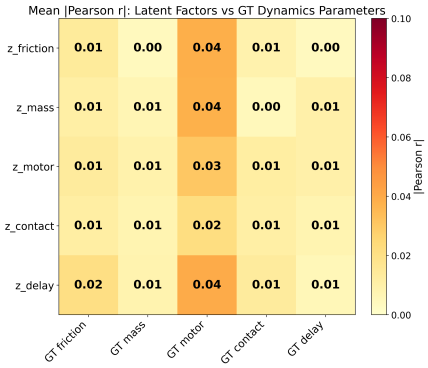

Figure 6: Latent correlation heatmap showing the Pearson correlation between each factor subspace dimension and the corresponding ground-truth dynamics parameter.

Holm–Bonferroni correction ($p_{\text{adj}} < 0.05$). Most ranking claims are directional, not statistically confirmed at $n = 5$.

5.6 Probing and Intervention Do Not Support Decodable Factor Structure

5.6.1 Factor Alignment

On a custom within-factor correlation metric (chance level = 0.20 for five factors), DynaMITE achieves a mean alignment score of 0.500 ± 0.020 across three seeds. Figure 6 shows the factor–subspace structure.

5.6.2 Intervention Analysis (Negative Result)

Table 12 presents the reward change when clamping each factor subspace. All $|\Delta\text{reward}| < 0.05$ across 3 seeds $\times$ 5 factors $\times$ 3 DR levels (90

evaluations total). No evidence of factor-specific behavioral control.

Table 12: Latent intervention analysis: average absolute reward change per factor subspace (3 seeds × 5 factors × 3 DR levels = 90 conditions, 50-ep eval).

Factor (dims)	Avg |ΔReward|	Interpretation
Friction (0–3)	0.007	Negligible
Mass (4–9)	0.012	Negligible
Motor (10–15)	0.021	Negligible
Contact (16–19)	0.020	Negligible
Delay (20–23)	0.020	Negligible

5.6.3 Latent Probe Analysis

We train Ridge regression (linear) and MLP (nonlinear) probes to predict ground-truth dynamics parameters from each model's internal representation (DynaMITE's 24-d factored latent vs. LSTM's 128-d hidden state), using ∼36,000 samples per run with 5-fold cross-validation (Table 13).

Table 13: Latent probe R^2 for DynaMITE vs. LSTM (5 seeds, 5-fold CV, ∼36k samples). Bold indicates best per factor.

Factor	DM Lin.	DM MLP	LSTM Lin.	LSTM MLP
Friction	−0.000	−0.001	0.026	**0.101**
Mass	0.000	−0.002	0.012	**0.018**
Motor	0.000	−0.002	0.010	**0.015**
Contact	0.000	−0.000	0.006	**0.045**
Delay	0.000	−0.000	0.011	**0.041**
Overall	**0.000**	**−0.001**	**0.013**	**0.044**

Despite being trained with per-factor auxiliary losses, DynaMITE's latent shows $R^2 \approx 0$ for both linear and nonlinear probes across all five factors. LSTM's hidden state, with no auxiliary signal, achieves R^2 up to 0.101 (friction, MLP probe). However, absolute R^2 values are low for both models (LSTM overall: 0.044).

5.7 Ablation Study

To assess the contribution of each architectural component, we expand the ablation study to $n = 10$ seeds (42–51) per variant on the randomized task. Table 14 reports the results. The **Single Latent** variant—which replaces the five-factor partition with a single 24-d unfactored latent trained with a monolithic auxiliary loss—reaches statistical significance ($p = 0.048$), with 8 of 10 seeds degrading (mean $\Delta = -0.23$). No Latent shows consistent degradation (9/10 seeds worse, $\Delta = -0.18$) but does not reach significance ($p = 0.225$) due to high variance. No Aux Loss remains directionally negative but inconsistent (6/10 seeds worse, $p = 0.641$). The **Aux Only** variant—auxiliary losses active but the policy sees only the full 128-d transformer features—performs identically to No Latent (−4.64 vs. −4.64, $p = 0.251$).

Table 14: Ablation study on the randomized task (10 seeds, paired t-test vs. full DynaMITE). Bold p-value indicates $p < 0.05$.

Variant	Reward	Δ	p	Worse
Full	**−4.46**	—	—	—
No Aux	−4.51	−0.05	0.641	6/10
No Latent	−4.64	−0.18	0.225	9/10
Single	−4.69	−0.23	**0.048**	8/10
Aux Only	−4.64	−0.18	0.251	7/10

5.7.1 Factorial: Bottleneck vs. Aux Loss

To decompose whether the observed reward difference comes from the tanh bottleneck, the auxiliary losses, or both, we add a fourth ablation cell—**Aux Only**: the latent head and per-factor auxiliary losses are active (providing gradient regularization to the transformer), but the policy and value heads see only the full 128-d trans-

former features, not the 24-d compressed bottleneck. Table 15 presents the 2×2 factorial.

Table 15: 2×2 factorial: bottleneck vs. auxiliary loss (10 seeds, randomized task). Each cell averages 10 paired seeds.

	No Aux Loss	**Aux Loss**
Bottleneck	-4.51 ± 0.20	$\mathbf{-4.46 \pm 0.25}$
No Bottleneck	-4.64 ± 0.36	-4.64 ± 0.34

Factorial main effects (paired t-tests, $n = 10$): the auxiliary losses show **no measurable effect** on ID reward ($+0.03$, $p = 0.732$); the bottleneck shows a **consistent advantage** ($+0.16$, $p = 0.207$); the interaction is negligible ($+0.05$, $p = 0.787$).

5.7.2 Factorial: OOD Robustness

To test whether the bottleneck's advantage amplifies under distribution shift, we evaluate all four 2×2 cells on **combined-shift severe** (Levels 3–4). Table 16 presents the severe OOD results.

Table 16: 2×2 factorial under severe combined-shift (avg. of Levels 3–4, 10 seeds, 50-ep eval per level).

	No Aux Loss	**Aux Loss**
Bottleneck	-4.59 ± 0.21	$\mathbf{-4.55 \pm 0.22}$
No Bottleneck	-4.68 ± 0.28	-4.66 ± 0.23

OOD factorial main effects: auxiliary losses show **no measurable effect** ($+0.034$, $p = 0.669$); the bottleneck advantage **persists but does not amplify** ($+0.100$, $p = 0.208$); interaction is absent ($+0.015$, $p = 0.912$). Table 17 shows that all four variants are remarkably robust under severe perturbation (degradation 0.5–2.1%), and that bottleneck models degrade slightly *more* than no-bottleneck variants.

Table 17: Factorial degradation analysis (10 seeds, randomized task). Degradation = |Severe − ID|/|ID|.

Variant	**ID**	**Severe**	**Δ**	**Degrad.**
Full (B+A)	−4.46	−4.55	−0.09	2.1%
No Aux (B)	−4.51	−4.59	−0.08	1.9%
No Latent	−4.64	−4.68	−0.04	0.9%
Aux Only (A)	−4.64	−4.66	−0.02	0.5%

5.8 Summary of Evidence

Table 18 summarizes the key claims and their evidence status.

Table 18: Summary of key claims and evidence status.

Claim	Evidence	Status
Latent decodable?	Probe $R^2 \approx 0$; MIG/DCI/SAP ≈ 0	**No**
Factor-specific effect?	$\lvert\Delta r\rvert < 0.05$	No evidence
Aux heads learn targets?	MSE ≈ 3.7 (48% drop)	Not accurately
LSTM best ID?	$p < 0.03$, all 4 tasks	**Yes**
DynaMITE lower OOD sens.?	0.25 vs. 1.57	Directional
DynaMITE faster recovery?	$\sim$6 vs. 9–21 steps	Directional
Regularization mechanism?	Gradient orth., low rank, MI	Consistent
Bottleneck drives reward?	$+0.16$ ($p = 0.207$)	Consistent
Bottleneck drives OOD?	$+0.10$ ($p = 0.208$)	Consistent

6 Mechanistic Analysis

To investigate potential mechanisms behind DynaMITE's less pronounced OOD degradation despite the absence of a decodable dynamics representation, we examine four aspects of the training and representation: gradient flow, representation geometry, mutual information, and standard disentanglement metrics.

6.1 Gradient Flow Analysis

We retrain DynaMITE (3 seeds, 10M steps each) with gradient instrumentation, computing separate gradient vectors for the PPO objective and each of the five auxiliary factor losses at every 10th iteration ($\sim$81 data points per seed). Table 19 reports the cosine similarity between PPO and auxiliary gradients.

Table 19: Cosine similarity between PPO and auxiliary gradients (3 seeds, averaged over training).

Factor	Mean $\cos(\nabla_{\mathbf{PPO}}, \nabla_{\mathbf{aux}})$	Std
Friction	-0.001	0.005
Mass	-0.002	0.010
Motor	-0.005	0.016
Contact	-0.008	0.026
Delay	-0.002	0.009

All cosine similarities are indistinguishable from zero ($\lvert\cos\rvert < 0.01$), as shown in Figure 7. The auxiliary gradients are orthogonal to the PPO gradient throughout training. They contribute approximately 20–40% of the total gradient norm but in orthogonal directions, consistent with a regularization-like effect rather than direct optimization synergy.

Auxiliary head convergence. The total auxiliary MSE drops from 7.2 ± 0.7 at the start of training to 3.7 ± 0.1 at convergence (5 seeds), a 48% reduction. However, the residual MSE remains high (per-factor average ≈ 0.75), indicating that the auxiliary heads do not learn to accurately predict their target dynamics parameters.

6.2 Representation Geometry

We collect $\sim$36,000 representations per model/seed (200 episodes $\times$ 32 environments) from trained checkpoints and compute SVD-based geometry metrics (Table 20).

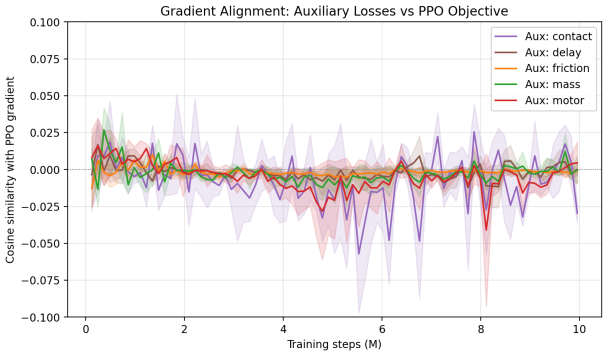

Figure 7: Cosine similarity between PPO and auxiliary gradients over training (3 seeds). All five factor-specific auxiliary gradients remain orthogonal to the PPO gradient ($|\cos| < 0.01$) throughout the entire training run.

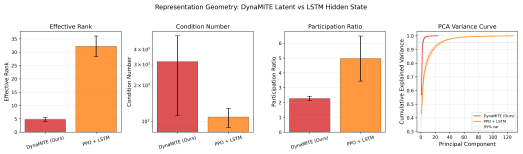

Figure 8: Representation geometry. DynaMITE's 24-d latent collapses to effective rank ∼5; LSTM's 128-d hidden state uses ∼32 effective dimensions.

Table 20: Representation geometry comparison (5 seeds, SVD on ∼36k representations per model).

Metric	DynaMITE (24-d)	LSTM (128-d)
Effective rank	$\mathbf{4.78 \pm 0.72}$	32.20 ± 3.85
Participation ratio	$\mathbf{2.27 \pm 0.15}$	4.96 ± 1.52
Condition number	315.9 ± 204.1	$\mathbf{108.4 \pm 19.9}$

DynaMITE's latent is much lower-dimensional in practice: effective rank ∼5 out of 24 possible dimensions, compared with LSTM's ∼32 out of 128. The high condition number (316 vs. 108) indicates a highly anisotropic structure.

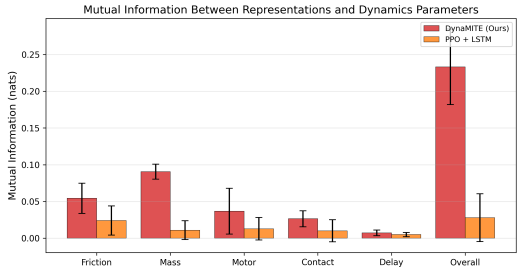

Figure 9: Mutual information between learned representations and ground-truth dynamics factors (KNN estimation). DynaMITE retains more MI with dynamics parameters than LSTM (0.233 vs. 0.028 nats), but absolute values for both models are low.

6.3 Mutual Information Estimation

We estimate mutual information (MI) between learned representations and ground-truth dynamics parameters using KNN estimation (`sklearn mutual_info_regression`, $k = 5$). Table 21 reports the results.

Table 21: Mutual information between representations and dynamics factors (5 seeds, KNN estimation $k = 5$, ∼36k samples, nats).

Factor	DynaMITE (nats)	LSTM (nats)
Overall	$\mathbf{0.233 \pm 0.052}$	0.028 ± 0.033
Friction	$\mathbf{0.054 \pm 0.021}$	0.024 ± 0.020
Mass	$\mathbf{0.091 \pm 0.010}$	0.011 ± 0.013
Motor	$\mathbf{0.037 \pm 0.031}$	0.013 ± 0.015
Contact	$\mathbf{0.026 \pm 0.011}$	0.010 ± 0.015
Delay	0.007 ± 0.004	0.005 ± 0.003

DynaMITE's latent contains more MI with dynamics parameters than LSTM's hidden state (0.233 vs. 0.028 nats overall). However, all MI values are small in absolute terms (< 0.25 nats).

6.4 Standard Disentanglement Metrics

We compute MIG [36], DCI [37], and SAP [38] on the same latent representations (Table 22).

Table 22: Standard disentanglement metrics (5 seeds). Lower DCI-I is better (prediction error).

Metric	DynaMITE	LSTM
MIG	0.0013	0.0013
DCI Disent.	0.0195	**0.0929**
DCI Complete	0.0168	**0.0542**
DCI Info.	**0.0846**	0.1397
SAP	0.0001	0.0006

All scores are very low for both models—neither architecture produces cleanly disentangled representations. The near-zero MIG and SAP for both models confirm that no individual latent dimension aligns with a single dynamics factor.

6.5 Mechanistic Synthesis

We separate the findings into four categories:

Observed facts. (i) Auxiliary and PPO gradients are orthogonal throughout training ($|\cos| < 0.01$). (ii) The 24-d latent collapses to effective rank $\sim$5. (iii) The compressed latent retains more MI with dynamics than LSTM's hidden state (0.233 vs. 0.028 nats), though absolute MI is low for both. (iv) Neither representation is probe-decodable. (v) Standard disentanglement metrics (MIG, DCI, SAP) are near zero for both models; LSTM is $\sim 5\times$ more disentangled per DCI.

Plausible interpretation. The auxiliary losses are consistent with a regularization-like effect that constrains features into a low-dimensional subspace. This compression could contribute to less OOD degradation by limiting the representational capacity available for overfitting to the training distribution, but this interpretation is speculative.

Not established. Whether compression causally mediates the observed OOD sensitivity difference. The 2×2 factorial identifies the tanh bottleneck—not the auxiliary losses—as the component driving the reward advantage, but neither main effect reaches significance ($p \approx 0.2$). Architectural capacity differences and optimization landscape effects remain viable alternative explanations.

7 Discussion

7.1 Observed Operating Regimes

The results suggest an ID–OOD tradeoff under our evaluation protocol (Table 23).

Table 23: Observed patterns under our simulation protocol.

Scenario	Favored	Evidence
Nominal ID reward	LSTM	Confirmed
Moderate mismatch	LSTM	Directional
Severe mismatch	DynaMITE	Directional
Fast re-adaptation	DynaMITE	Directional
Inspect latent	Neither	Confirmed

7.2 Practical Takeaway

- **For nominal reward:** LSTM remains the strongest choice. It achieves the best ID reward on all tasks and degrades gracefully under moderate perturbation.
- **For severe OOD robustness:** DynaMITE shows less degradation than LSTM under combined shift (2.3% vs. 16.7%), but the 2×2 factorial attributes this to the bottleneck compression, not the auxiliary supervision.
- **For interpretability:** Auxiliary dynamics supervision does not produce latent representations useful for online dynamics monitoring or debugging.
- **Mechanism:** The bottleneck is the primary component driving observed reward differences. The auxiliary losses show no measurable contribution when the bottleneck is held constant ($p > 0.66$).

7.3 Why Does LSTM Degrade More Under Combined Shift?

LSTM degrades more than DynaMITE under combined perturbation. The 2×2 factorial clarifies this: within the DynaMITE architecture, the bottleneck shows a consistent advantage under severe combined shift ($+0.10$, $p = 0.208$), while the auxiliary losses show none ($+0.03$, $p = 0.669$). Crucially, bottleneck models degrade slightly *more* than no-bottleneck variants (2.1% vs. 0.5%, Table 17), confirming the advantage is a training-time benefit that carries through to OOD rather than a specific robustness mechanism. The larger DynaMITE-vs.-LSTM gap (2.3% vs. 16.7%) therefore reflects architectural differences between the two model families, not the auxiliary supervision.

8 Limitations

- **Partially deconfounded comparison.** The 2×2 factorial identifies the bottleneck as the primary component, but neither main effect reaches significance ($p \approx 0.2$).
- **Limited statistical power.** Only 3 of 42 pairwise OOD comparisons survive Holm–Bonferroni correction at $n = 5$. Most OOD claims are directional.
- **Probe family.** Our probes are Ridge regression and small MLPs (1 hidden layer, 64 units). More expressive decoders might recover information our probes miss.
- **Sensitivity metric.** The OOD sensitivity metric ($\max - \min$) does not distinguish genuine robustness from uniformly poor performance.
- **Simulation only.** All experiments use Isaac Lab. Hardware transfer is untested.
- **Deterministic evaluation only.** Stochastic policy behavior is unreported.
- **Narrow reward spread.** The top two models span $\sim$0.6 reward units; practical significance for deployment is unknown.
- **Incomplete perturbation coverage.** Mass, contact stiffness, and observation noise are untested as single-axis sweeps.

What this paper does not claim. This paper does not show that decodable or disentangled latent dynamics are impossible in locomotion—only that one specific supervision strategy did not produce them in this setup. It does not establish real-world transfer. It does not claim that auxiliary losses are without value; they may act as useful regularizers even when they fail to produce interpretable latent structure.

9 Future Work

- **Sim-to-real validation.** Hardware deployment on a Unitree G1 to test whether the OOD robustness tradeoff transfers.
- **Extending the factorial to push recovery.** The 2×2 factorial has not yet been applied to push-recovery protocols. Additionally, varying bottleneck width could further clarify the role of compression.
- **Advanced disentanglement methods.** Nonlinear ICA or manifold-aware probes may recover structure invisible to standard metrics.
- **Increased statistical power.** Expanding seeds beyond $n = 10$ (e.g., $n = 20$) with bootstrap confidence intervals.

10 Conclusions

We evaluated whether factor-wise auxiliary dynamics supervision produces decodable or functionally separable latent structure in simulated humanoid locomotion. Under the probes, interventions, and disentanglement metrics adopted here, it does not: probes yield $R^2 \approx 0$, clamping produces negligible reward change, and standard metrics are near zero. An unsupervised LSTM hidden state achieves higher probe accuracy.

A 2×2 factorial ablation ($n = 10$) isolates the contributions of the tanh bottleneck and auxiliary losses. The observed performance differences are

attributable primarily to the bottleneck compression, not the auxiliary supervision: the auxiliary losses show no measurable effect on either ID reward (+0.03, $p = 0.732$) or severe OOD reward (+0.03, $p = 0.669$), while the bottleneck yields a consistent advantage in both regimes (ID: +0.16, $p = 0.207$; OOD: +0.10, $p = 0.208$). The bottleneck's advantage persists under severe perturbation but does not amplify, and bottleneck models degrade slightly more than no-bottleneck variants—confirming a training-time representation benefit rather than a robustness-specific mechanism.

For practitioners: LSTM remains the strongest choice when nominal performance is the priority. Auxiliary dynamics supervision does not produce an interpretable estimator and does not measurably improve reward beyond what the bottleneck alone provides. The value of multi-component architectures like DynaMITE, if any, lies in the implicit compression imposed by the bottleneck, not in the explicit supervision signal.

Data Availability

All training scripts, evaluation code, configuration files, and raw results are publicly available at https://github.com/fjkrch/g1-factorized-latent-locomotion. No pretrained checkpoints are provided; all models must be trained from scratch using the released code and configurations. Full reproduction from source requires approximately 35 hours on a single NVIDIA RTX 4060 GPU.

Acknowledgments

During the preparation of this manuscript, the author used GitHub Copilot (powered by Claude, Anthropic) for the purposes of drafting and editing selected sections. The author has reviewed and edited the output and takes full responsibility for the content of this publication.

References

[1] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid Motor Adaptation for Legged Robots," in *Proc. Robotics: Science and Systems (RSS)*, 2021.

[2] N. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik, "Adapting Rapid Motor Adaptation for Bipedal Robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2022, pp. 1757–1764.

[3] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2019, pp. 8973–8979.

[4] A. Anand, E. Racah, S. Ozair, Y. Bengio, M.-A. Côté, and R. D. Hjelm, "Unsupervised State Representation Learning in Atari," in *Proc. NeurIPS*, 2019, pp. 8766–8779.

[5] R. Agarwal, M. C. Machado, P. S. Castro, and M. G. Bellemare, "Contrastive Behavioral Similarity Embeddings for Generalization in Reinforcement Learning," in *Proc. ICLR*, 2021.

[6] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *Proc. IEEE/RSJ IROS*, 2017, pp. 23–30.

[7] OpenAI et al., "Solving Rubik's Cube with a Robot Hand," *arXiv:1910.07113*, 2019.

[8] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *Proc. IEEE ICRA*, 2018, pp. 3803–3810.

[9] E. Parisotto et al., "Stabilizing Transformers for Reinforcement Learning," in *Proc. ICML*, 2020, pp. 7487–7498.

[10] J. Hwangbo et al., "Learning Agile and Dynamic Motor Skills for Legged Robots," *Sci. Robot.*, vol. 4, eaau5872, 2019.

[11] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid Locomotion via Reinforcement Learning," in *Proc. RSS*, 2022.

[12] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild," *Sci. Robot.*, vol. 7, eabk2822, 2022.

[13] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning," in *Proc. RSS*, 2021.

[14] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots," in *Proc. IEEE ICRA*, 2021, pp. 2811–2817.

[15] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-Real Learning of All Common Bipedal Gaits via Periodic Reward Composition," in *Proc. IEEE ICRA*, 2022, pp. 7309–7315.

[16] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Learning Humanoid Locomotion with Transformers," in *Proc. ICLR*, 2024.

[17] T. Haarnoja et al., "Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning," *Sci. Robot.*, vol. 9, eadi8022, 2024.

[18] B. Mehta, M. Diaz, F. Golber, M. Pfeiffer, A. E. Eiben, and D. Stork, "Active Domain Randomization," in *Proc. CoRL*, 2020, pp. 1162–1176.

[19] F. Muratore et al., "Robot Learning from Randomized Simulations: A Review," *Front. Robot. AI*, vol. 9, 799893, 2022.

[20] Q. Vuong, S. Vikram, H. Su, S. Gao, and H. I. Christensen, "How to Pick the Domain Randomization Parameters for Sim-to-Real Transfer of Reinforcement Learning Policies?" *arXiv:1903.11774*, 2019.

[21] N. Heess et al., "Emergence of Locomotion Behaviours in Rich Environments," *arXiv:1707.02286*, 2017.

[22] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-Aware Legged Locomotion Using Reinforcement Learning and Optimal Control," *IEEE Trans. Robot.*, vol. 38, pp. 2908–2927, 2022.

[23] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the Unknown: Learning a Universal Policy with Online System Identification," in *Proc. RSS*, 2017.

[24] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged Locomotion in Challenging Terrains using Egocentric Vision," in *Proc. CoRL*, 2023.

[25] L. Chen et al., "Decision Transformer: Reinforcement Learning via Sequence Modeling," in *Proc. NeurIPS*, 2021, pp. 15084–15097.

[26] M. Laskin et al., "In-Context Reinforcement Learning with Algorithm Distillation," in *Proc. ICLR*, 2023.

[27] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "AMP: Adversarial Motion Priors for Stylized Physics-Based Character Control," *ACM Trans. Graph.*, vol. 40, pp. 1–20, 2021.

[28] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning Quadrupedal Locomotion over Challenging Terrain," *Sci. Robot.*, vol. 5, eabc5986, 2020.

[29] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," in *Proc. CoRL*, 2022, pp. 91–100.

[30] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Prentice Hall, 1999.

[31] C. G. Atkeson, C. H. An, and J. M. Hollerbach, "Estimation of Inertial Parameters of Manipulator Loads and Links," *Int. J. Robot. Res.*, vol. 5, pp. 101–119, 1986.

[32] D. Hafner et al., "Learning Latent Dynamics for Planning from Pixels," in *Proc. ICML*, 2019, pp. 2555–2565.

[33] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images," in *Proc. NeurIPS*, 2015, pp. 2746–2754.

[34] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables," in *Proc. ICML*, 2019, pp. 5331–5340.

[35] L. Zintgraf et al., "VariBAD: A Very Good Method for Bayes-Adaptive Deep RL via Meta-Learning," in *Proc. ICLR*, 2020.

[36] R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud, "Isolating Sources of Disentanglement in Variational Autoencoders," in *Proc. NeurIPS*, 2018, pp. 2610–2620.

[37] C. Eastwood and C. K. I. Williams, "A Framework for the Quantitative Evaluation of Disentangled Representations," in *Proc. ICLR*, 2018.

[38] A. Kumar, P. Sattigeri, and A. Balakrishnan, "Variational Inference of Disentangled Latent Concepts from Unlabeled Observations," in *Proc. ICLR*, 2018.

[39] I. Higgins et al., "DARLA: Improving Zero-Shot Transfer in Reinforcement Learning," in *Proc. ICML*, 2017, pp. 1480–1490.

[40] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep Reinforcement Learning that Matters," in *Proc. AAAI*, 2018, pp. 3207–3214.