Multi-Embodiment Locomotion at Scale with extreme Embodiment Randomization

Nico Bohlinger[1], Jan Peters[1,2]

***Abstract*— We present a single, general locomotion policy trained on a diverse collection of 50 legged robots. By combining an improved embodiment-aware architecture (URMAv2) with a performance-based curriculum for extreme Embodiment Randomization, our policy learns to control millions of morphological variations. Our policy achieves zero-shot transfer to unseen real-world humanoid and quadruped robots.**

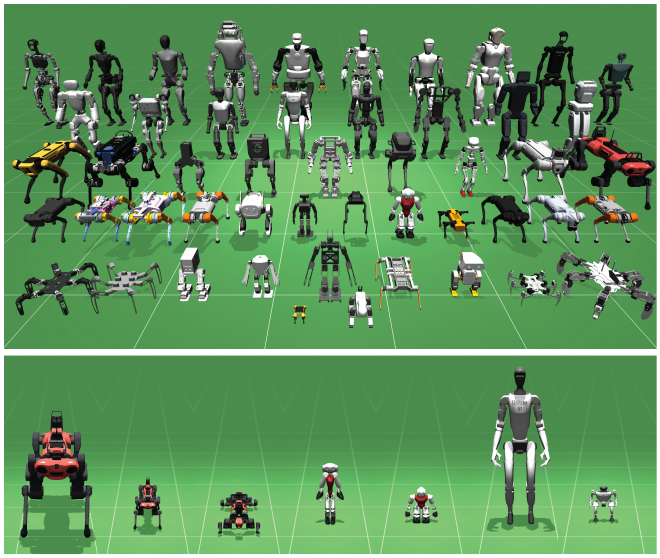

Fig. 1. (Top) We collected a diverse set of 50 legged robots, including 15 quadrupeds, 23 humanoids, 8 bipeds and 4 hexapods. (Bottom) We train the policy on all robots simultaneously using 25600 parallel environments. The performance-based curriculum on extreme Embodiment Randomization leads to the policy seeing gradually more challenging embodiments throughout training. This results in a curriculum of up to 10 million different embodiments per training run. Here different generated varations of the ANYmal C, Nao v5 and Unitree G1 are shown as examples.

I. INTRODUCTION

Recent advances and availability of powerful robot hardware, like humanoid robots, have enabled researchers all around the world to tackle more complex tasks in robotics [1], [2]. Deep Reinforcement Learning (DRL) has shown impressive results in many of these tasks, especially in the field of locomotion [3], [4]. With more and more robot platforms being developed and finding their way into research labs and real-world applications, the current paradigm of training a control policy tailored to a specific robot can become increasingly inefficient. Robot platforms change, adapt, and evolve over time, but many current training approaches do not consider robot morphologies as a key factor. Their learning process is agnostic or simply unaware of the specific characteristics and capabilities of the robot's embodiment, making cross-embodiment transfer difficult or even impossible. We build upon the recently introduced Unified Robot Morphology Architecture (URMA) [5], an embodiment-aware learning framework that addresses these challenges for the field of robot locomotion. We train a single unified embodiment-aware policy across 50 different legged robots with massive Embodiment Randomization (ER). This results in a curriculum of up to 10 million embodiments per training run, in order to learn a robust and adaptive general locomotion policy, that can be directly zero-shot transferred to unseen humanoid and quadruped robots in the real world.

[1]Technical University of Darmstadt. [2]hessian.AI. & German Research Center for AI (DFKI). & Robotics Institute Germany (RIG).

II. RELATED WORK

Robot locomotion has seen significant advancements in recent years, particularly with the rise of DRL techniques [3], [4], [6]. Leveraging fast and highly parallelizable simulators, like Issac Gym/Sim [7] or MuJoCo XLA (MJX) [8], in combination with strong Domain Randomization (DR), and the scalability of on-policy DRL algorithms, like Proximal Policy Optimization (PPO) [9], has enabled learning robust and high-performing locomotion policies for various quadruped and humanoid robots. Techniques such as DR [10] and student-teacher learning [4] are used to bridge the sim-to-real gap and other methods, like curriculum learning, help to speed up and stabilize the training process [11]. While training directly on the real robot system would be ideal to make the policy fully aware of the true capabilities of its embodiment, it is often impractical due to safety concerns, wear and tear, and the extensive time required for the unparallelized training [12].

In the pursuit of obtaining a foundation model for robotics tasks, like locomotion, that can generalize across different embodiments, the concept of embodiment-aware learning and the technical challenge of different amounts of sensors and actuators (meaning different observation and action spaces in the language of DRL) come into play. It is natural to consider the field of Multi-Task Reinforcement Learning (MTRL) here, where a single policy is trained to solve multiple different tasks. However, many existing MTRL approaches simply zero-pad observations and actions or learn different input and output heads for each task, which can severely limit their ability to generalize across different embodiments, as they neglect their structural similarities [5]. Therefore, prior work has proposed Graph Neural Networks (GNNs) as an architecture to better capture the structure of robot embodiments [13]. Following work has used Trans-

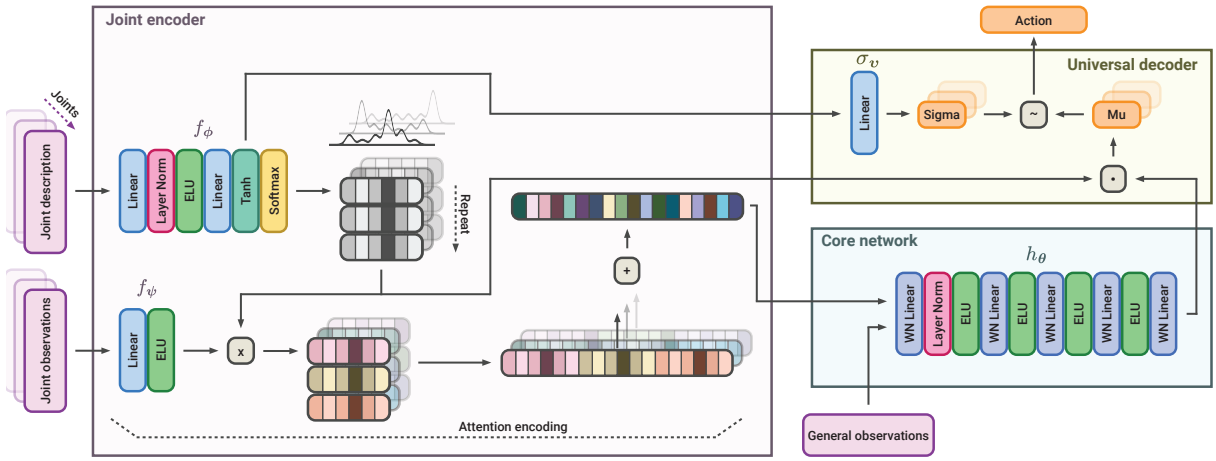

Fig. 2. Overview of URMAv2. We extend the original URMA architecture to improve its scalability, learning stability and empirical performance in the massively multi-embodiment setting. We increase the capacity of the encoder and core network, add WeightNorm layers for more stable training, and replace the original universal decoder with a streamlined attention-based decoding mechanism.

formers in combination with different graph-based features or attention mechanisms to improve the scalability in the multi-embodiment setting [14]. Only recently, the URMA [5] could show the applicability of multi-embodiment learning to real-world robots. While the initial study only used 16 robots and could not show sim-to-real transfer to seen or unseen humanoids, following work has shown embodiment scaling laws for training URMA on up to 1000 offline generated robots (based on three template morphologies) and demonstrated its transfer to a real humanoid [15].

III. METHOD

We build upon the original URMA architecture and training framework, and scale it to the massively multi-embodiment setting of 50 different base robots with 10 million variations. To achieve this, we modify the neural network architecture to be larger, more stable and leverage the attention mechanism also for the policy output (see Figure 2), which we call Unified Robot Morphology Architecture v2 (URMAv2). Furthermore, we introduce a performance-based curriculum learning strategy in combination with extreme ER to expose the policy to gradually more diverse and difficult embodiments.

A. URMAv2 Architecture

Inputs: Following the original URMA architecture, the inputs are split into three categories: per-joint description vectors $\{d_j\}_{j\in\mathcal{J}}$ for the set $\mathcal{J}$ of all joints in a given robot that uniquely describe a joint's static properties (e.g., rotation axis, torque limits), per-joint observations $\{o_j\}_{j\in\mathcal{J}}$ containing dynamic state information (e.g., position, velocity), and general robot observations $\mathbf{o}_g$ (e.g., trunk velocity, gravity vector). URMAv2 includes an additional per-joint observation to indicate whether a joint should track its nominal position or can be controlled freely, allowing for task-specific joint-level conditioning. Also, we remove the feet-specific observations and their encoding from the policy network, as contact sensors are not available on many of the considered robots. The critic network keeps the feet observations and now also receives the noise-free observations to better estimate values.

Joint Encoder: URMAv2 keeps the same attention-based joint encoder, which processes each joint's description (attention keys) and observation vector (attention values), and aggregates them into the combined joint latent vector

$$\bar{z}_{\text{joints}} = \sum_{j\in\mathcal{J}} z_j, \quad z_j = \alpha_j f_\psi(o_j), \\ \alpha_j = \frac{\exp(f_\phi(d_j)/\tau)}{\sum_{L_d}\exp(f_\phi(d_j)/\tau)}. \tag{1}$$

where f_ϕ (with latent dimension L_d) and f_ψ are the encoders for the joint descriptions and joint observations, respectively, and τ is the learnable temperature parameter of the softmax. URMAv2 uses a wider Multilayer Perceptron (MLP) for f_ψ (2x 256 units) for the policy network to increase its capacity for the larger number of robots and variations.

Core Network: The joint latent vector contains all joint information in a fixed-size vector, so it can be concatenated with the fixed-size general observations and processed by the core network h_θ to generate the action latent vector

$$\bar{z}_{\text{action}} = h_\theta(o_g, \bar{z}_{\text{joints}}). \tag{2}$$

URMAv2 uses a deeper stack of 5 instead of 3 hidden layers to increase the model capacity. To stabilize training, WeightNorm layers [16] are used around every Dense layer, which decompose the weights as $w = g\frac{v}{\|v\|_2}$, where the optimizer acts on g, a learnable scalar, and v, representing the raw weights.

Action Decoder: The most significant architectural change is the replacement of URMA's universal decoder with an attention-based decoding mechanism. Instead of concatenating the action latent vector in batch with encoded joint description latents to produce actions for every joint, URMAv2 computes the mean action μ_j for each joint via a simple dot product between the action latent vector $\bar{\mathbf{z}}_{\text{action}}$

and the corresponding joint's attention weights α_j that were generated in the encoder:

$$\mu_j = \overline{\mathbf{z}}_{\text{action}} \cdot \alpha_j \tag{3}$$

Also, the per-joint standard deviations are predicted with a linear layer σ_v from the same joint description encoding calculated for the attention weights, which results in actions being sampled from

$$a_j \sim \mathcal{N}(\mu_j, \sigma_v(f_\phi(d_j))). \tag{4}$$

This creates a streamlined architecture that is both conceptually simpler, computationally more efficient and empirically more performant than the original URMA decoder.

B. Embodiment Randomization

To improve the generalization capabilities of the policy across different embodiments, we apply extreme ER online during training (see Figure 1). ER differs from standard DR in that all the generated values are seen by the policy through the description vectors, allowing the policy to condition and adapt to them. We use DR after the ER sampling to further modify the sampled parameters but keep them hidden from the policy, to add robustness and improve sim-to-real transfer. Our ER includes scaling of: body part size and position in every dimension, coupled and decoupled mass and inertia, center of mass, inertia and body part and joint axis orientation, IMU position, motor torque and velocity and position limits, joint damping and friction and armature and stiffness, joint nominal position, PD gains, and action scaling factor. Our framework samples a new embodiment during every episode step with a probability of 0.2% which corresponds to once every 10 seconds of simulated time on average at the highest curriculum level. This leads to up to 10 million different embodiments per training run.

C. Performance-based Curriculum

When drastically increasing the number of robots and their variations, the learning problem becomes significantly more challenging. To tackle this, we introduce a performance-based curriculum learning strategy that attaches every component of the learning framework to a single curriculum coefficient $\beta \in [0, 1]$. This coefficient is initialized to $\beta = 0$ and is increased by $n\Delta\beta$ whenever an episode is deemed successful, e.g., a minimum tracking error, episode length, or return threshold is reached. n is the number of consecutive successful or unsuccessful episodes, depending on whether β should be increased or decreased, and allows the curriculum to quickly adapt to the current performance of the policy. $\Delta\beta$ is a small constant step size that determines the granularity of the curriculum. We attach all training components: domain and embodiment randomization ranges, perturbations, sampling probabilities, terrain attributes, termination conditions, and reward penalty coefficients, to this single curriculum coefficient β. This significantly helps to speed up the training process for challenging embodiments, especially humanoids, and leads to more stable training runs.

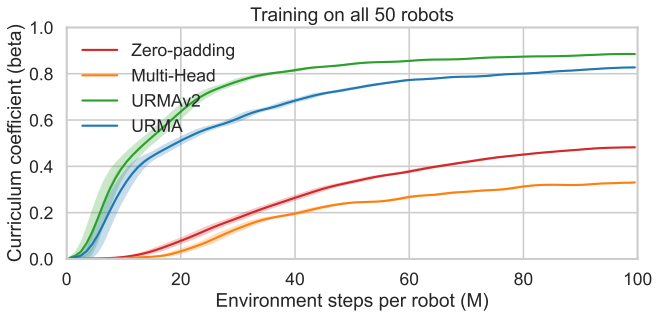

Fig. 3. Comparison of the training performance of URMA, URMAv2, zero-padding and multi-head baselines when training on all 50 robots.

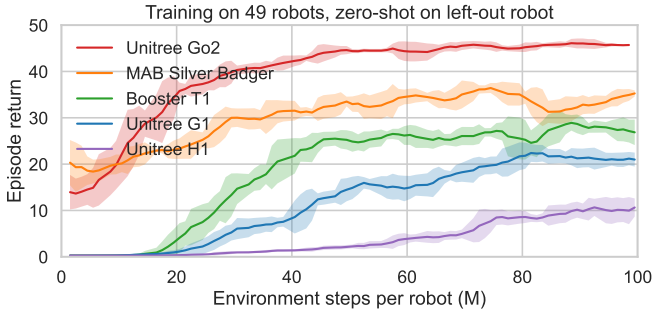

Fig. 4. Different setups of URMAv2 trained on 49 robots and zero-shot performance evaluated on the left-out robot. We test on the MAB Silver Badger and Unitree Go2 for quadrupeds, and the Unitree H1, Unitree G1 and Booster T1 for humanoids.

Furthermore, we define all mentioned components in a percentage-based manner based on the robot's nominal parameters from the URDF, which allows us to use the exact same parameters, ranges and reward coefficients for all robots. For any given robot, only the nominal joint positions and the PD gains have to be specified, the rest of the training framework is fully shared between all robots.

IV. EXPERIMENTS

We train URMAv2 on a set of 50 legged robots, including 15 quadrupeds, 23 humanoids, 8 bipeds and 4 hexapods, collected from various freely available URDFs (see Figure 1). We use MJX as the physics engine and PPO as the DRL algorithm, which we implement with the RL-X library [17]. With a total of 25600 parallel environments (512 per robot), we collect 1.6 million samples (32768 per robot) and use 16 minibatches of size 102400 (2048 per robot) for 10 epochs for every policy update. We train for a total of 5 billion environment steps (100 million per robot), which takes approximately 40 hours on a single NVIDIA A100 GPU.

Figure 3 compares the training performance of URMAv2 with the original URMA architecture, as well as a zero-padding and multi-head baseline (one head per robot). We measure the performance by the average curriculum coefficient β over all robots. URMA and URMAv2 significantly outperform the zero-padding and multi-head baselines, showing the effectiveness of the embodiment-aware architecture in

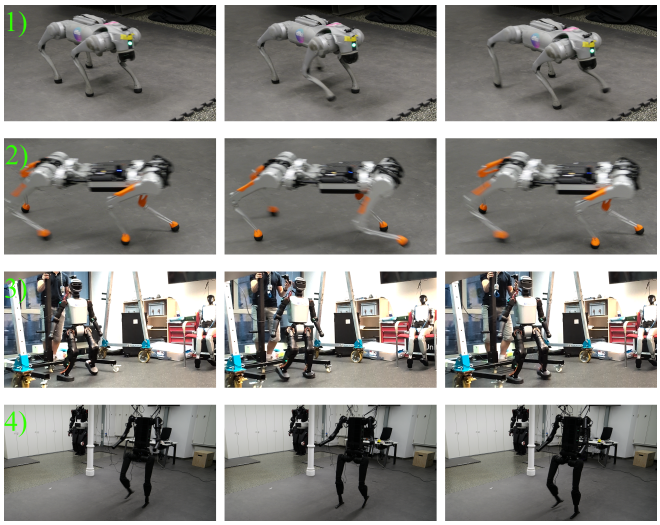

Fig. 5. Shows the zero-shot transfer of URMAv2 to the Unitree Go2 (row 1), MAB Silver Badger (row 2), and Booster T1 (row 3). We also deploy the policy trained on all 50 robots on the Unitree H1 (row 4).

general. URMAv2 outperforms URMA in terms of learning speed and final performance ($\beta = 0.88$ vs. $\beta = 0.82$).

Figure 4 shows the zero-shot transfer performance of URMAv2 trained on 49 robots and evaluated on the left-out robot. For quadrupeds, we test on the Unitree Go2 and MAB Silver Badger (has an additional spine joint), and for humanoids, we test on the Unitree H1, Unitree G1 and Booster T1. URMAv2 shows strong zero-shot performance on especially the quadruped robots, even the MAB Silver Badger, which proved to be challenging in the original URMA study due to its additional spine joint that is not present in any other training robot. Zero-shot performance for the humanoids is clearly lower, but still a significant improvement over the reported 0 return for the Unitree H1 in the original URMA study. In simulation, the policy is able to control all three humanoids fairly well, especially the Booster T1 and the Unitree G1, but under server perturbations or really rough terrain, it still falls occasionally.

A. Sim-to-Real Transfer

While inspecting the learned policy in simulation can give a good indication of its performance, the ultimate test is its transfer to the real robots. Figure 5 shows the zero-shot sim-to-real transfer of URMAv2 trained on 49 robots to the Unitree Go2, MAB Silver Badger, and Booster T1. The policy is able to control both quadrupeds very well, even under disturbances like pushes and pulling on the legs. The policy is able to walk forward and sidewards reliably on the Booster T1, but struggles with turning and walking backwards, leading to regular falls. We could not zero-shot transfer to the Unitree H1 as the policy was not stable enough, but we transfered to URMAv2 policy trained on all 50 robots, which was able to locomote well on the H1 in every direction.

V. CONCLUSION

We presented URMAv2, an improved embodiment-aware architecture for learning a general locomotion policy across a diverse set of 50 legged robots with extreme ER and a performance-based curriculum. While URMAv2 shows strong training performance and zero-shot transfer to unseen quadruped and humanoid robots in simulation, sim-to-real transfer to unseen humanoids still remains challenging. Even more embodiment diversity through more base robots and curriculum learning methods that can explore the embodiment space more effectively could help to improve the generalization capabilities to obtain a true foundation model for robot locomotion.

ACKNOWLEDGMENT

This research is funded by the National Science Centre Poland (Weave programme UMO-2021/43/I/ST6/02711), and by the German Science Foundation (DFG) (grant number PE 2315/17-1).

We gratefully acknowledge support from the hessian.AI Service Center (funded by the Federal Ministry of Education and Research, BMBF, grant no. 01IS22091) and the hessian.AI Innovation Lab (funded by the Hessian Ministry for Digital Strategy and Innovation, grant no. S-DIW04/0013/003).

We acknowledge EuroHPC Joint Undertaking for awarding us access to MareNostrum5 at BSC, Spain.

REFERENCES

[1] Y. Ma, A. Cramariuc, F. Farshidian, and M. Hutter, "Learning coordinated badminton skills for legged manipulators," *Science Robotics*, vol. 10, no. 102, p. eadu3922, 2025.

[2] Z. Su, B. Zhang, N. Rahmanian, Y. Gao, Q. Liao, C. Regan, K. Sreenath, and S. S. Sastry, "Hitter: A humanoid table tennis robot via hierarchical planning and learning," *arXiv preprint arXiv:2508.21043*, 2025.

[3] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning*. PMLR, 2023, pp. 22–31.

[4] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *RoboLetics: Workshop on robot learning in athletics @ CoRL*, 2023.

[5] N. Bohlinger, G. Czechmanowski, M. Krupka, P. Kicki, K. Walas, J. Peters, and D. Tateo, "One policy to run them all: an end-to-end learning approach to multi-embodiment locomotion," *Conference on Robot Learning*, 2024.

[6] M. Stasica, A. Bick, N. Bohlinger, O. Mohseni, M. J. A. Fritzsche, C. Hübler, J. Peters, and A. Seyfarth, "Bridge the gap: Enhancing quadruped locomotion with vertical ground perturbations," *International Conference on Intelligent Robots and Systems*, 2025.

[7] 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*, vol. 8, no. 6, pp. 3740–3747, 2023.

[8] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

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

[10] 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 *International conference on intelligent robots and systems*, 2017.

[11] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *International journal of robotics research*, vol. 43, no. 4, pp. 572–587, 2024.
[12] N. Bohlinger, J. Kinzel, D. Palenicek, L. Antczak, and J. Peters, "Gait in eight: Efficient on-robot learning for omnidirectional quadruped locomotion," *International Conference on Intelligent Robots and Systems*, 2025.
[13] T. Wang, R. Liao, J. Ba, and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," in *International conference on learning representations*, 2018.
[14] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei, "Metamorph: learning universal controllers with transformers," in *International Conference on Learning Representations*. ICLR, 2022.
[15] B. Ai, L. Dai, N. Bohlinger, D. Li, T. Mu, Z. Wu, K. Fay, H. I. Christensen, J. Peters, and H. Su, "Towards embodiment scaling laws in robot locomotion," *Conference on Robot Learning (CoRL)*, 2025.
[16] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *Advances in neural information processing systems*, vol. 29, 2016.
[17] N. Bohlinger and K. Dorer, "Rl-x: A deep reinforcement learning library (not only) for robocup," in *Robot world cup*. Springer, 2023, pp. 228–239.