Sampling Strategies for Robust Universal Quadrupedal Locomotion Policies

David Rytz [1], Kim Tien Ly[1], and Ioannis Havoutis[1]

[1]Dynamic Robot Systems, Oxford Robotics Institute, University of Oxford

***Abstract*— This work focuses on sampling strategies of configuration variations for generating robust universal locomotion policies for quadrupedal robots. We investigate the effects of sampling physical robot parameters and joint proportional-derivative gains to enable training a single reinforcement learning policy that generalizes to multiple parameter configurations. Three fundamental joint gain sampling strategies are compared: parameter sampling with (1) linear and polynomial function mappings of mass-to-gains, (2) performance-based adaptive filtering, and (3) uniform random sampling. We improve the robustness of the policy by biasing the configurations using nominal priors and reference models. All training was conducted on RaiSim, tested in simulation on a range of diverse quadrupeds, and zero-shot deployed onto hardware using the ANYmal quadruped robot. Compared to multiple baseline implementations, our results demonstrate the need for significant joint controller gains randomization for robust closing of the sim-to-real gap.**

I. Introduction

The increasing maturity of quadrupedal robotic systems has led to a wide range of applications, including industrial inspection, surveillance, research, and search and rescue operations. Advances in numerical optimization [1], [2], Reinforcement Learning (RL) [3], [4], and combined [5] control approaches made essential contributions to the development of quadrupedal locomotion and navigation in complex environments with increased agility.

Most existing control designs are tailored to specific robots, optimized for a single platform, and require extensive retraining and parameter tuning whenever the robot's morphology or dynamics change. These platform-specific approaches pose a fundamental scalability challenge: developing a controller for each new quadruped requires notable computational resources, laborious manual parameter tuning and reward engineering, as well as hardware expertise. As the diversity of commercial quadrupeds continues to grow – from lightweight such as the A1 (12 kg, [6]) to larger ANYmal (50 kg, [7]) – such individualized design architectures become increasingly impractical.

In this paper, we identify three central challenges for achieving *universal* quadrupedal locomotion. First, quadrupeds exhibit substantial variation in mass distribution, joint configurations, actuator properties, and kinematic structures, which hinders the design of policies that generalize across platforms. Second, these morphological differences directly affect locomotion dynamics, requiring different gait patterns and control strategies, and thereby complicating controller design, hardware transfer, and performance evaluation. Third, a suitable robot parameter generation approach is needed to support efficient and stable training across a wide range of configurations, despite the absence of known optimal proportional-derivative (PD) gain schedules for previously unseen robots.

Corresponding author: rytz@robots.ox.ac.uk. Video can be found here.

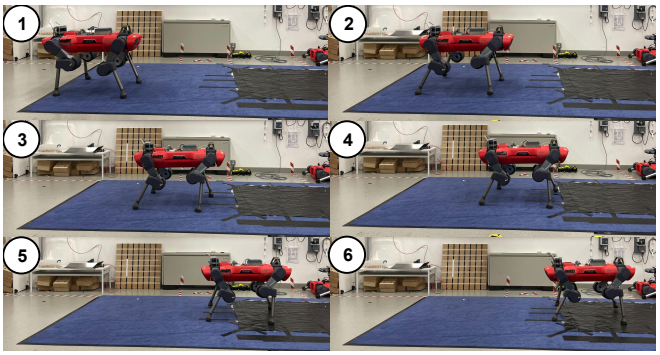

Fig. 1: Example run of ANYmal hardware used. Stand still at frames 1 and 6.

To address these challenges, we investigate two approaches to sampling robot configurations and compare three strategies for PD gain tuning. As demonstrated in the supplementary video, strong sim-to-sim performance does not guarantee reliable hardware behavior. We validate our methods in simulation on a range of quadrupeds and in hardware experiments on the ANYmal platform (Fig. 1), using models that were not encountered during training.

Our contributions include: (1) a novel sampling method of morphologies for a *universal* RL locomotion policy with a wide range of kinematics and dynamics randomization, (2) improving on state-of-the-art sampling methods for joint PD gain tuning and comparison on how these parameters affect locomotion performance, and (3) successful zero-shot transfer to various quadrupeds in simulation (ranging from the A1, with 12 kg, to ANYmal, with 50 kg) and on hardware, including the ANYmal robot as shown in Fig. 1.

II. Related Work

A. Reinforcement Learning for Legged Locomotion

Deep RL has emerged as a powerful paradigm for developing robust and agile controllers for legged robots, with early success including policies trained for dynamic and agile quadrupedal skills [8] and enabling robust rough terrain

locomotion in indoor and outdoor settings [9], [10]. More recent works include highly dynamic parkour maneuvers [4], wheeled-legged locomotion [11], as well as whole-body multi-legged control [12].

Standard RL pipelines often rely on extensive reward shaping, long training times, and remain sensitive to morphology-specific variations to achieve natural and smooth locomotion motions. Motion imitation, based on animal motion capture [13], partial demonstrations [14], adversarial motion priors [15], or optimal control-based reference motions generated by a trajectory optimizer [16] have become a common strategy to acquire mammal-like smooth gaits with reduced reward engineering.

Another major challenge with RL policy is sim-to-real transfer, mostly addressed by using one of the two strategies: (1) applying domain randomization during training to increase robustness [17], or (2) increasing simulation fidelity via accurate system modelling [18] and identification [8]. While the latter approach aligns dynamics in simulation with reality, domain randomization provides general robustness by exposing the learned policy to a range of varying parameters such as ground and motor friction, latency, and other robot or environment parameters. Balancing training stability and speed with deployability of the resulting policy requires careful tuning of the noise injection.

B. Universal Locomotion Control

Building upon locomotion controllers for single robots, recent research has shifted toward *universal* controllers that generalize across different robot morphologies, actuation, and sensing setups. One major line of work incorporates morphology into the policy architecture, often using Graph Neural Networks (GNN) [19] or transformers [20], [21] to represent robot structure. While these methods demonstrate promising transfer capabilities, most have been validated primarily in simulation, with limited real-world results. Meta-RL methods provide another perspective, with techniques such as Model-Agnostic Meta-Learning (MAML) [22] enabling policies to adapt rapidly to new morphologies [23]. These approaches aim for few-shot adaptation and, thus, will not be further considered in this work, which aims at zero-shot deployment onto hardware.

A critical factor in enabling universal locomotion is the sampling strategy used to expose policies to diverse morphologies during training. Prior controller architectures that were deployed on hardware differ significantly in this respect:

- **GenLoco** [24] utilizes imitation learning based on motion capture data combined with a large state-action history to train a phase-variable locomotion controller. They use a fixed set of reference robot models and uniformly randomized morphology parameters such as masses and link sizes, mapping the resulting total robot mass to joint PD gains through a linear function with uniform scaling. Additional domain randomization is applied to link inertias, control latency, and actuator properties.
- **ManyQuadrupeds** [25] relied on Central Pattern Generators (GPG) for controls but handled morphology diversity and robot models similarly to GenLoco. Using CPGs necessitates this control architecture to rely on robot-specific inverse kinematics. No domain randomization is applied.
- **MorAL** [26] trained a morphology encoder to embed robot-specific information into the locomotion policy to discover joint action signals based off a single robot reference model. Next to domain randomization of body masses and link sizes, they also randomize motor gains, latency, and friction parameters. To further improve system modelling during training, an actuator network is used.
- **URMA** [27] employs a transformer-based policy across bipeds, quadrupeds, and hexapods by encoding robot-specific observations (joints and feet) with an attention mechanism using joint description vectors. A universal decoder then maps the latent representation back to joint actions. URMA uses a fixed set of robots and applies domain randomization through uniform sampling around robot-specific nominal values, along with observation noise. Compared to other works [26], [28], its sampling ranges are relatively narrow, which limits robustness despite strong simulation and hardware results.
- **PAL** [28], building on MorAL, incorporates a broader range of robot models. This enables deployment of a universal locomotion policy on ANYmal (50 kg), demonstrating robustness beyond small to medium-sized platforms. Robot parameters, including PD gains, are sampled using uniform sampling within pre-defined parameter ranges.

These approaches have been shown to function effectively on small to medium-sized robots (below 30 kg), though often with reduced smoothness and stability as robot size increases. PAL [28] demonstrated that broader model diversity is critical for scaling locomotion policies to larger hardware.

Building on this observation, we introduce a novel morphology parameter sampling method based on particle filter techniques, inspired by their use in terrain generation [29]. Our approach enables policies to generalize across a broader spectrum of robot dynamics and kinematics, leading to more robust training and zero-shot transfer to quadrupeds in simulation and on hardware.

III. Preliminaries - Reinforcement Learning

We formulate the locomotion problem as a Markov Decision Process in an RL setting. At each discrete time step t the agent observes the state $\mathbf{s}$ and takes action $\mathbf{a}$ transitioning to state $\mathbf{s_{i+1}}$ with transition probability $\mathcal{P}(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}_{i+1})$. The agent receives a scalar reward $\mathrm{r}_i = \mathcal{R}(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}_{i+1})$ with -$\mathcal{R}$ being the reward function. Starting from an initial state s_0, this process continues until a terminal condition is reached, such as a finite time horizon or success/failure. The objective that the agent is maximizing is the expected cumulative discounted

reward:

$$J(\pi) \doteq \mathop{\mathrm{E}}_{\mathcal{T}\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t R(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\right], \tag{1}$$

where $\gamma \in [0,1)$ is the discount factor and $\mathcal{T}$ denotes the trajectory induced by policy π and $\pi : \mathbf{s} \rightarrow a$ is a parametrized mapping from states to actions. In this work, π represents a *universal* locomotion controller, mapping observations to actions such that quadrupeds with diverse morphological properties achieve consistent and stable behavior.

IV. Methodology

The following section details the proposed pipeline illustrated in Fig. 2, which includes the architectures of the dynamics encoding network (shown in red), the control policy (shown in blue), and the generation of morphology and control parameters for training (shown in green).

A. Universal Locomotion Controller

The control framework used was built following the asymmetric actor critic architecture first presented in MorAL [26].

The input states for each module (red for estimation and blue for actor and critic), as per Fig. 2 in our control architecture, are:

1) **Estimator:** By not using imitation learning, the architecture necessitates estimating base linear velocity $\mathbf{v}^*_{\mathrm{x,y,z}}$. This network is extended to also observe robot-specific morphology parameters $\mathbf{o}^*_m$ consisting of base body centre of mass $\mathbf{com}_{\mathrm{base},x,y,z} \in \mathbb{R}^3$, body masses $\mathbf{m}_{\mathrm{base,\ hip,\ thigh,\ shank}} \in \mathbb{R}^4$, joint link offsets $\mathbf{c}_{q_1^{\mathrm{xyz}},q_2^{\mathrm{xyz}},q_3^{\mathrm{xyz}}} \in \mathbb{R}^9$ and foot offset $c_{f,z} \in \mathbb{R}^1$.
The network receives the robot state history $\mathbf{o}^H_t = \langle \mathbf{o}^h_{t=0}, \mathbf{o}^h_{t=1}, \ldots, \mathbf{o}^h_{t=5} \rangle$ as input. We define

$$\mathbf{o}^h_{t=0} = \langle \mathbf{e}^B_z, \omega_B, \Delta\mathbf{q_t}, \dot{\mathbf{q}}_\mathbf{t}, \mathbf{q}^{\mathrm{des}}_{\mathbf{t-1}}, \mathbf{v}^{\mathrm{des}}_{x,y,\omega} \rangle, \tag{2}$$

consisting of robot base orientation $\mathbf{e}^B_z \in \mathbb{R}^3$ and angular velocity $\omega_B \in \mathbb{R}^3$, joint position offset from nominal position $\Delta\mathbf{q_t} \in \mathbb{R}^{12}$, joint velocity $\dot{\mathbf{q}}_\mathbf{t} \in \mathbb{R}^{12}$, previous joint action $\mathbf{q}^{\mathrm{des}}_{\mathbf{t-1}} \in \mathbb{R}^{12}$, and desired velocity command $\mathbf{v}^{\mathrm{des}}_{x,y,\omega} \in \mathbb{R}^3$ as current robot state for $t = 0$.
2) **Actor:** The actor generates the desired joint position $\mathbf{q}^{\mathrm{des}}_j \in \mathbb{R}^{12}$ action by receiving the input $\mathbf{o}_a$ comprised of estimated base linear velocity $\mathbf{v}^*_{\mathrm{x,y,z}}$ and morphology parameters $\mathbf{o}^*_m$, the current robot state $\mathbf{o}^h_{t=0}$ and joint nominal position $\mathbf{q}^n \in \mathbb{R}^{12}$:

$$\mathbf{o}^a_t = \langle \mathbf{v}^*_{\mathrm{x,y,z}}, \mathbf{o}^h_{t=0}, \mathbf{q}^n, \mathbf{o}^*_m \rangle. \tag{3}$$

The action is sampled from a Gaussian distribution with zero mean and standard deviation $\sigma_a = 0.6$, added to the nominal joint configuration before passing it through the joint PD controller.
3) **Critic:** The critic receives additional privileged information that would be difficult to infer during hardware deployment, including foot contact states $\mathbf{c}^{f,i}_t \in \mathbb{R}^4$, feet contact friction $\mu^{f,i} \in \mathbb{R}^4$, feet height $\mathbf{h}^{f,i}_t$ and joint PD gains $\mathbf{k}_{PD} \in \mathbb{R}^{24}$. Instead of inputting the estimated morphology parameters, it processes the privileged parameters from the simulation environment, resulting in the input:

$$\mathbf{o}^v_t = \langle \mathbf{v}_{\mathrm{x,y,z}}, \mathbf{o}^h_{t=0}, \mathbf{q}^n, \mathbf{o}_m, \mathbf{c}^{f,i}_t, \mu^{f,i}, \mathbf{h}^{f,i}_t, \mathbf{k}_{PD} \rangle. \tag{4}$$

Compared to [26], [28], we included the centre of mass of the base robot body into our architecture and removed the height map information, reducing training time and simplifying the evaluation of this work.

B. Reward Structure

The reward is implemented following [28] with the weights used for the sum of its terms described in Table I:

Component	Weight	Component	Weight
Base linear velocity	3.0	Joint position	-0.2
Base angular velocity	1.5	Joint velocity	-3×10^{-4}
Base orientation	-5.0	Joint acceleration	-2×10^{-7}
Base height	-20.0	Joint torque	-3×10^{-5}
Base undesired motion	-0.5	Joint action smoothness	-0.12
Foot slip	-0.2	Joint action smoothness 2	-0.05
Air time	-6.0		

TABLE I: Reward term definitions after [28].

All policies are trained with the same reward structure and weights, thus resulting in a similar locomotion gait pattern.

The task for this controller structure is to follow a velocity command of dimension $\mathbb{R}^3$ expressed in base frame as $\mathbf{c}^{\mathrm{des}} = [\mathrm{v}^{\mathrm{des}}_x \mathbf{e}^B_x \quad \mathrm{v}^{\mathrm{des}}_y \mathbf{e}^B_y \quad \omega^{\mathrm{des}}_z \mathbf{e}^B_z]^T$. The command during training is uniformly sampled within the ranges $v^{\max}_x = \pm 1\,\mathrm{m/s}$, $v^{\max}_y = \pm 0.75\,\mathrm{m/s}$, and $\omega^{\max}_z = \pm 1.5\,\mathrm{rad/s}$ for durations of 3 s to 6 s.

C. Joint controller

As depicted in Fig. 2 on the top in green, we use a Proportional-Derivative (PD) controller for each joint $j \in \{1,\ldots,12\}$ to regulate its desired position $\mathrm{q}^{\mathrm{des}}_j$ output from the actor. The controller outputs the demanded torque τ^{cmd}_j which is then applied to the actuators as

$$\tau^{\mathrm{cmd}}_j = K_p(\mathrm{q}^{\mathrm{des}}_j - \mathrm{q}_j) - K_d\dot{\mathrm{q}}_j. \tag{5}$$

with K_p and K_d being the proportional and derivative gains. During training, we randomize between the joint PD controller in the form of Equation (5) and the readily available Raisim physics simulator's built-in joint PD controller [30]. As we could not guarantee joint torque limits to be enforced during training with the Raisim PD controller, we clip τ_j between the actuator torque limits $[-\tau^{max}_j, \tau^{max}_j]$.

D. Robot Configuration Generation and Randomization

1) Quadruped reference models: Following [28], we utilize four simplified reference models Unitree's A1 [6] and Aliengo quadrupeds, as well as ANYbotics' ANYmal B [31] (an older and smaller version) and ANYmal C [32]. Their nominal values are used to resample a new set of parameters $\mathbf{c}_i$ to generate 40 robot configurations per robot model by randomizing their kinematic and dynamic properties.

A generic quadruped is modeled as a set of rigid bodies b_{AB} with mass $m_{AB} \in \mathbb{R}^+$, connected by either fixed or

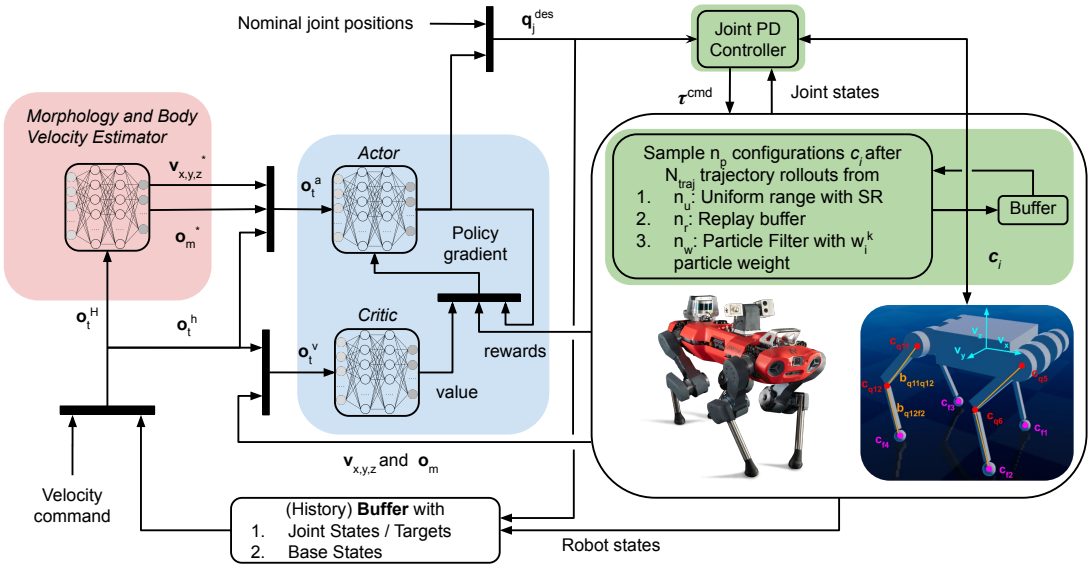

Fig. 2: Extending the frameworks of [26], [28], we propose the following pipeline. A buffer stores the base states, joint states, and actions, which are passed to the morphology and base linear velocity estimator (red). These estimates are then input to the actor to produce a control action $\mathbf{q}^{des}$ (blue), enabling quadruped locomotion through a joint PD controller. Candidate morphologies and actuator parameters (green) are trained and assessed using our model generation procedure on nominal robot models (grey, bottom right; Section IV-D). The final actor is deployed on the ANYmal platform.

revolute joints, where A and B denote parent and child joints, respectively. The base body b_{base} is treated as a floating joint, and the relative placement of child joints is defined by $c_B \in \mathbb{R}^3$. Each foot is assigned a friction coefficient μ_f and a local frame offset $c_{f,z}$. Leg geometries are sampled from two distinct configurations: type A (knees pointing backward) or type X (knees pointing toward the base).

2) Robot model parameter randomization: Based on the *reference quadrupeds*, we uniformly sample the morphology parameters as described in more detail in [28], showing zero-shot deployment onto systems such as the ANYmal. The approach described for the estimator in Section IV-A is extended by the base center of mass $\text{com}_{\text{base,x}}$ sampled uniformly $\mathcal{U}(-0.15, 0.15)$ and $\text{com}_{\text{base,y,z}}$ as $\mathcal{U}(-0.1, 0.1)$ added to $\mathbf{c}_i$ for all reference models. Suppose that a robot can stand collision-free in its nominal configuration for 2 s using the RaiSim [33] simulator's joint controller. In that case, we admit it into the set of randomized robots used for training.

We investigate the following strategies for sampling n_p robot configurations with parameters $\mathbf{c}_i$ during training after $N_{\text{traj}} = 40$ sec simulated steps:

- **Uniform randomization:** n_u configurations are sampled from uniform distributions. This simple strategy ensures broad coverage but does not adapt sampling to training progress. We resample n_r configurations from a replay buffer of previously sampled configurations. This approach has been the state-of-the-art parameter generation for GenLoco, ManyQuadrupeds, MorAL, URMA, and PAL, with differences in sampling strategies for joint PD gains.
- **Performance-based curriculum:** We generate n_{SR} configurations from the robot nominal parameters c_i^{nom} by clipping the uniform sampling ranges (SR) of the **uniform randomization** to $SR \in (0.1, 1)$ with $SR = 0.1$ for initialization. Whenever the policy exceeds a task-specific threshold, we adjust the sampling range to account for the different difficulty of the training. By comparison, we set $SR = 1$ for uniform randomization in GenLoco, ManyQuarupeds, MorAL, and PAL, and SR = small for URMA at all times. We define the instantaneous linear and zero velocity thresholds inspired by the terrain-based traversability [29]:

$$v^{lin}(t) = \begin{cases} 1 & \text{if } |v_{pr}^{lin}| > 0.3 \cdot |\left[v_x^{\max},\ v_y^{\max}\right]| \\ 0 & \text{else} \end{cases}, \quad (6)$$

$$v^{zero}(t) = \begin{cases} 1 & \text{if } |\mathbf{v}_{x,y,\omega}^{\text{cur}}| < 0.2 \\ 0 & \text{else} \end{cases}. \quad (7)$$

where v_{pr}^{lin} is the scalar projection at time t between the state and commanded base linear velocity in the robot base frame. $|\mathbf{v}_{x,y,\omega}^{\text{cur}}|$ is the norm of the current base velocity during a zero velocity command. Given all robot configurations $\mathbf{c}_i$ and the current policy, we calculate over all training environments and timesteps the linear and zero velocity command tracking

$$Tr_i^{\text{cmd}} = \sum_{t}^{0,\dots,T_{\text{tot}}} \frac{v_i^{cmd}(t)}{T_i^{\text{tot}}} \quad (8)$$

where T_{tot} is the total count of time steps that a command was active for a given robot i after N_{traj}. The ideal policy would achieve a value of 1 for the tasks cmd $\in \{\text{lin}, \text{zero}\}$, while a failing policy converges to

0. Computing over all c_i, the mean $Tr^{\text{cmd}}_{\text{mean}}$, we then update the SR after N_{traj} as:

$$SR = \begin{cases} SR + 0.01, & \text{if } Tr^{\text{lin}}_{\text{mean}} > 0.65 \wedge Tr^{\text{zero}}_{\text{mean}} > 0.55, \\ SR - 0.01, & \text{if } Tr^{\text{lin}}_{\text{mean}} < 0.55 \vee Tr^{\text{zero}}_{\text{mean}} < 0.40, \\ SR, & \text{otherwise.} \end{cases} \tag{9}$$

This adaptive scheme increases the task difficulty in accordance with the agent's learning progress, avoiding overly hard or trivial training conditions $\mathbf{c}_i$.

- **Adaptive curriculum with particle filtering:** Inspired by [29], we maintain a distribution over parameter sets and update it through a Sequential Importance Resampling (SIR) particle filter. The importance weight for a given $\mathbf{c}_i$ is used to update n_w configurations using a random nearest neighbor walk and is defined as:

$$w_i^k = \frac{1}{2} \cdot \sum^{cmd} Tr_i^{\text{lin}} \in [0.4, 0.9] \wedge Tr_i^{\text{zero}} \in [0.4, 0.9]. \tag{10}$$

The performance ranges are chosen such that parameters yielding mid-to upper range performance are used only, ensuring that training emphasizes robot parameter configurations $\mathbf{c}_i$ that are neither too easy nor too difficult. Command tracking-like metrics guide desirability, and $n_{\text{r}} > 0$ prevents degeneration of the parameter set. Compared to the first two approaches, this method adaptively shapes the training distribution to maximize robustness and generalization.

Joint nominal configuration and joint torque limit parameters are kept at full uniform sampling range for all cases, enabling early adaptation to default standing poses and understanding of the resulting policies' torque limits.

3) Joint PD gains randomization: A central design choice in universal locomotion is how the robot joint PD gains c_i^{PD} are sampled and then selected for deployment. We evaluate several strategies inspired by prior work:

- **Mass-dependent mapping:** Similar to GenLoco [24] and MorAL [26], we parametrize the c_i^{PD} as a function of the robot's total mass. Both polynomial mapping (as MorAL) and linear function (GenLoco) are considered. The latter additionally scales c_i^{PD} with a uniformly sampled value between 0.7 and 1.1.
- **Uniform randomization:** Following the *uniform randomization* procedure in PAL [28], all $c_i^{\text{PD}} \in \mathbb{R}^2$ are drawn uniformly within SR = 1.
- **Interpolation with nominal sets:** Inspired by URMA [27], we interpolate between a set of predefined nominal gains associated with different robots and apply a limited uniformly sampled noise, ie, SR = small. For robots between these reference values, interpolation is applied jointly on both the nominal gains and their associated SR.
- **Performance-based adaptive particle filtering:** Following the computation of Equation (9) and (10), we update n_{SR}, n_w, and n_r configurations scaling SR of the joint PD gain parameters in an adaptive way.

The nominal PD gain values for MorAL, PAL, URMA, ManyQuadrupeds, and GenLoco are indicated with an asterisk in Table III or as a red cross in Figure 4.

4) Domain Randomization: For successful sim-to-real transfer of policies trained in simulation dynamics, randomization has been shown to be an important aspect [34], [35]. Many morphology and dynamics parameters are randomized through the robot generation described in Section IV-D. In any sampling case we additionally rescale for all j in $c_{i_j}^{\text{PD}} \in \mathbb{R}^{24}$ by $\tilde{c}_{i_j}^{\text{PD}} = c_{i_j}^{\text{PD}} \cdot \text{clip}\,(1+\varepsilon, 0.95, 1.05)$ where $\varepsilon \sim \mathcal{N}(0,1)$ is a standard normal variable scaling the PD gain, and $\text{clip}(\cdot)$ enforces lower and upper bounds. We introduce actuation command tracking delays and dynamics as in [28].

E. Training Setup - RL Algorithm

We train the locomotion policy using Proximal Policy Optimization (PPO) [36]. The locomotion policy π and the estimator policy are parameterized as a Multi-Layer Perceptron (MLP) with leaky-relu activation. The locomotion policy is modelled with hidden layers of size [512, 256, 128], and the estimator policy is modelled with layers of size [512, 256, 64]. We apply a penalty of -0.25 for early termination if a collision is detected as a self-collision or ground collision with a body other than a foot.

The training time and hyperparameters were hand-tuned based off [28] and are provided in Table II. We trained the policies using 8 CPU cores (@4 GHz) and an NVIDIA RTX 4090 GPU. The hyperparameters used for the algorithm choice are described in Table II with 50000 training steps taking approximately 34 h.

Parameter	Value	Parameter	Value
Discount factor, γ	0.997	Entropy coefficient	0
Learning rate	adaptive	Value coefficient	0.5
Batch size	60750	GAE	True
Mini-batch size	10	GAE λ	0.95
Epochs	6	Steps per iteration	135
Parallel envs, n_{env}	450	PPO time/iteration	$\sim$2.2 s

TABLE II: RL Training time and hyperparameters.

V. Simulation and Hardware Validation

In this section, we compare the effectiveness of our controllers on the locomotion task for quadrupedal robots with different kinematic and dynamic parameters.

A. Performance Benchmark

The comparative evaluation is performed among the following reference-free locomotion controllers for the flat terrain walking task. Unless noted otherwise, we compare the state-of-the-art uniform range sampling (SR = 1) with the performance-based particle filter approach (SR is set adaptively).

1) **Particle Filter, adaptive Sampling Range (Ours):** For the performance based $n_u = 10\% \cdot n_p$ samples we adjust the SR following Equation (9). Additionally, we resample $n_r = 10\% \cdot n_p$ replay buffer samples, and $n_w = 10\% \cdot n_p$ particle filter weighted samples as

per Equation (10). The PD gains are sampled on the available range clipped by SR.

2) **PAL/Uniform, Sampling Range = 1** [28]: We resample $n_r = 15\% \cdot n_p$ with SR $= 1$ and $n_r = \%15 \cdot n_p$. The PD gains are also sampled uniformly with SR $= 1$.
3) **GenLoco** [24]: The PD gains are sampled using a linear function mapping robot weight to joint PD gains and additional uniform noise range. The original work only trains with a fixed set of robots, whereas we enable larger unseen parameter ranges; thus, we interpolate between nominal values to better cover out-of-bounds robot mass values.
4) **ManyQuadruped** [25]: Depending on the sampled total weight given a $\mathbf{c}_i$ we interpolate between the nominal PD gain values assigned to each reference robot model (similar to GenLoco).
5) **MorAL** [26]: We apply the polynomial function mapping from total robot weight to scaled joint PD gains.
6) **URMA** [27]: We apply the same nominal PD gains with tight uniform noise ranges SR $=$ small. The parameters are interpolated from the values given in the original work.

Only our approach and PAL yielded sufficiently stable locomotion behavior on hardware. In contrast, MorAL and URMA achieved stable standing but were unable to realize walking. ManyQuadrupeds and GenLoco both required PD gains outside safe operating ranges and were therefore excluded from hardware evaluation. For safety reasons, experiments were limited to PD gains known to avoid high stress or destructive actuator behavior, such as joint derivative gains above 1.5. These qualitative findings are illustrated in the supplementary video material.

B. Sim-to-Sim Robustness

We use the definition of success rate (SR*) as a performance metric for robustness as: SR$^* = 1 - \frac{N_e}{N_T}$. With N_e referring to the number of rollouts that terminated early due to a prohibited collision and N_T being the total number of rollouts. Using $N_T = 100$, we randomize the base linear velocity command for 4 s before re-spawning. The same early termination criteria defined in Section IV-D are applied. The ANYmal C results are shown for hardware-tuned PD gains of $(85, 0.5)$ and for simulation-tuned gains for A1 with $(35, 0.5)$ in Fig. 3. We test three parameter changes, measuring each time the SR*: horizontal force perturbation onto the base, changes in foot contact friction, and base mass. The SR* results over a range of PD gains are shown in Fig 4.

Overall, our proposed mass independent sampling strategy with full PD range yields the most robust *uniform* locomotion policy for ANYmal and for A1 is performing competitively with the original GenLoco. Note that compared to PAL, MorAL, or URMA, both ManyQuadrupeds and GenLoco were trained with significantly higher PD gains, shown as red markers in Fig. 4, which did not enable deployment on the ANYmal robot due to requiring unsafe PD gain values. We associate this with the latter two controllers to be tuned for imitation-based feet trajectories using motion capture data

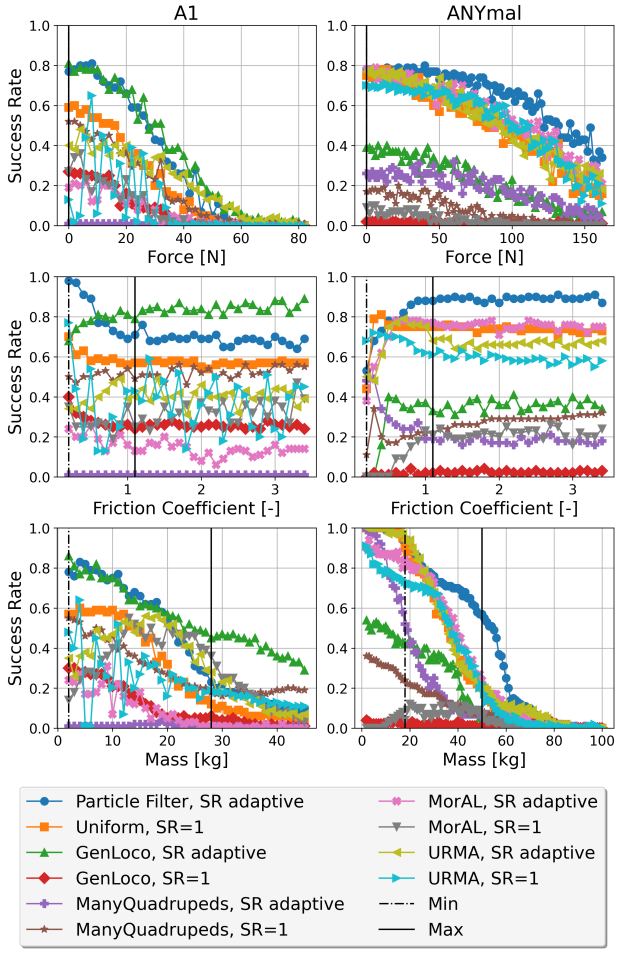

Fig. 3: Success rates for various parameter sampling strategies for different perturbations and dynamics parameters for the A1 (left column) and ANYmal (right column) quadrupeds. During random walking, we measure SR* over base perturbation in the horizontal plane, ground friction coefficient, and base mass changes, showing in black the minimum and maximum training range of the disturbance parameters.

or central pattern generators. We also observe that adaptive configuration sampling results mostly in a better locomotion policy, with outliers related to high values of PD Gains in ManyQuadrupeds and GenLoco. To further understand the influence of PD gain sampling approaches, we plot SR* over the PD ranges for A1 and ANYmal in Fig. 4.

The results show a better capability for larger quadrupeds with higher SR* coverage over wider ranges of parameters for the particle filter adaptive weighting. URMA's [27] approach of using nominal values and then uniformly adding noise around the PD gains appears to be a good way to inject prior knowledge into the training, especially when the ranges are increased as in this work. In general, we conclude that performance adaptive sampling leads to more robust locomotion policies for zero-shot deployment in simulation.

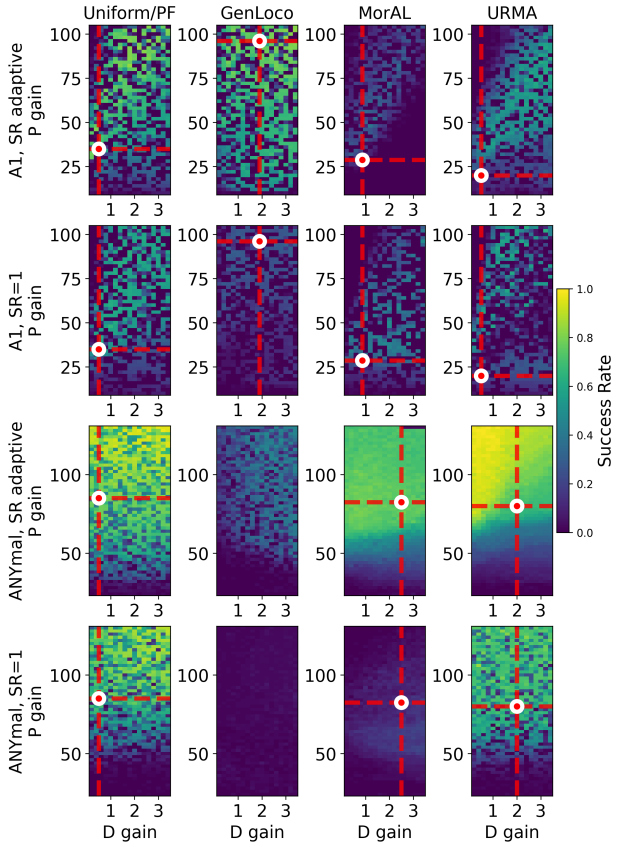

Fig. 4: The success rate SR* is measured over a range of PD gain combinations, with yellow indicating good locomotion capabilities and dark blue indicating low to no stable performance. The red-white dot indicates the nominal PD gains as per the original implementation. The tests were run in RaiSim for the A1 (top two rows) and ANYmal (bottom two rows). The different $\mathbf{c}_i$ sampling strategies go from left to right as Uniform (SR = 1) or particle filter (with SR adaptive), GenLoco (SR = 1 or adaptive), MorAL (SR = 1 or adaptive), and URMA (SR = 1 or adaptive). We removed ManyQuadrupeds as it performed worse than GenLoco. In red, we show the nominal gains for the respective sampling type and robot.

C. Sim-to-Real Transfer

In our hardware experiments, we transferred the trained dynamics encoding and locomotion policy networks to the ANYmal quadruped.

1) Hardware Velocity Command Tracking: We evaluate the effectiveness of the learned controllers on the robot in the real world. We successfully deploy controller policies with different sampling strategies. The results in Table III show the Root Mean Square Error (RMSE) for command tracking by comparing the target velocity command with the estimated base linear heading, lateral, and turn velocity.

2) Hardware Base Velocity Estimator: The velocity estimate is another measure of the quality of the controllers and robot IDs chosen. We compute the RMSE of the trained MLP estimator network with the onboard manufacturer ANYmal state estimator in Table IV.

Sampling	SR	PD	X_{RMSE}	Y_{RMSE}	θ_{RMSE}
Particle Filt	adaptive	85.0:0.5	0.1101	0.0663	0.2267
Particle Filt	adaptive	82.5:0.5	0.1009	0.0625	0.2194
Particle Filt	adaptive	80.0:0.5	0.0983	**0.0575**	0.2248
Particle Filt	adaptive	85.0:1.0	0.1100	0.0764	0.2530
Particle Filt	adaptive	82.5:1.0	**0.0957**	0.0814	0.2797
Particle Filt	adaptive	80.0:1.0	0.1065	0.0765	0.3067
Uniform	=1	85.0:0.5	0.0983	0.0922	0.2294
Uniform	=1	82.5:0.5	0.1094	0.0903	0.2309
Uniform	=1	80.0:0.5	0.1058	0.0833	0.2723
Uniform	=1	85.0:1.0	0.1082	0.0878	0.2955
Uniform	=1	82.5:1.0	0.1374	0.0930	0.3264
Uniform	=1	80.0:1.0	0.1139	0.0677	**0.2184**
URMA	=1	80 : 2*	NA	NA	NA
URMA	adaptive	85.0:1.0	0.1179	0.0881	0.1540
MorAL	=1	82 : 5*	NA	NA	NA
MorAL	adaptive	85.0:1.0	0.1864	0.1227	0.3384
ManyQuad	=1 / adap	430 : 20.7*	NA	NA	NA
GenLoco	=1 / adap	400 : 8.*	NA	NA	NA

TABLE III: Walking test: velocity command tracking error comparison with RMSE of heading (-X), lateral (-Y), and turn (-θ) commands on the hardware ANYmal.

Sampling	SR	PD	X_{RMSE}	Y_{RMSE}	Z_{RMSE}
Particle Filt	adaptive	85.0:0.5	0.1345	0.1242	0.0857
Particle Filt	adaptive	82.5:0.5	0.1566	0.1124	0.0882
Particle Filt	adaptive	80.0:0.5	0.1442	0.1148	0.0834
Particle Filt	adaptive	85.0:1.0	0.1771	0.1053	0.0824
Particle Filt	adaptive	82.5:1.0	0.1541	0.0985	0.0821
Particle Filt	adaptive	80.0:1.0	0.1855	0.1174	0.0824
Uniform	=1	85.0:0.5	**0.1007**	0.1039	0.0537
Uniform	=1	82.5:0.5	0.1202	0.1378	0.0739
Uniform	=1	80.0:0.5	0.1393	0.1240	0.0680
Uniform	=1	85.0:1.0	0.1384	0.1148	0.0601
Uniform	=1	82.5:1.0	0.1657	0.1240	0.1176
Uniform	=1	80.0:1.0	0.1065	**0.0829**	**0.0499**
URMA	=1	80 : 2*	NA	NA	NA
URMA	adaptive	85.0:1.0	0.0863	0.0701	0.0762
MorAL	=1	82 : 5*	NA	NA	NA
MorAL	adaptive	85.0:1.0	0.0942	0.0797	0.1080
ManyQuad	=1 / adap	430 : 20.7*	NA	NA	NA
GenLoco	=1 / adap	400 : 8.*	NA	NA	NA

TABLE IV: Walking test: base velocity estimate RMSE of heading (-X), lateral (-Y), and height (-z) estimate with onboard state estimator base twist for hardware ANYmal.

Given the similar ranges of velocity command tracking and base linear velocity estimation, we conclude that either sampling approaches, Uniform and Particle Filter, are robust enough for zero-shot hardware deployment. The uniform sampling leads to better velocity estimation, which we attribute to the early exposure of the estimator network to the full range of possible $\mathbf{c}_i$ parameters from the start of the training phase. The velocity tracking is only slightly better using the particle filter approach. These results show that larger PD gain sampling ranges during training enable zero-shot *universal* locomotion.

VI. Conclusion

We presented the influence of configuration parameter sampling in the context of *universal* quadrupedal locomotion. Our results demonstrate the need for careful sampling of the PD gains for the joint controllers to bridge the sim-to-real gap for larger quadrupeds such as the ANYmal.

We demonstrate that the resulting policy achieves robust performances across a range of PD gains in simulation and on hardware. In simulation, our proposed approach yields improved locomotion capability and robustness by zero-shotting the controller policy on both the A1 and ANYmal platforms. Baseline approaches that inject a standard range-based PD sampling achieve less stable controllers and fail to transfer reliably onto hardware. Future work aims to improve upon perceptive *universal* locomotion and to deploy on a broader set of quadrupeds from different manufacturers.

REFERENCES

[1] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, S. Vijayakumar, and N. Mansard, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," in *2020 International Conference on Robotics and Automation*. IEEE. [Online]. Available: http://arxiv.org/abs/1909.04947

[2] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," in *2023 Transactions on Robotics*. IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/10138309/

[3] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *2023 Conference on Robot Learning*. [Online]. Available: http://arxiv.org/abs/2309.05665

[4] H. Kim, H. Oh, J. Park, Y. Kim, D. Youm, M. Jung, M. Lee, and J. Hwangbo, "High-speed control and navigation for quadrupedal robots on complex and discrete terrain," in *2025 Robotics and Automation Letters*. IEEE.

[5] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "DTC: Deep tracking control," in *2024 Science Robotics*, vol. 9. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.adh5401

[6] U. Robotics. A1. [Online]. Available: https://www.unitree.com/en/a1/

[7] ANYbotics. ANYmal d. [Online]. Available: https://www.anybotics.com/robotics/anymal/

[8] Learning agile and dynamic motor skills for legged robots | science robotics. [Online]. Available: https://www.science.org/doi/full/10.1126/scirobotics.aau5872

[9] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," in *2022 Science Robotics*, vol. eabk2822. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.abk2822

[10] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo, "Learning quadrupedal locomotion on deformable terrain," in *2023 Science Robotics*, vol. 8. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.ade2256

[11] J. Lee, M. Bjelonic, A. Reske, L. Wellhausen, T. Miki, and M. Hutter, "Learning robust autonomous navigation and locomotion for wheeled-legged robots," in *2024 Science Robotics*. [Online]. Available: http://arxiv.org/abs/2405.01792

[12] Y. Ma, A. Cramariuc, F. Farshidian, and M. Hutter, "Learning coordinated badminton skills for legged manipulators," in *2025 Science Robotics*, vol. 10.

[13] X. B. Peng, E. Coumans, T. Zhang, T. W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *arXiv preprint arXiv:2004.00784*.

[14] C. Li, M. Vlastelica, S. Blaes, J. Frey, F. Grimminger, and G. Martius, "Learning agile skills via adversarial imitation of rough partial demonstrations," in *2023 Conference on Robot Learning*.

[15] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," in *2022 International Conference on Intelligent Robots and Systems*. IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/9981973/

[16] Y. Fuchioka, Z. Xie, and M. v. d. Panne, "OPT-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors," in *2023 International Conference on Robotics and Automation*. IEEE. [Online]. Available: http://arxiv.org/abs/2210.01247

[17] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 International Conference on Robotics and Automation*. IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/8460528/

[18] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Van De Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *2020 Conference on Robotic Learning*.

[19] W. Huang, I. Mordatch, and D. Pathak, "One policy to control them all:shared modular policies for agent-agnostic control," in *2020 International Conference on Machine Learning*.

[20] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei, "MetaMorph: Learning universal controllers with transformers," in *2022 International Conference on Learning Representations*. [Online]. Available: https://openreview.net/forum?id=Opmqtk_GvYL

[21] B. Trabucco, M. Phielipp, and G. Berseth, "AnyMorph: Learning transferable polices by inferring agent morphology," in *2022 International conference on machine learning*. PMLR. [Online]. Available: http://arxiv.org/abs/2206.12279

[22] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *2017 International conference on machine learning*. PMLR.

[23] A. Belmonte-Baeza, J. Lee, G. Valsecchi, and M. Hutter, "Meta reinforcement learning for optimal design of legged robots," in *2022 Robotics and Automation Letters*, vol. 7. IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/9910025/

[24] G. Feng, H. Zhang, Z. Li, X. B. Peng, B. Basireddy, L. Yue, Z. Song, L. Yang, Y. Liu, K. Sreenath, and S. Levine, "GenLoco: Generalized locomotion controllers for quadrupedal robots," in *2022 Conference on robot learning*. PMLR. [Online]. Available: http://arxiv.org/abs/2209.05309

[25] M. Shafiee, G. Bellegarda, and A. Ijspeert, "ManyQuadrupeds: Learning a single locomotion policy for diverse quadruped robots," in *arXiv preprint arXiv:2310.10486*. arXiv. [Online]. Available: http://arxiv.org/abs/2310.10486

[26] Z. Luo, Y. Dong, X. Li, R. Huang, Z. Shu, E. Xiao, and P. Lu, "MorAL: Learning morphologically adaptive locomotion controller for quadrupedal robots on challenging terrains," in *2024 Robotics and Automation Letters*. IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/10463132/

[27] 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," in *2024 Conference on Robot Learning*. [Online]. Available: http://arxiv.org/abs/2409.06366

[28] D. Rytz, S. Choi, W. Yu, W. Merkt, J. Hwangbo, and I. Havoutis, "Reference free platform adaptive locomotion for quadrupedal robots using a dynamics conditioned policy," in *arXiv preprint arXiv:2505.16042*. [Online]. Available: http://arxiv.org/abs/2505.16042

[29] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," in *2020 Science Robotics*, vol. 5. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.abc5986

[30] R. Tech, "RaiSim – v1.1.7 documentation." [Online]. Available: https://raisim.com/

[31] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "ANYmal - a highly mobile and dynamic quadrupedal robot," in *2016 International Conference on Intelligent Robots and Systems*. IEEE, ISSN: 2153-0866.

[32] E. Ackerman. ANYbotics introduces sleek new ANYmal c quadruped. [Online]. Available: https://spectrum.ieee.org/anybotics-introduces-sleek-new-anymal-c-quadruped

[33] R. Tech. Articulated systems — RaiSim v1.1.7 documentation. [Online]. Available: https://raisim.com/sections/ArticulatedSystem.html

[34] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," in *2018 Robotics: Science and Systems*. MIT Press Journals. [Online]. Available: http://arxiv.org/abs/1804.10332

[35] Z. Xie, X. Da, M. van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited: A case study for quadrupedal locomotion," in *2021 IEEE International Conference on Robotics and Automation*. [Online]. Available: https://ieeexplore.ieee.org/document/9560837/

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," in *arXiv preprint*. [Online]. Available: https://arxiv.org/abs/1707.06347