GenLoco: Generalized Locomotion Controllers for Quadrupedal Robots

Anonymous Author(s)
Affiliation
Address
email

Abstract: Recent years have seen a surge in commercially-available and affordable quadrupedal robots, with many of these platforms being actively used in research and industry. As the availability of legged robots grows, so does the need for controllers that enable these robots to perform useful skills. However, most learning-based frameworks for controller development focus on training robot-specific controllers, a process that needs to be repeated for every new robot. In this work, we introduce a framework for training generalized locomotion controllers (GenLoco) for quadrupedal robots. Our framework synthesizes general-purpose locomotion controllers that can be deployed on a large variety of quadrupedal robots with similar morphologies. We present a simple but effective morphology randomization method that procedurally generates a diverse set of simulated robots for training. We show that by training a controller on this large set of simulated robots, our models acquire more general control strategies that can be directly transferred to novel simulated and real-world robots with diverse morphologies, which were not observed during training.

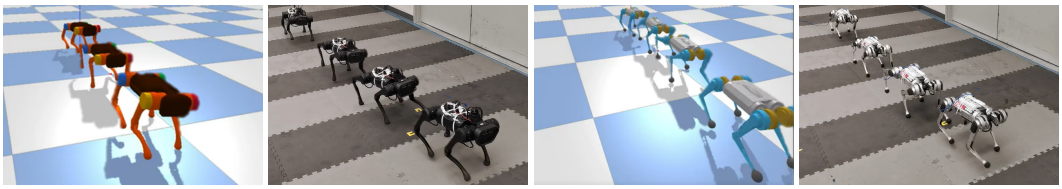
Keywords: Legged Locomotion, Reinforcement Learning, Transfer Learning

1 Introduction

Quadrupedal locomotion offers the potential for robotic agents to traverse and operate in complex unstructured environments. However, designing effective locomotion controllers for quadrupedal robots is challenging, as it typically requires detailed knowledge of the dynamics of a particular system and careful controller design for each desired skill. Model-free reinforcement learning (RL) provides a paradigm that can automate much of the controller engineering process, where an agent learns locomotion skills automatically through trial-and-error. RL techniques have been effective for developing locomotion controllers for a large variety of quadrupedal robots [1, 2, 3, 4, 5, 6, 7]. While RL provides a general framework that can in principle be applied to any robot, the resulting controllers are most often specific to the particular robot that they were trained on. Therefore, these controllers will generally be ineffective when deployed on another robot, and new controllers will need to be trained from scratch specifically for the new system. This requires either extensive data collection or, if using simulation to real-world transfer, detailed simulated models.

Recent years have seen the emergence of a growing catalog of commercially available quadrupedal robots, with many of these systems converging on similar body plans. This similarity in the morphologies of quadrupedal robots may lead one to wonder: is it possible to create a generalized locomotion controller that can be broadly applied to different quadrupedal robots? If so, such generalized controllers can greatly reduce the labor-intensive process of continually developing robot-specific controllers for new systems. In this work, we present a framework for developing more general locomotion controllers, which can be applied to a range of different quadrupedal robots. We focus our investigation on the setting where the test-time morphology is unknown, and thus the goal is to develop controllers that generalize to new robots not observed during training. To this end, we propose a morphology randomization method that reduces the need for robot specific information during training by procedurally generating a diverse set of morphologies. By randomizing the morphology

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.



(a) Simulated A1 (b) Real A1 (c) Simulated Mini Cheetah (d) Real Mini Cheetah

Figure 1: Testing of different simulated and real quadrupedal robots performing pacing gaits using a single locomotion controller. Our controllers can be directly deployed from simulation to the real world and across robots with different morphology and dynamics, without explicitly training on the specific robots used during testing.

and dynamics of the simulated robots, our system is able to train generalized controllers that can be deployed across a variety of different robots, as shown in Fig. 1.

The core contribution of this work is the development of a reinforcement learning framework that produces generalized locomotion controllers (GenLoco) for quadrupedal locomotion skills, which can be deployed on a large variety of different robot morphologies. By training controllers on a wide range of randomly generated robot morphologies, our system is able to learn policies that generalize to new robots not seen during training. We demonstrate the effectiveness of our model on many notable quadrupedal robots in simulation and in real-world experiments. We show that the learned controllers can be deployed directly on a number of commonly used quadrupedal robots, including the Unitree's A1 and MIT's Mini Cheetah [8]. We also demonstrate that, by introducing morphology randomization into the training process, our framework is able to more effectively transfer policies from simulation to the real-world than policies trained on a specific robot. In this paper, we focus on the randomized robot morphology while keeping number of Degree of Freedoms (DoFs) and links constant. Code and obtained policies in our work will be released upon publication.

2 Related Work

RL provides a general framework for learning robotic controllers for a large array of tasks. Instead of requiring tedious manual controller engineering, RL techniques can automatically synthesize controllers for a desired task by optimizing the controller against an objective function [9]. RL has been applied to develop a wide range of motor skills for agents in simulation [10, 11, 12, 13], and in the real world [14, 15, 16, 17, 18, 19]. But due to challenges associated with applying RL algorithms on real-world systems [20], sim-to-real techniques are commonly used to adapt and transfer controllers trained in simulation to a physical system [21, 22, 4, 3, 19]. Domain randomization is one of the most commonly used sim-to-real transfer techniques, where the dynamics of a simulator are randomized during training in order to develop controllers that can adapt to differences between simulated and real-world dynamics [23, 24, 21, 4, 25, 26]. Real-world data can also be leveraged to further improve real-world performance by adapting the dynamics of the simulator [27, 28, 19, 29, 30], or directly finetuning a controller's behaviors [31, 3, 5]. In this work, we also take a domain randomization approach to transfer controllers trained in simulation to robots operating in the real world. However, unlike many prior sim-to-real methods, which assume a known morphology for the real-world robot, our work explores the development of more general locomotion controllers deployable on a variety of different robots, including morphologies not known during training.

Developing general-purpose controllers that can be deployed on robots with different embodiments can greatly mitigate the overhead of creating individual robot-specific controllers. Devin et al. [32] utilized a modular network architecture to transfer manipulation skills across a small set of manually-crafted simulated robotic arms. Chen et al. [33] proposed conditioning policies on learned encodings of a robot's morphology, which allowed a controller to be deployed on a number of different robotic arms. A similar morphology encoding approach has also been applied to train full-body motion controllers, which can be deployed on a large variety of simulated humanoid agents [34]. Graph convolutional networks have been used to implicitly encode the morphological structure of simulated robots with varying numbers of degrees-of-freedom [35, 36]. Gupta et al. [37] trained domain invariant features spaces in order to transfer manipulation skills across simulated robots with different morphologies. While these transfer learning techniques have shown promising results

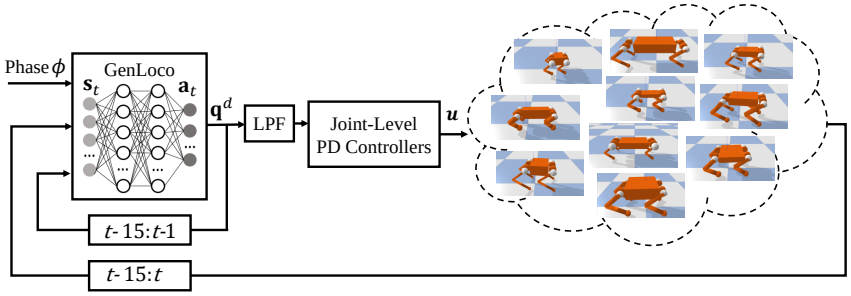


Figure 2: The proposed generalized locomotion control framework for quadrupedal robots. GenLoco is designed to work on a large collection of robots with different morphology and dynamics. The input observations of the policy consist of a phase variable ϕ representing progression along a motion, a history of the robot's raw sensor feedback, and a history of past actions. The controller then outputs desired motor positions, which are first passed through a low-pass filter, before being applied to joint-level PD controllers to generate motor torques.

in simulation, they have yet to be demonstrated on robots in the real world. In this work, we aim to develop general locomotion controllers that can be deployed on a large variety of robots with different morphologies. Although prior works suggest that recurrent or memory-based policies can improve transfer by performing "implicit system identification," we found that our approach was able to produce effective policies with fully feedforward neural networks. By taking in past observations and actions as input, the model is able to infer the morphological structure of a robot, and successfully transfer locomotion skills across a diverse set of quadrupedal morphologies without requiring an explicit representation or encoding of a particular robot's body structure.

3 Generalized Quadrupedal Locomotion Controllers

In this section, we present a framework for training Generalized Locomotion Controller (GenLoco), where a single controller can be deployed on a large variety of quadrupedal robots with different morphologies. An overview of our system is shown in Fig. 2. The controllers are trained through model-free reinforcement learning. But unlike most prior RL frameworks for robotic locomotion, which assume a fixed morphology for the robot during both training and testing, our work explores the setting where the embodiment of the robot can vary, and the particular embodiment at test time is not known a priori. Our Models are trained by randomizing the morphology and other dynamics properties during training in simulation, thereby encouraging the controller to learn adaptable strategies, which can be effectively applied on different robots. At every timestep, the controller receives a history of sensory observations as input, which can be used to infer the morphology of the robot it is currently deployed on. The controller then outputs actions, which specify target motor positions for each joint. The motor targets are first passed through a low-pass filter [3], to mitigate undesirable high-frequency movements, before being applied to joint-level PD controllers to generate motor torques.

3.1 Morphology Generation

The key to developing generalized locomotion controllers lies in training the controller on a diverse collection of robots, which encourages the controller to learn strategies that can generalize to new robots not observed during training. At the start of each training episode, we procedurally generate a random robot morphology based on a predefined morphology template. As illustrated in Fig. 3, many commonly used quadrupedal robots follow a similar body structure, consisting of a robot base and four legs, each of which has 3 DoFs: hip, thigh, and knee joints. The robot's feet can be modeled as spheres, and the contacts between the feet and the ground can be approximated as point contacts. This design template has been followed in widely used quadrupedal robots, including Unitree's A1, Go1, Laikago and Aliengo, Boston Dynamics' Spot, MIT's Mini Cheetah, and CUHK's Sirius. Therefore, our system generates random morphological variations based on this template by randomizing the proportions of the various body parts and their respective dynamics properties. A sample of robots produced by our morphology generation procedure are shown in Fig. 2. The randomized morphology parameters and their respective ranges are recorded in Table 1. Below, we provide a more detailed account of the major parameters of variations in our morphology generation procedure.

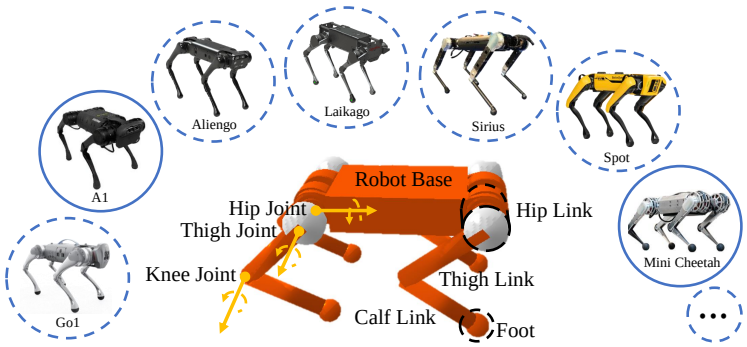


Figure 3: Many common quadrupedal robots follow a common morphological template, consisting of a robot base (6 DoFs) and four 3-DoF legs. This design is followed in robots such as Unitree's A1, Go1, Aliengo, Laikago, MIT's Mini Cheetah, CHUK's sirius, and Boston Dynamics' Spot. The robots highlighted with solid circles are the ones used to validate our system in the real world.

Table 1: Morphology generation parameters.

Parameter	Min and Max Values	Units
Size factor α	$[0.8, 1.2]$	1
Base length b_l	$\alpha \times [0.007, 0.527]$	m
Base width b_w	$\alpha \times [0.005, 0.383]$	m
Base height b_h	$\alpha \times [0.003, 0.225]$	m
Base density b_d	$[400, 1200]$	kg/m^3
Calf length l_c	$\alpha \times [0.11, 0.33]$	m
Calf radius r_c	$\alpha \times [0.01, 0.03]$	m
Thigh length l_t	$[0.75, 1.25] \times$ Calf length l_c	m
Thigh radius r_t	$[0.75, 1.25] \times$ Calf radius r_c	m
Foot radius r_f	$1.5\times$ Calf radius r_c	m
Link masses	$[0.5, 1.5] \times$ nominal value	kg
Joint PD gains	nominal value $\times \frac{\text{robot mass}}{\text{nominal mass}} \times [0.7, 1.3]$	Nm/rad, Nms/rad

Size factor. We introduce a size factor α, which uniformly scales the size of each body part (e.g. robot base and legs). Because α multiplicatively scales along every dimension, it is a sensitive parameter, and we found randomly sampling values of α from the range $[0.8, 1.2]$ to be sufficient in capturing the variation among most robots. This factor facilitates positive correlation between robot component sizes, reducing the likelihood of generating morphologies with distorted proportions.

Robot base parameters. The parameters of the robot base include its size and density. We model the base geometrically as a rectangular box. The dimensions, such as length b_l, width b_w, and height b_h, of the base are randomized to cover a wide range of robot base sizes. The density of the base is also randomized between $[400, 1200]\text{kg/m}^3$. The ranges of values, detailed in Table 1, are selected to encapsulate the sizes and masses of popular industrial robots.

Leg parameters. The robot's upper leg (thigh) and the lower leg (calf) are modeled as cylindrical solids, and the robot's feet are represented by spheres, as shown in Fig. 3. Specifically, the sizes of the robot's thigh and foot are chosen based on the robot's calf link dimensions, as detailed in Table 1. The range of values for the thigh length l_t is specified to be $\pm 25\%$ of the calf length l_c. Similarly, the range of values for the thigh radius is defined as $\pm 25\%$ of the calf radius r_c. By correlating the properties of different segments of the legs, we can prevent the morphology generator from producing implausible designs, such as a robot with large calves but very small thighs and feet, which would hinder the robot's locomotion capabilities.

PD gains. Larger and heavier robots typically require stiffer joint-level PD gains in order to generate larger motor torques. Therefore, the gains used in the PD controllers are also scaled with respect to the mass of the robot. The *nominal mass* is set to the mass of the A1 (12.458 kg). This randomization scheme helps to ensure that the PD controllers are sufficiently strong for larger and heavier morphologies.

4 Training

Our controllers are trained to perform various locomotion skills using a reinforcement learning-based motion imitation framework [3]. The goal of these controllers is to imitate a given reference motion $\mathbf{q}^r = \{\mathbf{q}_0^r, \mathbf{q}_1^r, \ldots, \mathbf{q}_T^r\}$, which specifies target poses $\mathbf{q}_t^r$ at each timestep t, The reward r_t at timestep t is computed according to:

$$r_t = w^{\text{p}} r_t^{\text{p}} + w^{\text{v}} r_t^{\text{v}} + w^{\text{bp}} r_t^{\text{bp}} + w^{\text{bv}} r_t^{\text{bv}},$$

$$\text{with } w^{\text{p}} = 0.6, \quad w^{\text{v}} = 0.1, \quad w^{\text{bp}} = 0.15, \quad w^{\text{bv}} = 0.15.$$

The pose reward $r_t^{\text{p}} = \exp\left[-5 \sum_{j=1}^{12} \|\hat{q}_t^j - q_t^j\|^2\right]$ encourages the robot to match its local joint rotations $\mathbf{q}_t$ with those specified by the reference motion, where q_t^j denotes the rotation of the j-th joint. The velocity reward r_t^{v} follows a similar form, and encourages the robot to match the joint velocities of the reference motion. The base position reward r_t^{bp} and base velocity reward r_t^{bv} encourage the robot to track the motion of the base from the reference motion. A more detailed description of the reward function is available in Appendix B.

State and action spaces. As shown in Fig. 2, actions from the controller $\mathbf{a}_t$ specify target joint positions $\mathbf{q}^d \in \mathbb{R}^{12}$, which are used by the joint-level PD controllers to determine desired motor torques. The controller operates at 30 Hz. To encourage smoother motions, actions are processed with a low-pass filter before being applied to the robot. The observation $\mathbf{s}_t$ of the controller includes three components: 1) a 15-timestep history of robot raw sensor feedback $\mathbf{q}_{t-15:t}^{\text{base}}$ and $\mathbf{q}_{t-15:t-1}$, 2) a 15-timestep history of past actions $\mathbf{a}_{t-15:t-1}$, and 3) a phase variable $\phi \in [0, 1]$ that indicates the robot's progress along the reference motion, where 0 denotes the start of the motion and 1 denotes the end [10]. The robot's sensor feedback at each timestep consists of the base orientation $\mathbf{q}^{\text{base}} \in \mathbb{R}^4$ from IMU sensors, recorded as a quaternion, and the measured joint positions $\mathbf{q} \in \mathbb{R}^{12}$. Note that the observations do not include quantities that require an explicitly state estimator, such as the linear velocity of the base [2, 3, 26, 7] or contact sensors [4]. The history of sensor readings and actions provides the GenLoco policy some information that can be used to perform state estimation and to infer the robot's dynamics and morphology.

Dynamics randomization. In addition to randomizing the morphological structure of the robots, we also randomize the dynamics parameters of the simulation during training [38] to improve the robustness of our policy and facilitate sim-to-real transfer. The randomized dynamics parameters include three categories: 1) link mass, link inertia, and ground friction to deal with modeling errors, 2) motor strength, joint-level PD gains and motor friction to mitigate the uncertainties of motor dynamics, and 3) the latency between the policy and the joint-level controllers. The randomization ranges of each parameter are detailed in Table 3 in Appendix C.

Episode design and curriculum learning. Each episode has a horizon of 100 timesteps, lasting about 3 seconds. Early termination is applied when the robot deviates too far from the reference base position and orientation [3]. At the start of each episode, a new morphology is generated according to the procedure described in Sec. 3.1, and the dynamics parameters are also randomly selected according to Sec. 4. The introduction of the morphology randomization significantly increases the difficulty of the task, and the agent can fail to learn desired skills. Therefore, we utilize a training curriculum to assist in training with our morphology generation procedure. During training, we linearly increase the width of the random interval detailed in Table 1 with respect to the training iteration. Specifically, in the first training iteration, we use a fixed morphology (i.e., sample parameters from a 0-width interval centered on the average of the min and max values in Table 1). In each subsequent training iteration, the widths of the intervals from which we sampled parameter values are linearly increased until the interval bounds reach the minimum and maximum values.

GenLoco policies π_θ are trained using Proximal Policy Optimization (PPO) [39], and all simulations are performed using Pybullet simulation [40]. Each policy is trained with about 800 million samples, requiring approximately 2 weeks on 16 CPU workers. The morphology randomization curriculum linearly anneals the range of parameter values from 0 to their respective maximum range over the course of 200 million samples. Learning curves are presented in Appendix D.

Figure 4: GenLoco policies deployed on a collection of quadrupedal robots that exist in real life. Two separate models are trained to perform a pacing gait and a spinning gait respectively. The GenLoco policies are trained in simulation using only procedurally generated robots, and robots used for testing are not used in the training process. The learned controllers can be directly deployed on all of these robots, including the ANYmal-B robot which has a distinct knee joint design, to perform agile maneuvers without additional training.

5 Simulation Validation

In this section, we validate the effectiveness of our models on controlling quadrupedal robots with different morphologies by direct transfer in simulation. We compare our GenLoco policies with the policies trained on specific robots, which use the same training settings introduced in Sec. 4. We train controllers for imitating two reference motions: a forward moving pacing gait and an in-place spinning motion. The reference motions are retargeted to match our robot's morphology from motion captured data of a real dog [3].

Our controllers are trained using procedurally generated robots, and then tested on simulated models of commercially available robots. The robots used for testing include the A1, Aliengo, Go1, Mini Cheetah, Sirius, Laikago, and ANYmal-B. The A1 and Go1 are small robots with a mass of about 12 kg and a height of about 0.4m. The Sirius, Aliengo and Laikago are heavier and larger with each standing at 0.6m in height. Furthermore, ANYmal-B has a different knee joint design from the design template used for training. Detailed specifications of these robots are available in Appendix E.

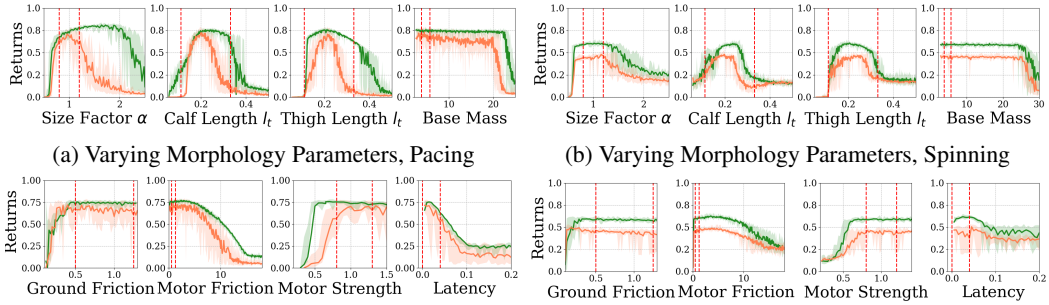
5.1 Zero-Shot Transfer to Novel Robots

As shown in Fig. 4 and the supplementary video, our controllers can be directly deployed on a large variety of quadrupedal robots. None of the robots we tested on were used during training. Notably, GenLoco policies successfully transfer to the ANYmal-B robot, which possesses an inverted knee joint design different from the template used during training (Fig. 3), without additional tuning. This highlights the generalizability of our proposed GenLoco framework.

Our GenLoco policies are some of the first locomotion controllers that can be deployed directly on different quadrupedal robots without further fine-tuning. By providing a history of past observations and actions to the controller, we enable it to leverage the history to infer the morphology of the robot it is currently deployed on, and by randomizing the morphology during training the model develops more general control strategies that can effectively execute a desired skill on different morphologies.

5.2 Out-of-Distribution Generalization

To further evaluate the ability of GenLoco to generalize learned locomotion skills to different morphologies, we test the controllers extensively on a large range of simulated robot morphologies. We evaluate the controller's performance when varying four morphology parameters: 1) body size,



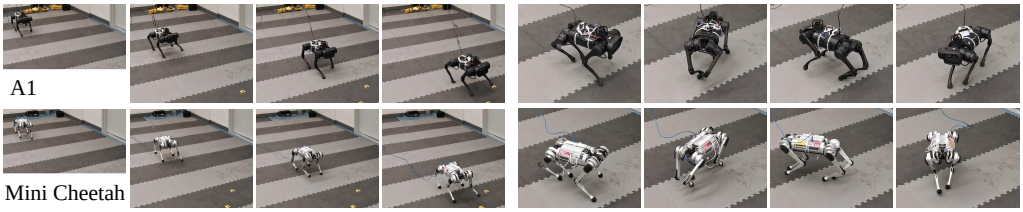
(a) Varying Morphology Parameters, Pacing

(b) Varying Morphology Parameters, Spinning

(c) Varying Dynamics Parameters, Pacing

(d) Varying Dynamics Parameters, Spinning

Figure 5: Benchmark of performance of GenLoco policies and policies trained specifically for A1 robot to perform pacing and spinning skills on a range of different robot morphologies and dynamics parameters in simulation. Green lines are the normalized return of GenLoco (ours) while orange ones are those of the A1-specific policy. The red dashed lines indicate the training range. Note that for the varying morphology test the A1-specific policy is not trained with randomized morphology, and the dynamics randomization range is the same for all of these policies. Overall, the GenLoco policies outperform the A1-specific policies over different morphology and dynamics parameters. Furthermore, GenLoco is able to generalize over a larger range of morphologies and dynamics. Each testing episode lasts 100 timesteps and returns are averaged across 10 trials.



(a) Pacing gait using a single GenLoco

(b) Spinning gait using a single GenLoco

Figure 6: A single GenLoco policy can be deployed directly on different robots, which it was not trained on, in the real world, enabling them to perform agile maneuvers such as pacing and spinning.

which is scaled by size factor α, 2) calf length l_c, 3) thigh length l_t, and 4) body mass b_m. The testing range is set to be much larger than the range used during training detailed in Table 1. We compare GenLoco to policies trained specifically for the A1, since the nominal values used in the morphology generation process (Table 1) are based on the original A1 robot. The comparisons with the pacing gait controllers and spinning gait controllers are shown in Fig. 5a and Fig. 5b, respectively. The GenLoco policies are overall able to better generalize to out-of-distribution robots, exhibiting more robust behaviors and maintaining higher returns across larger variations in the morphological parameters than the A1-specific policies. In the case of the pacing skill, GenLoco's performance does not degrade until the robot size is more than two times larger than the maximum training size. While the A1-specific policy shows some robustness to small variations in the morphology parameters, the policy fails when deployed on other robots, such as the Sirius.

In order to understand the advantages of the morphology randomization during training, we test the robustness of the GenLoco policies and A1-specific policies by varying the dynamics parameters. In addition to generalization to different morphologies, our model is also robust to large variations in the dynamics of a system. Fig. 5c compares the performance of GenLoco policies to A1-specific policies that were also trained using the same range of dynamics randomization detailed in Table 3. As demonstrated in Fig. 5c, 5d, for the same range of dynamics randomization during training, GenLoco policies consistently outperform the A1-specific policies. In particular, GenLoco policies demonstrate notable robustness to changes in the dynamics parameters related to actuated joints, such as motor friction and strength. This improved performance on out-of-distribution settings is likely in part due to random morphologies introducing a more diverse range of dynamics for the policy to train on.

6 Real-World Deployment

Finally, we evaluate the effectiveness of the learned controllers on robots in the real world. We test the controllers on two robots, Unitree's A1 and MIT's Mini Cheetah. As shown in Fig. 6 and the

Table 2: Normalized return of policies deployed on real robots

Pacing	**GenLoco on A1**	**GenLoco on Cheetah**	**A1 Specific**	**Cheetah Specific**
Return	0.702	0.546	0.657	0 (Failed)
Spinning	**GenLoco on A1**	**GenLoco on Cheetah**	**A1 Specific**	**Cheetah Specific**
Return	0.752	0.528	0.62	0.262

supplementary video, our GenLoco policies can be deployed on the real robots in a zero-shot manner, enabling the robots to perform agile maneuvers such as pacing and spinning. We further compare the real-world performance of GenLoco to policies trained specifically for the A1 and Mini Cheetah. The performance of these policies are available in Table 2. We note that the performance of GenLoco policies, which work on both robots and were trained on neither, is overall better than the policies trained for specific robots, as shown in the supplementary video. Furthermore, there is a failure when we deploy the Mini Cheetah-specific policy for pacing on the real robot, even though the same policy works well in simulation as shown in Fig. 4. Moreover, when testing the A1's robot-specific spin policy on the hardware, the robot drifts in the transverse plane during its motion. Notably, these two performance degradations during sim-to-real transfer do not occur for the GenLoco policies under identical environment circumstances, as shown in the supplementary video. This highlights the advantages of GenLoco in terms of robustness and generalization during sim-to-real transfer.

7 Limitations

As shown in Fig. 5a, 5b, the performance of GenLoco policies drops dramatically for larger robots, such as Boston Dynamics' Spot, which has a mass of 31 kg and is 0.7m tall. This may suggest that more aggressive randomization and a more expressive model architecture with recurrence are needed during training, like [41]. Moreover, our model does not generalize well to robots that have morphologies drastically different from the template used in our morphology generation procedure (Fig. 3), and it fails to transfer to the ANYmal-C robot [42]. Lastly, our models cannot be deployed on robots with a different number of DoFs. More flexible architectures, such as graph neural networks [35, 36], may allow for more adaptable models that can handle variable numbers of DoFs.

8 Conclusion

In this paper, we present an RL-based framework for training generalized locomotion controllers that can be deployed on a diverse set of quadrupedal robots. We show that a simple history-based model, trained on procedurally generated robots in simulation, can be successfully transferred to a large variety of new robots, which were not observed during training. The trained models can also be deployed directly on real robots, without requiring any additional training on the physical systems. While our experiments have been focused on quadrupedal robots, our method is general and can also be applied to robots in other domains, such as robotic arms and quadrotors. However, the effectiveness of our models remains limited to robots that have the same number of DoFs and follow a predefined morphological template. Despite these limitations, we hope our work will provide a stepping stone towards more general-purpose controllers that can be widely and conveniently deployed on a diverse catalog of robots.

References

[1] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 2019.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[3] 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 *Robotics: Science and Systems*, 07 2020. doi:10.15607/RSS.2020.XVI.064.

[4] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[5] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world, 2021.

[6] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022.

[7] G. Ji, J. Mun, H. Kim, and J. Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7 (2):4630–4637, 2022.

[8] B. Katz, J. Di Carlo, and S. Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *Int. Conf. on Robotics and Automation (ICRA)*, pages 6295–6301, 2019.

[9] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[10] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi:10.1145/3197517.3201311. URL http://doi.acm.org/10.1145/3197517.3201311.

[11] L. Liu and J. Hodgins. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Trans. Graph.*, 37(4), July 2018. ISSN 0730-0301. doi:10.1145/3197517.3201315. URL https://doi.org/10.1145/3197517.3201315.

[12] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[13] S. Lee, M. Park, K. Lee, and J. Lee. Scalable muscle-actuated human simulation and control. *ACM Transactions On Graphics (TOG)*, 38(4):1–13, 2019.

[14] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 3, pages 2619–2624. IEEE, 2004.

[15] R. Tedrake, T. W. Zhang, and H. S. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, volume 3, pages 2849–2854, Piscataway, NJ, USA, 2004. IEEE. ISBN 0-7803-8463-6. URL http://www.cs.cmu.edu/~cga/legs/01389841.pdf.

[16] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng. Learning cpg sensory feedback with policy gradient for biped locomotion for a full-body humanoid. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, page 1267–1273. AAAI Press, 2005. ISBN 157735236x.

[17] 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 *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi:10.15607/RSS.2018.XIV.010.

[18] T. Haarnoja, A. Zhou, S. Ha, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. *CoRR*, abs/1812.11103, 2018. URL http://arxiv.org/abs/1812.11103.

[19] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. doi:10.1126/scirobotics.aau5872. URL https://robotics.sciencemag.org/content/4/26/eaau5872.

[20] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine. The ingredients of real world robotic reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJe2syrtvS.

[21] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[22] 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 *Proc. Conference on Robot Learning (CORL 2019)*, 2019.

[23] W. Yu, J. Tan, C. K. Liu, and G. Turk. Preparing for the unknown: Learning a universal policy with online system identification. In N. M. Amato, S. S. Srinivasa, N. Ayanian, and S. Kuindersma, editors, *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*, 2017. doi:10.15607/RSS.2017.XIII.048. URL http://www.roboticsproceedings.org/rss13/p48.html.

[24] 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 *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. doi:10.1109/IROS.2017.8202133.

[25] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=wK2fDDJ5VcF.

[26] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2811–2817. IEEE, 2021.

[27] J. Tan, Z. Xie, B. Boots, and C. K. Liu. Simulation-based design of dynamic controllers for humanoid balancing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2729–2736, Oct 2016. doi:10.1109/IROS.2016.7759424.

[28] J. Hanna and P. Stone. Grounded action transformation for robot learning in simulation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, February 2017.

[29] K. Lowrey, S. Kolev, J. Dao, A. Rajeswaran, and E. Todorov. Reinforcement learning for non-prehensile manipulation: Transfer from simulation to physical system. *CoRR*, abs/1803.10371, 2018. URL http://arxiv.org/abs/1803.10371.

[30] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. D. Ratliff, and D. Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *CoRR*, abs/1810.05687, 2018. URL http://arxiv.org/abs/1810.05687.

[31] W. Yu, V. C. V. Kumar, G. Turk, and C. K. Liu. Sim-to-real transfer for biped locomotion. *CoRR*, abs/1903.01390, 2019. URL http://arxiv.org/abs/1903.01390.

[32] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2169–2176. IEEE, 2017.

[33] T. Chen, A. Murali, and A. Gupta. Hardware conditioned policies for multi-robot transfer learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[34] J. Won and J. Lee. Learning body shape variation in physics-based characters. *ACM Trans. Graph.*, 38(6), nov 2019. ISSN 0730-0301. doi:10.1145/3355089.3356499. URL https://doi.org/10.1145/3355089.3356499.

[35] T. Wang, R. Liao, J. Ba, and S. Fidler. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1sqHMZCb.

[36] W. Huang, I. Mordatch, and D. Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *ICML*, 2020.

[37] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

[38] 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, 2017. URL https://arxiv.org/abs/1703.06907.

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

[40] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[41] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[42] Anybiotics, "anymal c". https://www.anybotics.com/anymal-autonomous-legged-robot/. Accessed: 2022-06-15.

Appendix A Nominal Morphology Generation Parameter Values

Parameter	Nominal Value
Hip Link Mass	0.696 kg
Upper Leg Link Mass	1.013 kg
Lower Leg Link Mass	0.166 kg
Toe Link Mass	0.06 kg
Mass (for joint PD gains scaling)	12.458 kg

Appendix B Reward Function

Similar to Peng et al. [3], we compute our reward function as follows:

$$r_t = w^{\text{p}} r_t^{\text{p}} + w^{\text{v}} r_t^{\text{v}} + w^{\text{bp}} r_t^{\text{bp}} + w^{\text{bv}} r_t^{\text{bv}}$$

$$w^{\text{p}} = 0.6, \quad w^{\text{v}} = 0.1, \quad w^{\text{bp}} = 0.15, \quad w^{\text{bv}} = 0.15$$

where the pose reward r_t^{p}, velocity reward r_t^{v}, base position reward r_t^{bp}, base velocity reward r_t^{bv} terms are defined according to:

$$r_t^{\text{p}} = \exp\left[-5 \sum_j \|\hat{q}_t^j - q_t^j\|^2\right]$$

$$r_t^{\text{v}} = \exp\left[-0.01 \sum_j \|\hat{\dot{q}}_t^j - \dot{q}_t^j\|^2\right]$$

$$r_t^{\text{bp}} = \exp\left[-20\|\hat{\mathbf{x}}_t^{\text{base}} - \mathbf{x}_t^{\text{base}}\|^2 - 10\|\hat{\mathbf{q}}_t^{\text{base}} - \mathbf{q}_t^{\text{base}}\|^2\right]$$

$$r_t^{\text{bv}} = \exp\left[-2\|\hat{\dot{\mathbf{x}}}_t^{\text{base}} - \dot{\mathbf{x}}_t^{\text{base}}\|^2 - 0.2\|\hat{\dot{\mathbf{q}}}_t^{\text{base}} - \dot{\mathbf{q}}_t^{\text{base}}\|^2\right]$$

where at timestep t,

- q_t^j represents the local joint rotation of joint j of the robot
- $\dot{q}_t^j$ represents the local joint angular velocity of joint j of the robot
- $\mathbf{x}_t^{\text{base}}$ represents the global position of the robot
- $\dot{\mathbf{x}}_t^{\text{base}}$ represents the linear velocity of the robot
- $\mathbf{q}_t^{\text{base}}$ represents the 3D base rotation of the robot
- $\dot{\mathbf{q}}_t^{\text{base}}$ represents the angular velocity of the robot
- $\hat{q}_t^j$ represents the local joint rotation of joint j from the reference motion
- $\hat{\dot{q}}_t^j$ represents the local joint angular velocity of joint j from the reference motion
- $\hat{\mathbf{x}}_t^{\text{base}}$ represents the global position from the reference motion
- $\hat{\dot{\mathbf{x}}}_t^{\text{base}}$ represents the linear velocity from the reference motion
- $\hat{\mathbf{q}}_t^{\text{base}}$ represents the 3D base rotation from the reference motion
- $\hat{\dot{\mathbf{q}}}_t^{\text{base}}$ represents the angular velocity from the reference motion

The pose reward encourages the robot to match its local joint rotations $\mathbf{q}_t$ with those specified by the reference motion. The velocity reward r_t^{v} encourages the robot to match the joint velocities of the reference motion. The base position reward r_t^{bp} encourages the robot to track the base position and rotation of the reference motion. The base velocity reward r_t^{bv} encourages the robot to move at the same speed as the reference motion.

Appendix C Dynamics Randomization Ranges

Table 3: Ranges of Randomization Parameters used in Simulation

Parameter	Randomization Range
Link mass	$[0.8, 1.2] \times$ default value
Link inertia	$[0.5, 1.5] \times$ default value
Ground friction	$[0.5, 1.25]$
Motor strength	$[0.8, 1.2] \times$ default value
Joint-level PD gains	$[0.7, 1.3] \times$ default value
Motor friction	$[0, 0.05] Nm$
Latency	$[0.0, 0.04] s$

The default values used in the table refer to the values (mass, inertia, etc.) from the underlying morphology—note that during training, these values were themselves produced by randomization within the morphology generation procedure.

Appendix D Training Curriculum

We demonstrate the importance of the training curriculum introduced in Sec. 4. Fig. 7 shows the performance of the controllers over the course of training. The pacing controller trained with a curriculum achieves higher return than the controller trained without a curriculum. The curriculum allows the agent to learn more efficiently by gradually expanding the range of morphology randomization. This prevents the agent from adopting overly conservative behavior when subjected with a large amount of randomization at the beginning of training, potentially leading to a less optimal policy.

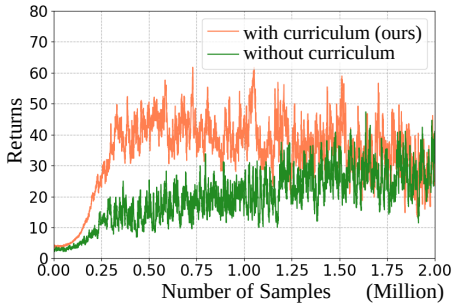


Figure 7: Learning curves comparing GenLoco policies trained with and without a curriculum. The use of a curriculum enables the model to learn the desired task and achieve higher returns much faster than a model trained without a curriculum.

Appendix E Existing Robot Specifications

Parameter	A1	Go1	Laikago	Sirius	Aliengo	Mini Cheetah	ANYmal-B
Total weight (kg)	12	13	24	15.2	21	9	30
Base length (m)	0.27	0.38	0.56	0.44	0.65	0.3	0.53
Base width (m)	0.19	0.09	0.17	0.14	0.15	0.2	0.27
Base height (m)	0.11	0.11	0.19	0.22	0.11	0.09	0.24
Height, fully standing (m)	0.4	0.4	0.6	0.54	0.6	0.3	0.5