Modular Recurrence in Contextual MDPs for Universal Morphology Control

Laurens Engwegen[1,2*] **Daan Brinks**[2] **Wendelin Böhmer**[1]
*Corresponding author, [1]Department of Intelligent Systems, [2]Department of Imaging Physics,
Delft University of Technology, Delft, 2628XE, The Netherlands
{l.r.engwegen,d.brinks,j.w.bohmer}@tudelft.nl

Abstract

A universal controller for any robot morphology would greatly improve computational and data efficiency. By utilizing *contextual* information about the properties of individual robots and exploiting their modular structure in the architecture of deep reinforcement learning agents, steps have been made towards multi-robot control. Generalization to new, unseen robots, however, remains a challenge. In this paper we hypothesize that the relevant contextual information is partially observable, but that it can be inferred through interactions for better generalization to contexts that are not seen during training. To this extent, we implement a modular recurrent architecture and evaluate its generalization performance on a large set of MuJoCo robots. The results show a substantial improved performance on robots with unseen dynamics, kinematics, and topologies, in four different environments.

1 Introduction

Reinforcement Learning (RL) has shown to be very promising for robotic control [Levine et al., 2016, Kalashnikov et al., 2018, Andrychowicz et al., 2020]. In an effort to close the gap with real-world applications, a lot of work has focussed on RL agents that are able to generalize control to different tasks, e.g. manipulating different objects or acting in different environments. Recently, large datasets of robot trajectories have been established and are being used to train such generalizable agents by learning from demonstrations in an offline fashion with foundation models [Brohan et al., 2022, Zitkovich et al., 2023, Vuong et al., 2023]. Several works show promising possibilities for a single model to adapt not only to different scenes and goals, but also to different embodiments that the model has seen demonstrations from [Doshi et al., 2024, Octo Model Team et al., 2024]. Zero-shot generalization to robots that were not seen during training, however, remains very challenging.

Different robots may be suitable for various different tasks and environments. The UNIMAL design space [Gupta et al., 2021] was developed for the evolution of diverse robots that can perform varied tasks. A universal controller that can generalize to any of such robots could drastically improve efficiency. Training (or even fine-tuning) a policy for every new robot that we are interested in requires expensive compute, excessive use of data, and often tedious hyperparameter tuning. The UNIMAL design space, for example, contains more than 1000 different robots. It is therefore suitable for the development and evaluation of RL agents that can control any robot with a single policy. Figure 1 shows a set of robots from the UNIMAL design space.

By framing this problem as a multi-task RL [Vithayathil Varghese and Mahmoud, 2020] problem, each robot can be considered a new task that has some specific *contextual* features, such as the mass of different limbs and the topology of the robot. The goal in this multi-task setting is to learn a universal controller, that can control any robot on basis of its observations and context.

18th European Workshop on Reinforcement Learning (EWRL 2025).

This is not only challenging because robots can have different action and state spaces, but also because robots with different morphologies and/or dynamics might learn tasks in different ways. The multi-task framework allows us to evaluate the performance of an agent during multi-robot training, and its zero-shot generalization performance [Kirk et al., 2023] to new, unseen robots.

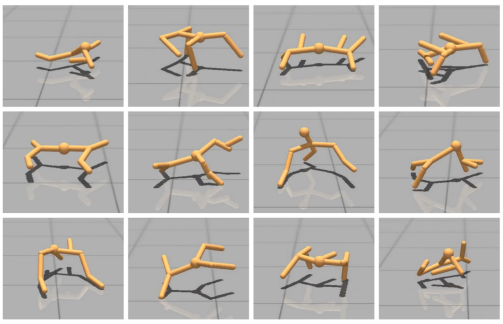

Figure 1: Example of robots that can be found in the UNIMAL design space [Gupta et al., 2021].

Previous work on universal morphology control assumes that the contextual features that describe properties of the robot are fully observable. In this paper, we hypothesize that those features are arbitrary and do not encapsulate enough information to be able to represent the true context needed for optimal (generalizable) control. We build upon previously found effective modular architectures for robotic control [Gupta et al., 2022, Xiong et al., 2023], and address the essentially unobservable context with a recurrent block, while retaining the system's modularity. This paper reports preliminary empirical results that show improved generalization performance to robots that are not seen during training.

2 Background

2.1 Contextual Markov Decision Process

Here, the problem of learning RL policies that are trained on a set of training robots and must generalize to unseen test robots is considered. This problem can be formulated as a Markov Decision Process (MDP) where the agent can only partially access the MDP during training. An MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho)$, with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$, mapping current state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$ to a probability distribution over next states $s_{t+1} \in \mathcal{S}$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and initial state distribution $\rho(s_0)$. A Contextual Markov Decision Process (CMDP [Hallak et al., 2015]) is an MDP where states can be decomposed $s_t = (s'_t, c)$, into the underlying state $s'_t \in \mathcal{S}'$ and a context $c \in \mathcal{C}$. The context is sampled at the start of each episode and remains static until the episode ends. In the current environment, the context defines the robot to be controlled. This framework allows to evaluate zero-shot generalization, that is, generalization to contexts not seen during training, by defining a training and testing set of robots [Kirk et al., 2023].

The UNIMAL design space contains modular robots [Gupta et al., 2021] that are simulated with the MuJoCo physics engine [Todorov et al., 2012]. Such robots consist of a set of nodes, or limbs, that share the same state space and action space, i.e. $\mathcal{S} = \{\mathcal{S}^i | i = 1, ..., N\}$ and $\mathcal{A} = \{\mathcal{A}^i | i = 1, ..., N\}$ for robots with N limbs. Besides the underlying states, a node-level context $\mathcal{C} = \{\mathcal{C}^i | i = 1, ..., N\}$ is provided in the state that describes information of the limbs (e.g. mass, initial position with respect to the parent limb, and the initial position of each joint attached to the limb). Previous methods exploited this modular structure in combination with modern architectures for effective multi-robot control [Huang et al., 2020, Kurin et al., 2021, Gupta et al., 2022, Xiong et al., 2023]. Rather than aiming to develop context-agnostic agents, Gupta et al. [2022] and Xiong et al. [2023] condition the agent on the available contextual information with methods coined *MetaMorph* and *ModuMorph*, respectively. In doing so, they did not only show improved performance during training, but also on generalization to unseen robots. However, the gap between the performance on training and testing robots remains substantial.

2.2 Partially Observable Context

In this paper, we recognize and exploit the fact that relevant contextual information can be partially observable. Robot observations received by the agent contain certain contextual properties of the robot morphology and topology, for example the mass and shape of the different limbs, and their initial positions. The features that are provided, however, depend on what information is available and whether they can be effectively processed in the agent architecture. A lot of (possibly) relevant contextual information is not available, such as the friction and damping of limbs. Additionally, the

true adjancency matrix that describes the organization of the limbs can not be provided effectively to an agent with a modular architecture, as pointed out by Xiong et al. [2023]. Therefore, topological information is provided rather implicitly instead, through the position of a limb with respect to its parent limb. Lastly, we do not have an adequate way of providing the agent with more abstract features, such as the influence of different limbs on each other during movement. It is only through interaction with the environment that the agent can infer such features to perform optimally in the current task. It stands to reason that state-of-the-art approaches, which rely on provided context features for generalization, can only learn these interactions by overfitting to the training contexts.

Although the underlying state is considered to be fully observable in MuJoCo [Todorov et al., 2012], the CMDP is (implicitly) partially observable [Ghosh et al., 2021]. In this setting, the emission or observation function defined in partially observable MDPs [Spaan, 2012], $\phi : \mathcal{S} \rightarrow \mathcal{O}$, maps a state to an observation $o_t \in \mathcal{O}$ that contains the underlying state and the observable context c^+ for every time step t: $o_t = \phi((s'_t, c)) = (s'_t, c^+)$. This is not a problem when we only want to learn control for a set of robots seen during training; with enough representational power, the agent can simply overfit. The challenge arises when the agent encounters robots with new contextual features. In this case, it can be more effective to use experience collected during an episode, as it contains information about the context that is not observable, for example how forces on limbs and actions for different limbs influence each other's underlying state. Incorporating memory into the agent architecture can allow the agent to (implicitly) infer and quickly adapt to necessary unobservable contextual information [Hausknecht and Stone, 2015]. Ideally, such memory mechanism must preserve the modular structure of the agent's architecture.

3 Related Work

3.1 Universal Morphology Control

We build upon previous work aimed at learning an RL policy that can generalize control to different robot morphologies, even when state and action dimensionalities can change. Effective approaches utilized the modularity in robots [Pathak et al., 2019] and introduced weight sharing across different modules [Huang et al., 2020]. Several works adopted graph neural networks [Scarselli et al., 2008, Wang et al., 2018] or transformers [Vaswani et al., 2017, Kurin et al., 2021] as inductive biases to more explicitly infer relationships between different limbs or modules through message-passing. More recently, multi-robot training has been evaluated on a larger scale of different morphologies with the introduction of the UNIMAL design space [Gupta et al., 2021]. Gupta et al. [2022] constructed training and testing sets of robots to evaluate multi-robot control, and showed the effectiveness of a modular transformer-based approach when contextual information is provided to the agent. By further exploiting this contextual information in the agent's architecture, Xiong et al. [2023] demonstrated improved training and generalization performance.

Steps towards robotic applications in the real world also exploit such modular transformer-based architectures for generalizable and transferrable control. Several recent works incorporate additional information about the robot topology through a graph encoding or sparse attention matrices [Patel and Song, 2024, Sferrazza et al., 2024] for improved generalization in simulation and the real world. Fine-tuning and policy distillation approaches for effective generalization have also been proposed for generalization [Przystupa et al., 2025, Xiong et al., 2024]. Lastly, more complex modular architectures were introduced for effective transfer to unseen robots in simulation and the real world [Bohlinger et al., 2024, Li et al., 2024], although evaluating on smaller sets of, and relatively similar robot morphologies. None of these approaches consider this generalization problem as partially observable, where information about the robot to be controlled can be inferred.

3.2 Neural Architectures

The effectiveness of the transformer architecture [Vaswani et al., 2017] for multi-robot control, lies in its capability to model pairwise dependencies between limbs with self-attention. Self-attention can be defined as $A = \sigma(QK^T/\sqrt{d})V$, with query, key and value matrices $Q, K, V \in \mathbb{R}^{N \times d}$, for robots with N limbs and a hidden size of d. Learnable parameters W_Q, W_K and W_V map the input $X \in \mathbb{R}^{N \times d}$ to those matrices, i.e. $Q = XW_Q$, $K = XW_K$ and $V = XW_V$, and $\sigma(\cdot)$ is a row-wise softmax function. In addition to the attention mechanism, Xiong et al. [2023] utilize

hypernetworks [Ha et al., 2016] to more explicitly condition the agent on the available node-wise contextual information. Briefly, they train a hypernetwork that is conditioned on the observable context to generate (1) the parameters of a node-wise encoder that produces X_V with which the value in the subsequent transformer encoder layer is calculated as $V = X_V W_V$, (2) X_Q and X_K, where the query and key matrices are now defined as $Q = X_Q W_Q$, $K = X_K W_K$, respectively, and (3) the parameters of a node-wise decoder that projects the output of the transformer encoder.
Recurrent neural networks (RNNs) like LSTMs [Hochreiter and Schmidhuber, 1997] are useful architectures to deal with partially observable domains in deep RL. Tang and Ha [2021] combined LSTMs with attention to develop systems that can adapt to changes (permutations) of the input. Here, we utilize a variation of this modular architecture to investigate its effectiveness for multi-robot control and zero-shot generalization.

4 Modular Recurrence

4.1 Recurrent PPO

In the current multi-task RL problem, we want to find a universal control policy that is effective for any robot we can encounter in the UNIMAL design space by only training on a set of K training robots. One effective approach to RL in partially observable domains, is to learn an encoding of the belief over the agent's true state. This is often done by, at every time step t, encoding the action-observation history (AOH) $\tau_t^k = (o_0^k, a_0^k, \ldots, o_{t-1}^k, a_{t-1}^k, o_t^k)$ with an RNN, for (in the current domain) robot k the agent is controlling. In this way, the training objective can be formulated as finding parameters θ for policy $\pi_\theta(a_t^k|\tau_t^k)$ that maximize the (discounted) cumulative reward, averaged over all training robots: $\max_\theta \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{H} \gamma^t r_t^k]$, with task horizon H. We implemented a recurrent version of Proximal Policy Optimization (PPO) [Schulman et al., 2017] to optimize this objective.

Recurrent experience replay [Kapturowski et al., 2018], originally developed for DRQN [Hausknecht and Stone, 2015], is used here to effectively sample from the roll-out buffer. In the current setting, episodes can namely be of varying lengths with a maximum of $H = 1000$ time steps. Parallel training on multiple complete trajectories therefore requires padding and can quickly saturate memory. This problem can be solved by storing overlapping chunks of episodes and using a burn-in period for the RNN during training, as introduced by Kapturowski et al. [2018]. The hidden states at the beginning of each chunk are stored and used at the start of the burn-in period. Here, a chunk size of $m = 80$ and a burn-in period of $l = 20$ will be used, as those values were reported to be effective.

4.2 Shared Recurrent Network

Normally, a single recurrent block is introduced in the agent's architecture to encode global actions and observations. To retain modularity, however, we cannot encode the global AOH. Instead, we adopt and adapt a recurrent architecture that processes components of the input separately [Tang and Ha, 2021]: every limb-level action and observation are processed individually through an RNN to encode local AOHs $\tau_t^i = (o_0^i, a_0^i, \ldots, o_{t-1}^i, a_{t-1}^i, o_t^i)$ for every limb i (we omit the superscript that indicates the robot as our policy is controlling only one robot at a time). Since nodes share the same state space, the parameters of this RNN can be shared to increase the scalability of this approach. We only keep track of individual hidden states $h_t^i = \mathrm{RNN}(o_t^i, a_{t-1}^i, h_{t-1}^i)$ that encode the local AOH τ^i, which are initialized with zeros. In this way, the agent can approximate the relevant history for every limb individually to deal with the partially observable CMDP.

There are various ways in which this modular recurrence can be incorporated in the architecture. Here, we implemented architectures that, besides the recurrent aspect, remain close to the previous state-of-the-art methods MetaMorph and ModuMorph. In this way, we can specifically evaluate the effect of treating the CMDP as partially observable and introducing the discussed memory mechanism. As shown in Figure 2, a recurrent version of MetaMorph (**R-MeMo**) encodes the underlying state and previous action with an RNN (shared among all limbs) and separately processes the observable context as this part of the observation remains constant throughout an episode. The recurrent ModuMorph version (**R-MoMo**), shown in Figure 3, uses the same hypernetwork and fixed attention as in the original architecture, but incorporates the shared RNN before the transformer to approximate AOHs from latent encodings of the observation and previous action. In both architectures, a transformer receives the local (AOH) encodings from the RNN to infer relationships between different limbs.

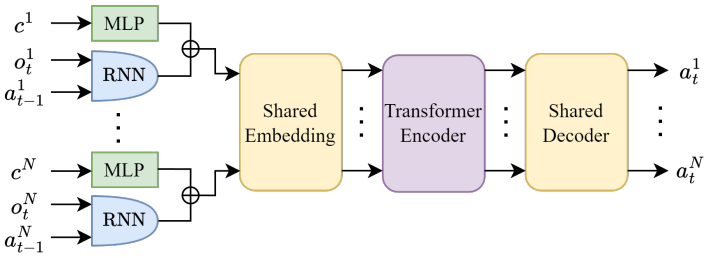

Figure 2: Illustration of the recurrent MetaMorph (R-MeMo) architecture.

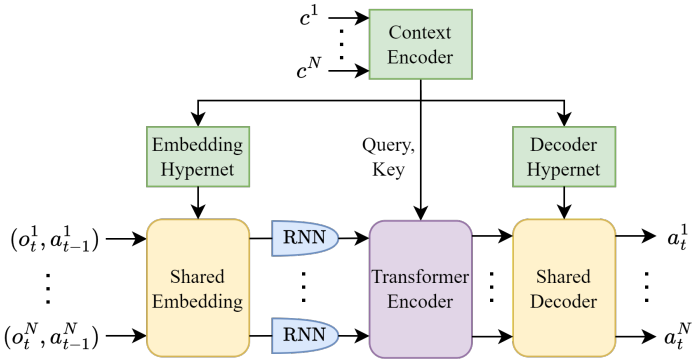

Figure 3: The ModuMorph architecture with added recurrence (R-MoMo).

5 Experiments

In this Section, experiments are performed with MetaMorph, ModuMorph, and their recurrent counterparts R-MeMo and R-MoMo, respectively. We use the MetaMorph version from Xiong et al. [2023], which they report to perform better than the original implementation.

5.1 Experimental setup

The training set of 100 robots, as constructed by Gupta et al. [2022], is used to train agents for multi-robot control. We first evaluate the agent's generalization performance on unseen variations of these training robots, where parameters that influence the dynamics and kinematics are altered (such as the damping of limbs or the angles joints can make). Subsequently, the performance on robots with unseen topologies is evaluated. The provided test set of robots with unseen topologies is randomly split into a validation (32 robots) and test (70 robots) set to experiment with different hyperparameters and evaluate generalization performance. We only validated two values for a regularization hyperparameter to select the models on which we report results. Xiong et al. [2023] namely found that this parameter can have a big impact on performance. See Appendix A for further details.

Agents are trained and evaluated in four different environments. In each of those, the agent has to maximize the robot's locomotion distance. In **Flat Terrain**, the agent needs to traverse a flat surface, while in **Incline** the robots are to be controlled on a surface that is inclined by 10 degrees. **Variable Terrain** contains a sequence of hills, steps and rubble, interleaved with flat terrain. Those sequences are randomly generated at the start of each episode. Finally, **Obstacles** is a flat terrain with randomly generated obstacles. In the latter two environments, the agent receives a 2D heightmap of its close surrounding, in addition to proprioceptive and contextual observations, to be able to react to changes in terrain. For more details on the environments, we refer to Gupta et al. [2021].

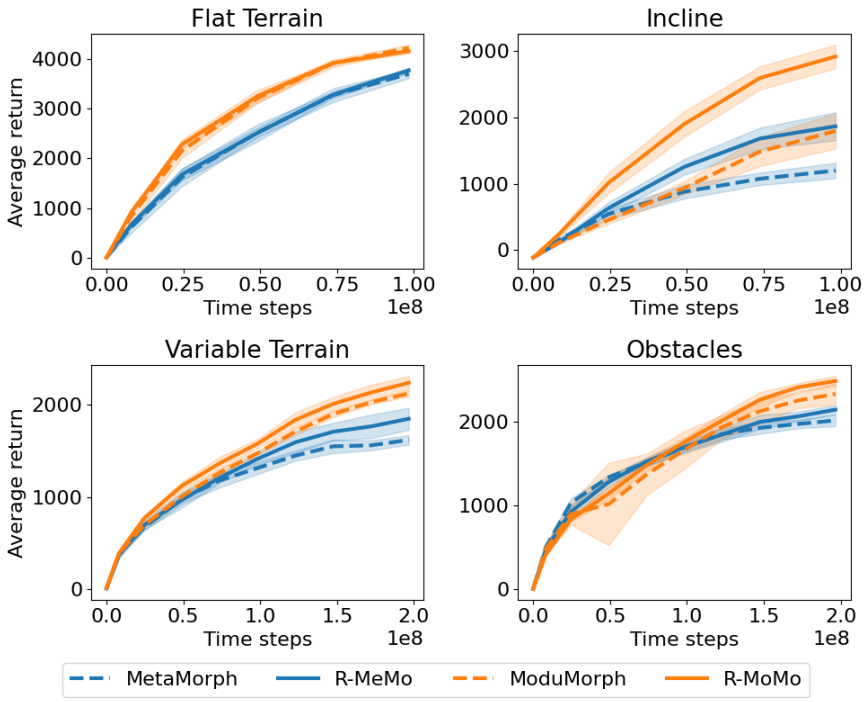

Figure 4: Training performance on the 100 training robots in the different environments. Mean and standard deviation over 5 seeds are shown.

5.2 Multi-Robot Training Performance

The training performance of the different methods on the 100 training robots, averaged over 5 seeds, is shown in Figure 4. In Flat Terrain, MetaMorph and ModuMorph perform equally well as their recurrent versions. In the other environments, the addition of modular recurrence results in higher overall performance, particularly in the Incline environment. In general, ModuMorph seems to perform better than R-MeMo on the training robots (except on Incline), illustrating the effectiveness of the hypernetworks conditioned on the available context during multi-robot training. However, the addition of modular recurrence, like in R-MoMo, can even further improve training performance.

5.3 Zero-Shot Generalization to Different Dynamics and Kinematics

Gupta et al. [2022] constructed a set of robots that have the same topologies as the training robots, but differ in a contextual feature, to evaluate zero-shot generalization to different dynamics or kinematics. For each training robot, they created four test robots with variations in armature, damping, gear, density, limb shapes or joint angles, resulting in a new set of 2400 test robots.

Figure 5 shows the performance of the four evaluated methods on the test robots, for each of the changed dynamics and kinematics parameters. Over all changes, R-MoMo obtains a higher average return than ModuMorph, consistently throughout the different environments. Similarly, R-MeMo performs better than MetaMorph, and in some environments and parameter changes even outperforms ModuMorph. These results demonstrate improved generalization performance with the recurrent methods.

5.4 Zero-Shot Generalization to Unseen Robot Topologies

After training on the 100 training robots, the methods were evaluated on the 70 test robots with unseen topologies. The averaged returns of the different methods in the four environments are denoted in Table 1. In each of the environments, ModuMorph and/or R-MoMo dominate training, but R-MoMo significantly outperforms the other methods in zero-shot generalization to the test robots. Interestingly, in all environments (except *Incline*) ModuMorph and R-MoMo perform roughly the same during

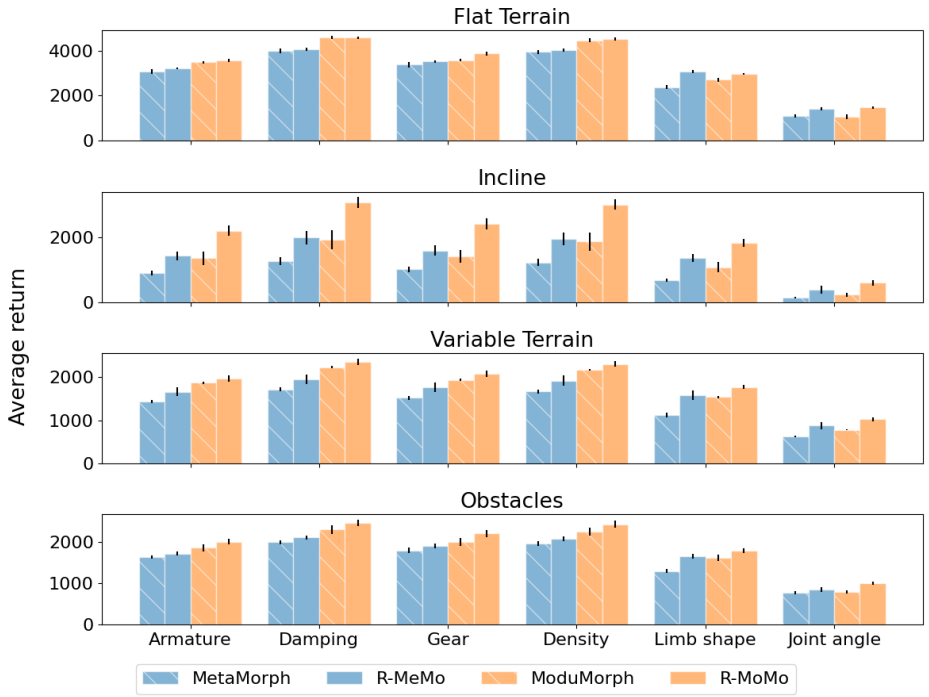

Figure 5: Test performance on training robot topologies with changes in contextual features that result in different dynamics or kinematics. Results are averaged over 5 seeds with the standard deviation shown as error bars. Note that our recurrent architectures improve the generalization to different dynamics in all environments, albeit not always significantly.

Table 1: Average training performance on 100 seen robots and zero-shot generalization performance on 70 unseen test robot topologies in different environments. Mean and standard deviation over 5 seeds are denoted. Significantly better performing algorithms for each environment are in bold face.

		MetaMorph	R-MeMo	ModuMorph	R-MoMo
Training	Flat Terrain	3697 ± 137	3819 ± 61	$\mathbf{4230 \pm 101}$	$\mathbf{4186 \pm 58}$
	Incline	1190 ± 112	1870 ± 202	1885 ± 194	$\mathbf{2910 \pm 173}$
	Variable Terrain	1606 ± 63	1894 ± 104	$\mathbf{2126 \pm 27}$	$\mathbf{2251 \pm 100}$
	Obstacles	2029 ± 87	2134 ± 63	$\mathbf{2359 \pm 119}$	$\mathbf{2519 \pm 88}$
Testing	Flat Terrain	1384 ± 92	1678 ± 87	1455 ± 94	$\mathbf{1829 \pm 97}$
	Incline	355 ± 95	576 ± 58	414 ± 159	$\mathbf{882 \pm 131}$
	Variable Terrain	710 ± 58	994 ± 75	972 ± 83	$\mathbf{1158 \pm 26}$
	Obstacles	874 ± 63	1067 ± 47	1024 ± 60	$\mathbf{1269 \pm 72}$

training, but the recurrent version performs significantly better in zero-shot generalization. This indicates that R-MoMo does not just learn a better policy, but a policy that *generalizes* much better to unseen robots. The recurrent version of MetaMorph shows a similar improvement on test robots for all environments, but also outperforms its baseline in more training environments, which makes it less clear that the advantage comes from better generalization. In comparison, R-MeMo performs roughly on a par with vanilla ModuMorph on test robots.

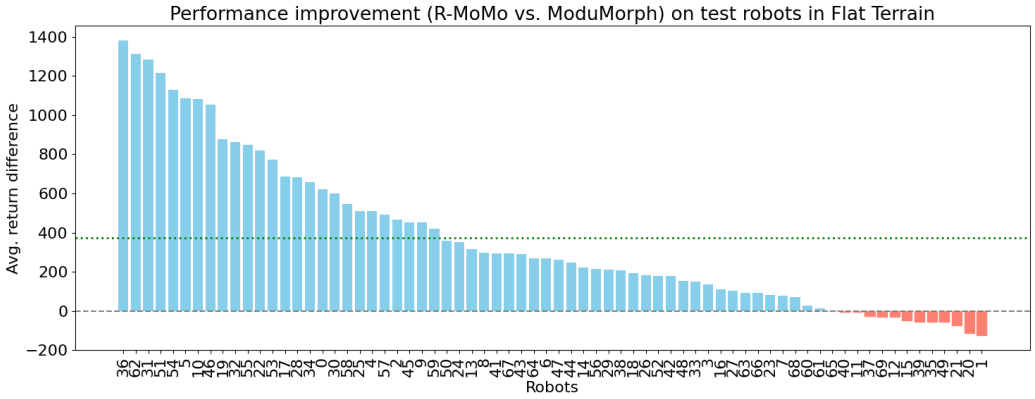

Figure 6: The difference in return between R-MoMo and ModuMorph for each of the 70 unseen test robots in the Flat Terrain environment. Returns are averaged over 5 seeds. The mean performance improvement over all test robots is indicated by the green dotted line.

Returns in each environment can range from very low (< 0) to very high (> 4000) values. Averaged returns over 70 test robots could therefore be misleading, as differences can be caused by only a small number of test robots. We therefore additionally report the per-robot difference in return (averaged over 5 seeds) between R-MoMo and ModuMorph on the Flat Terrain environment in Figure 6. These results show a consistent improvement over a majority of the test robots, illustrating the increased generalizability. Per-robot performance comparisons for the other environments can be found in Appendix B.

6 Discussion

This work explores improved generalization in multi-robot control. It was hypothesized that modular recurrence could allow the agent to infer relevant unobservable context to improve performance. The results have shown a consistent increase in generalization performance when such memory-mechanism was introduced across different environments, for robots with different dynamics, kinematics, and topologies. Whether the agent actually uses the RNN to infer relevant contextual features remains to be investigated. A limitation of the explored recurrent architecture, is the fact that hidden states have to be stored for each limb. Scaling to robots with a large number of limbs requires therefore more memory. Besides, the sequential processing of RNNs results in longer training times (although minimized by efficient batch processing through chunking episodes). An interesting direction for future research would therefore be to investigate a more efficient memory-mechanism in the architecture. Nonetheless, the combination of modular recurrence with transformers can be promising in domains with a graph-like structure of which the relevant properties are only partially observable.

Acknowledgements

This project was funded by the Delft AI initiative within the BIOlab. This work was also partially funded by the Dutch Research Council (NWO) project *Reliable Out-of-Distribution Generalization in Deep Reinforcement Learning* with project number OCENW.M.21.234. DB acknowledges support from an ERC Starting Grant (850818 - MULTI-VIsion) and the Convergence Health and Technology (Integrative Neuromedicine flagship).

References

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

Ria Doshi, Homer Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature Communications*, 12(1):5721, 2021.

Nelson Vithayathil Varghese and Qusay H Mahmoud. A survey of multi-task deep reinforcement learning. *Electronics*, 9(9):1363, 2020.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76: 201–264, 2023.

Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. Metamorph: Learning universal controllers with transformers. *International Conference on Learning Representations*, 2022.

Zheng Xiong, Jacob Beck, and Shimon Whiteson. Universal morphology control via contextual modulation. In *International Conference on Machine Learning*, pages 38286–38300. PMLR, 2023.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes, 2015.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020.

Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Böhmer, and Shimon Whiteson. My body is a cage: the role of morphology in graph-based incompatible control. *International Conference on Learning Representations*, 2021.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. *Advances in Neural Information Processing Systems*, 34:25502–25515, 2021.

Matthijs TJ Spaan. Partially observable markov decision processes. In *Reinforcement learning: State-of-the-art*, pages 387–414. Springer, 2012.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: a study of generalization via modularity. *Advances in Neural Information Processing Systems*, 32, 2019.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Austin Patel and Shuran Song. Get-zero: Graph embodiment transformer for zero-shot embodiment generalization. *arXiv preprint arXiv:2407.15002*, 2024.

Carmelo Sferrazza, Dun-Ming Huang, Fangchen Liu, Jongmin Lee, and Pieter Abbeel. Body transformer: Leveraging robot embodiment for policy learning. *arXiv preprint arXiv:2408.06316*, 2024.

Michael Przystupa, Hongyao Tang, Martin Jagersand, Santiago Miret, Mariano Phielipp, Matthew E Taylor, and Glen Berseth. Efficient morphology-aware policy transfer to new embodiments. *Reinforcement Learning Journal*, 2025.

Zheng Xiong, Risto Vuorio, Jacob Beck, Matthieu Zimmer, Kun Shao, and Shimon Whiteson. Distilling morphology-conditioned hypernetworks for efficient universal morphology control. *International Conference on Machine Learning*, 2024.

Nico Bohlinger, Grzegorz Czechmanowski, Maciej Krupka, Piotr Kicki, Krzysztof Walas, Jan Peters, and Davide Tateo. One policy to run them all: an end-to-end learning approach to multi-embodiment locomotion. *arXiv preprint arXiv:2409.06366*, 2024.

Boyu Li, Haoran Li, Yuanheng Zhu, and Dongbin Zhao. Mat: Morphological adaptive transformer for universal morphology policy learning. *IEEE Transactions on Cognitive and Developmental Systems*, 16(4):1611–1621, 2024.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Yujin Tang and David Ha. The sensory neuron as a transformer: Permutation-invariant neural networks for reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 22574–22587, 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*, 2018.

A Hyperparameters

In our experiments, we use the same hyperparameter values as in MetaMorph and ModuMorph. We only evaluate two different values for a regularization parameter on the validation set. Xiong et al. [2023] namely argued that this parameter can have a big impact on performance. This parameter defines the maximum approximate KL-divergence between the old and the new policy for every mini-batch before the update step. If this value is exceeded, the iteration of updates ends, and we return to sampling new trajectories. Figure 7 shows the average performance of the different methods with the two tested values (3 and 5) that were also evaluated in ModuMorph. In most cases, there is not a big difference in performance; standard deviations often overlap. For every method, the value that resulted in the highest average return on the validation set was used for the reported results in Section 5.

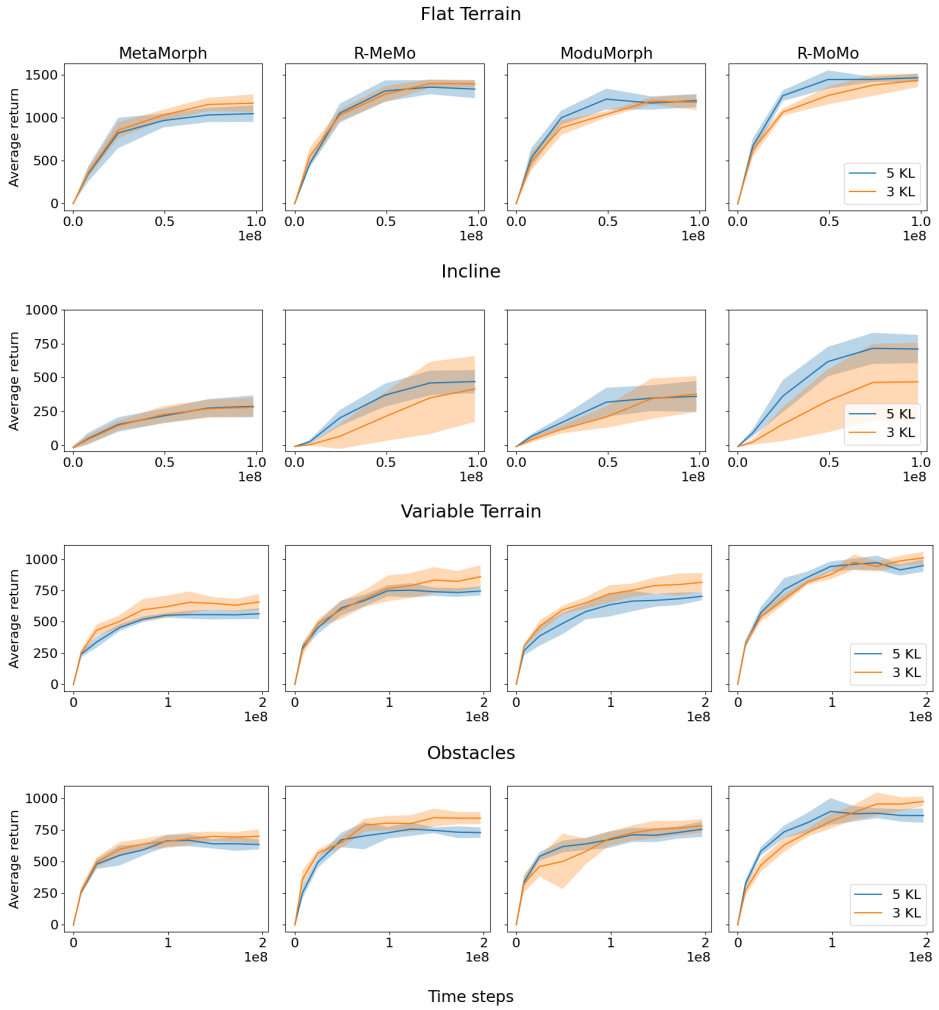

Figure 7: Test performance on the 32 robots in the validation set for different values of the regularization parameter that defines the maximum approximate KL-divergence between the old and the new policy (e.g. 5 KL corresponds to a max. approximate KL-divergence of 5.0), averaged over 5 seeds with shown standard deviation.

We additionally experimented with two versions of recurrent experience replay for R-MeMo and R-MoMo, in which we either reset hidden states at the start of each stored chunk in the roll-out buffer, or store the initial hidden state of each chunk. The latter showed a more stable training performance and was therefore used. This is to be expected, because a zero initialization at every chunk requires the agent to recover a meaningful hidden state during the burn-in period, in which it can only depend on transitions from the roll-out buffer. We therefore report results with stored hidden states.

B Zero-shot generalization performance comparison

In Figure 8, the per-robot average return difference between R-MoMo and ModuMorph is visualized on the Incline, Variable Terrain, and Obstacles environments. These results are consistent with those in Flat Terrain (Figure 6), where R-MoMo performs better on a majority of robots, while only obtaining a slightly lower return on a small set of robots.

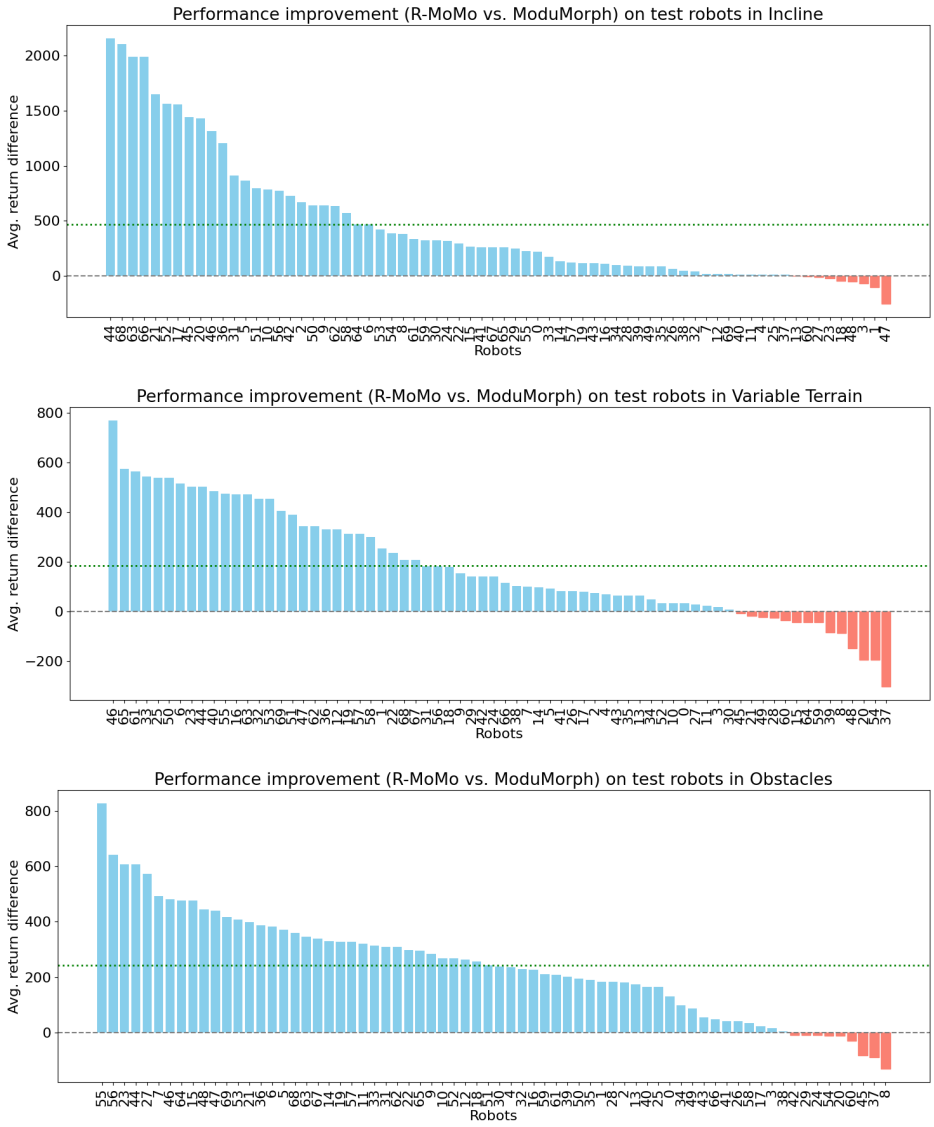

Figure 8: The difference in return between R-MoMo and ModuMorph for each of the 70 unseen test robots in the Incline, Variable Terrain and Obstacle environments. Returns are averaged over 5 seeds. The mean performance improvement over all test robots is indicated by the green dotted line.

Figure 9 shows the per-robot return difference between MetaMorph and ModuMorph, as a comparison for the differences between R-MoMo and ModuMorph. It is visible that, across all environments, ModuMorph performed better than MetaMorph on less robots than R-MoMo did with respect to ModuMorph.

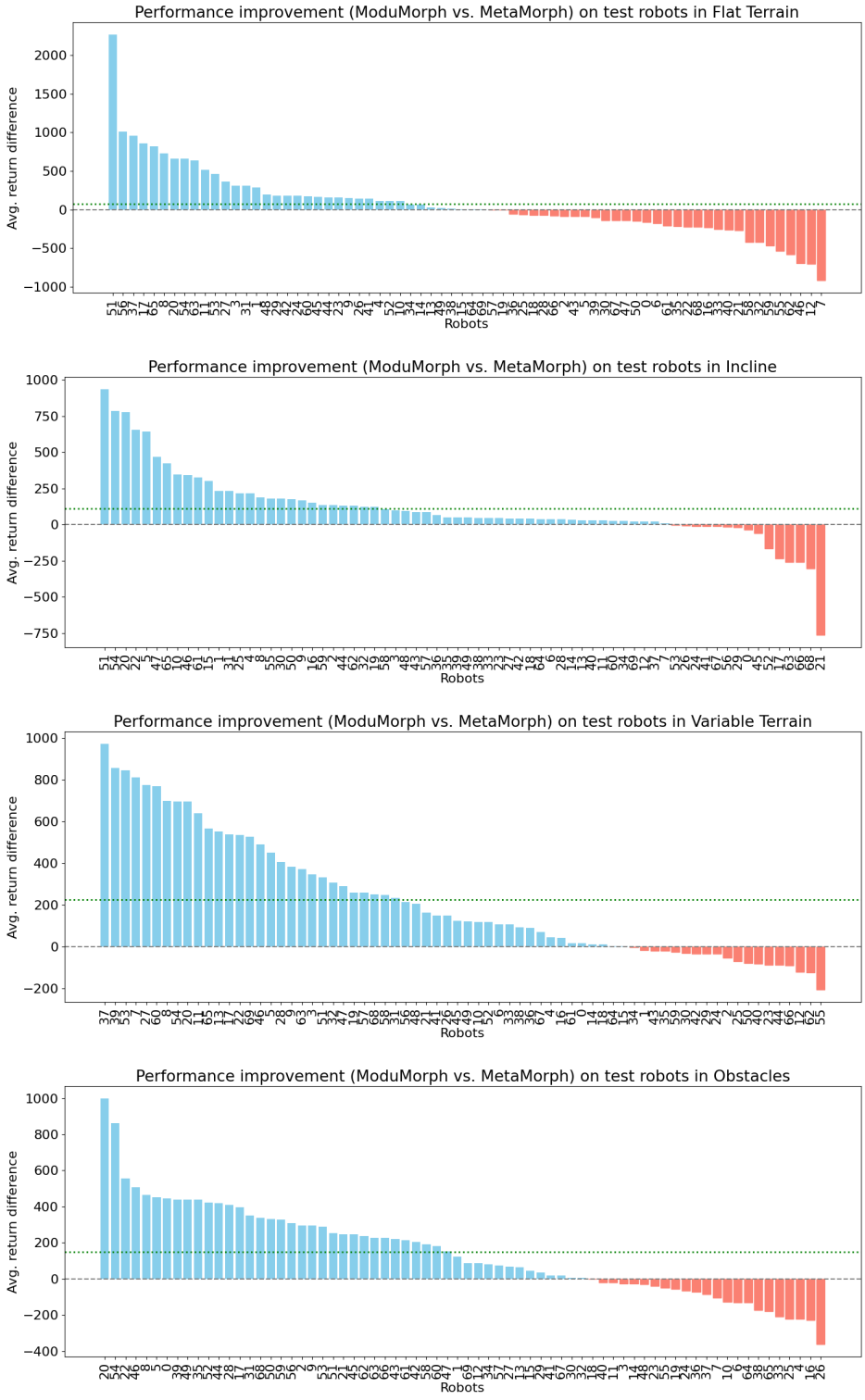

Figure 9: The difference in return between ModuMorph and MetaMorph for each of the 70 unseen test robots in the four environments. Returns are averaged over 5 seeds. The mean performance improvement over all test robots is indicated by the green dotted line.