THE EFFECTS OF VOLUMETRIC MORPHOLOGY ENCODINGS ON ONLINE UNIVERSAL CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Universal control methods tackle the problem of learning to control diverse robot morphologies from online interactions, using reinforcement learning. Recent works have demonstrated that proper conditioning on morphological context is key to obtaining good performance across these incompatible Markov decision problems, i.e. problems with varying input and output spaces. In this work, we identify the robot's kinematic context as a particularly important portion of the overall contextual information. To provide the body shape portion of the kinematic context to universal controllers, existing methods have leveraged the strong assumption that robot bodies are composed of simplified geometric primitives and have directly included the descriptors of those shapes in the context vector. In this paper, we propose volumetric morphology encodings (VME). These novel representations capture the body shape and connectivity of the robot, and are learned during a pretraining phase, from a 3D robot body shape dataset that we collect. We use VME embeddings to improve the training of universal controllers, on challenging online universal control tasks. Even if we only alter the body shape portion of the kinematic portion of the overall robot context, we observe substantially higher performance than baseline methods, with higher gains in harder tasks. Furthermore, our approach demonstrates superior zero-shot transfer to unseen embodiments.

1 INTRODUCTION

The idea of a single policy capable of controlling a diversity of distinct robots is compelling. One approach to achieving this goal is universal control (UC), which casts this as a multi-task online reinforcement learning problem. The policy must solve the control task for multiple robots with diverse sensors, actuators, and morphologies, using a single neural framework. Such a policy can drastically reduce per-robot data collection requirements, amortize costly pre-training, and enable rapid deployment to novel embodiments (Gupta et al., 2022; Kurin et al., 2021; Trabucco et al., 2022).

However, achieving this requires overcoming at least two fundamental challenges: architecture design and morphological representation. The first concerns the design of learning systems capable of addressing the incompatible control problem (Kurin et al., 2021), namely, learning policies that operate across robots with heterogeneous observation and action spaces. While previous approaches to this problem have relied on graph neural networks (GNNs) (Huang et al., 2020), the state of the art has shifted towards context-modulated transformers (Vaswani et al., 2017; Xiong et al., 2023). The second fundamental challenge lies in how to effectively convey a robot's morphological context to the policy so that its behaviour can be appropriately modulated. This problem has received comparatively less attention, with most prior work using tokens built from the ground truth robot description (Kurin et al., 2021; Gupta et al., 2022; Huang et al., 2020; Trabucco et al., 2022).

Xiong et al. (2023) showed that contextual conditioning plays a pivotal role in universal control. By properly incorporating information about a robot's embodiment, universal control policies can achieve significant improvements in both training efficiency and generalization. The morphological context is composed of two parts: *kinematics* (e.g. geometry, joints, connectivity) and *dynamics* (e.g. mass, actuation). In this paper, we identify kinematics context in particular as contributing much of the performance gains previously attributed to the full morphological context. Some aspects of kinematics are easily accessible as ground truth: the axis of the robot's links, the type of motor that is used, the joint types. But to provide the body shape and connectivity information, most prior

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

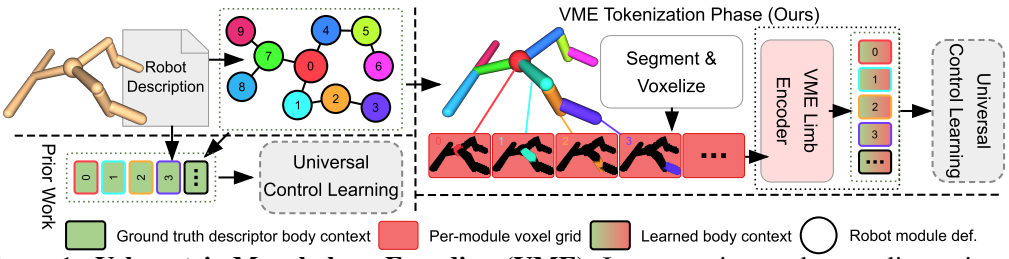


Figure 1: **Volumetric Morphology Encoding (VME)**: In most prior work on online universal control, the robot description file is read and parsed into a graph structure. The parametric body shape description is then directly used as the kinematics context vector fed to the controller. We instead propose to learn large parts of the kinematics context from the robots' 3D body shapes. *Here pictured: a* UNIMAL *robot (Gupta et al., 2021).*

work relies on the strong assumption that robots are composed of primitive shapes like capsules and spheres, and leverages these shape's ground truth description vectors to build the contextual tokens. A key insight of our work is that body shapes are inherently 3D, and should therefore be encoded using 3D data rather than descriptor surrogates. However, this seems to be an under-studied aspect of the online universal control problem. The open problem is the design of the 3D representation itself, such that it still respects the context format established by prior work in universal control.

We propose a standalone encoder model for body shapes that we call a "volumetric morphology encoder" (VME). The volumetric morphology encodings that it produces capture the volumetric aspects of robot morphology, which is to say a subset of the overall kinematic context: including limb shape, size, position, and connectivity. We train the VME vectors as part of a 3D convolutional autoencoder, which learns to compress a dataset of 3D scans of robot body plans (which we present limb-per-limb as a set of voxel grids) into a latent space that is shared across all robots in the dataset. The resulting body-shape context tokens are then fed into state-of-the-art universal control architectures (Gupta et al., 2022; Xiong et al., 2023) and trained with proximal policy optimization (PPO) (Schulman et al., 2017). We demonstrate on standardized UNIMAL tasks (Gupta et al., 2021) that by altering only the **body shape (limb shape, connectivity) portion** of the **kinematic portion** of the **overall context vector**, we obtain substantially varied final performances, supporting our claim that there is significant value in improving volumetric morphology representations. We observe that providing VME context yields performance improvements on most tasks, with higher gains on harder tasks. We also observe improved generalization to previously unseen robot embodiments, for both domain-randomized variants of seen robots and on entirely unseen robot morphologies.

2 Problem Definition

We address the problem of learning a single control policy for a collection M of diverse *morphological graphs*. We assume $|M|$ to be large. Each $m \in M$ is a robot tree structure composed of N_m limb nodes, where the edges represent the joints between the limbs. Each m defines a function $m : \mathcal{K}_m \times \mathcal{D}_m \rightarrow \mathcal{R}_m$ that instantiates a robot $r = m(\mathbf{k}_r, \mathbf{d}_r)$, with $r \in \mathcal{R}_m$, where $\mathbf{k}_r \in \mathcal{K}_m$ are the *kinematic parameters* encoding limb geometry and joints, and $\mathbf{d}_r \in \mathcal{D}_m$ are the *dynamic parameters* encoding inertial and actuation properties. There are three disjoint robot sets of interest. The policy trains on R_{train}, which are instances of the set of training graphs M_{train}. The policy can then be deployed on the R_{unseen} set (other, unseen robots sourced from the same M_{train}), or transferred to R_{test}, which contains instances of M_{test}, morphologies not seen during training.

Each robot r is modelled as a contextual Markov decision process (CMDP) (Hallak et al., 2015), a tuple $r = \langle S_r, A_r, C_r, T_r, \mathfrak{R}_r \rangle$, consisting of the state space S_r, the action space A_r, the task context C_r, the transition function T_r, and the reward function $\mathfrak{R}_r$. The elements of the CMDP are fixed for a given m but altered by the morphological parameters of the robot r. While the state $s_{r,t}$ is time-varying, the *morphological context* c_r is time-invariant and defined prior to universal control training. Both $s_{r,t}$ and c_r can be indexed by $i \in N_r$ to obtain the limb-specific $s^i_{r,t}$ and c^i_r, as necessary for existing universal control approaches as explained in Appendix A.5.1. At time step t, robot r has state $s_{r,t} \in S_r$, takes action $a_{r,t} \in A_r$, and receives reward $\mathfrak{r}_{r,t}$. Given a fixed task horizon H, the objective is to train a policy $\pi_\theta(a_{r,t}|s_{r,t}, c_r)$ that maximizes the average return across all training robots R_{train} (Gupta et al., 2022):

$$J(K) \triangleq \max_{\theta} \frac{1}{|R_{\text{train}}|} \sum_{r \in R_{\text{train}}} \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{H} \mathfrak{r}_{r,t} \,\middle|\, c_r = [k_r; d_r] \right] \tag{1}$$

The morphological context $c_{r,t}$ is composed of the kinematics context $k_r \triangleq [k_r^1 ... k_r^{N_r}]^T$ and the dynamics context $d_r \triangleq [d_r^1 ... d_r^{N_r}]^T$. They repsectively *represent* the underlying true kinematics and dynamics parameters in a way that helps train π_θ, but they do not necessarily capture the full kinematics or dynamics information (i.e. context vectors providing *partial* information are admissible here). While the set of kinematics context vectors $K \triangleq \{k_r\}_{r \in R_{train}}$ and the set of dynamics context vectors $D \triangleq \{d_r\}_{r \in R_{train}}$ are both important (though not equally so, as evidenced in Section 4), K in particular presents a difficult representation design problem because it aims to capture 3D data. Our objective, which is upstream from Eq. (1), is to find the best representation of K :

$$K^* = \arg\max_{K} \; J(K) \tag{2}$$

However, this max-max optimization problem is intractable given the complexity of the UC problem, its high computational cost, the desire to preserve UC sample efficiency, and the diversity of the morphological space. Consequently, we must instead *design* K, such that it not only improve the optimization of universal controllers with respect to Eq. (1) as defined by Eq. (2), but also equips them to learn in a manner that facilitates generalization to R_{unseen} and transfer to R_{test}.

Robot Definition Files: Classical robotics, robot simulation, and robot learning communities all use one standardized format to represent robots: universal robot definition files (URDF). The morphological graph m is expressed as an XML specification where nodes represent rigid *links* and edges represent articulated *joints*. The morphological parameters are expressed as attributes on the nodes $i \in N_m$ and their edges; by altering these attributes without altering the graph itself, we can obtain different concrete robots r. The vast majority of previous UC methods have assumed the shape of the robot's limbs to be simple 3D primitives expressible as descriptor vectors (like capsules, spheres, etc.). As shown in Figure 1, these descriptor vectors were then directly loaded as the kinematics context token k_r (Xiong et al., 2023; Gupta et al., 2022; Huang et al., 2020; Trabucco et al., 2022; Kurin et al., 2021; Wang et al., 2018). *Our core hypothesis is that we can **learn** k_r representations directly from the 3D data resulting from the URDF's description of the robot body, and that these **latents** will be more useful for UC policy training.*

3 Related Work

Universal control (UC) is a rich area of the larger cross-embodiment control literature where neural controllers learn high-frequency low-level control online, through interactions with a simulator (Huang et al., 2020; Wang et al., 2018; Kurin et al., 2021; Trabucco et al., 2022; Gupta et al., 2022; Xiong et al., 2023). These methods assume the set of morphological graphs M and associated set of robots R to be large, and so dedicated strategies must be leveraged to tractably represent the context for each so many different CMDPs. Early approaches modelled robots as graphs m and used graph neural networks (GNNs) to pass messages along kinematic chains (Wang et al., 2018; Huang et al., 2020). By altering these graphs into uninformative shapes, later work showed that GNNs provided limited benefit, proposing instead token-based representations with transformer controllers (Kurin et al., 2021). In MetaMorph (Gupta et al., 2022), the neural architecture is altered and applied to larger sets M of morphological graphs and associated robots R. Building upon MetaMorph, ModuMorph (Xiong et al., 2023) identifies contextual modulation capacity as a major bottleneck. More details on MetaMorph and ModuMorph are given in Appendix A.5.1. The URMA (Bohlinger et al., 2024) architecture proposes a training regime similar to those used by other online UC methods, but it has not been demonstrated in the large-M regime considered in online UC: URMA considers only $|M| = 4$ and $|R| \approx 20$. In contrast to the vast majority of online UC methods, however, URMA **has** been successfully deployed on a real robot, making it one of the few online UC works we are aware of to demonstrate real-world deployment. *In our work, we identify a specific part of the context vector (the kinematic context) as a critical contextual signal.*

Cross-embodiment control. Offline cross-embodiment learning methods such as certain vision-language-action (VLA) models (Doshi et al., 2024) have seen recent real-world success. However,

they operate under different assumptions than the online UC setting that we consider. First, they use offline training over very large robot interaction datasets, with visuomotor control (often with action chunking). Second, works in this area tend to assume relatively small M and R. For instance, π_0 (Black et al., 2024) trains on about a dozen morphological graphs ($|M_{\text{train}}| \approx 12$, (O'Neill et al., 2024)) and tests on $|M_{\text{deploy}}| = 4$ (single-arm and bimanual, 6DoF and 7DoF manipulators; where $M_{\text{deploy}} \subset M_{\text{train}}$). Third, due to the assumption of small M, it is tractable for methods such as CrossFormer (Doshi et al., 2024) and Octo (Team et al., 2024) to avoid morphological representation entirely by using morphology-specific (or robot-specific) action heads. Meanwhile, UC can consider $|M_{\text{train+test}}| \approx 200$ (Xiong et al., 2023), which would require ≈ 200 different action heads, which is neither tractable nor desirable. Furthermore, UC considers zero-shot transfer to wholly unseen robots where $m \notin M_{\text{train}}$, for which by definition it would not be possible to have an available action head. One example of an offline cross-embodiment method that supports large R is "embodiment scaling laws" (ESL) (Ai et al., 2025), with $|R| \approx 1000$. ESL is based on URMA, however, ESL reduces M from URMA's four morphologies to three (hexapod, quadruped, and humanoid), and adopts an offline training regime typical of cross-embodiment control, placing it outside the large-M large-R online UC setting that we consider. Indeed, ESL utilizes simplified morphological context (only information about robot feet at rest), yet we can empirically show that good context is absolutely necessary in our setting (see Figure 10).

Robot body representations. Most existing approaches for learned robot body representation were designed for the task of robot evolution or co-evolution and rely on structural assumptions about the robot design space. For example, Hu et al. (2022) learn latents tied to a specific robot design grammar, and uses those latents to generate new robots. Like us, Li et al. (2025) uses a voxel autoencoder to represent robots, but considers a specific soft robotics design space. Also, the latents are not used for control: they are used as representations for evolutionary methods such as Hansen (2016). This allows for larger robot representation latent spaces ($d = 512$, which for e.g. 16 limbs is a large $k_r = 8192$), unsuitable for UC (ModuMorph has $d \sim 50$, which for e.g. 16 limbs is a reasonable $k_r = 800$). BodyTransformer (Sferrazza et al., 2024) is an interesting case: successfully leveraging body information in a single-robot learning setting. There also exist related work designed **for gripper or manipulation representations** (e.g., UniGrasp (Shao et al., 2020), AdaGrasp (Xu et al., 2021)). While these works provide evidence that learned 3D representations can improve neural robotic decision-making, they tend to consider M to be small. For instance, while UniGrasp procedurally generates $|R| = 2000$ gripper shapes, it holds $|M| \approx 2$ (two and three fingered grippers). GET-Zero (Patel & Song, 2025) ablates over the number of links in a robot hand, enabling cross-embodiment generalization (larger M, but only subsets of a given fixed "full" hand). Like most works on UC (Kurin et al., 2021; Gupta et al., 2022; Xiong et al., 2023; Huang et al., 2020; Wang et al., 2018; Xiong et al., 2024; Furuta et al., 2023), we rely on specialized locomotion benchmarks, where it is possible to learn over hundreds of different robot graphs and varied limb geometries (combining train, test, and variant sets, Gupta et al. (2021) has $M \approx 200, K \approx 2500$). We note that we do not expect the latents we propose for the online low-level high-frequency UC setting to outperform dedicated gripper and grasp representations in manipulation tasks.

4 Motivating Experiment

Prior work in universal control has established the critical importance of contextual modulation (Kurin et al., 2021; Xiong et al., 2023). In this section, we establish that not all context is equally important. We design a motivating experiment for this purpose. Intuitively, we should expect that it is possible for a human to walk slowly without reasoning very much about about our bodies since our arms and upper body are not particularly needed to maintain balance. In contrast, when sprinting, full-body reasoning about the mass, strength, and shape of our limbs becomes much more important to shift our centre of mass and maintain balance. We aim to use this intuition to show how *dynamic* and *kinematic* context differently impact locomotion performance in universal control. We select the "variable terrain" (VT) task used by prior work (Xiong et al., 2023). The reward is directly proportional to the robot's x-axis velocity (its "running speed"). We introduce a target velocity as per Eq. (4), which we gradually increase to obtain a suite of tasks, from easiest (0.5 m/s) to hardest (3.5 m/s).

In Figure 2, we train five seeds of a ModuMorph controller (Xiong et al., 2023) per target velocity, and we vary the type of provided robot context. `D` context only provides dynamics information (limb mass, actuator strength, etc.) while `K` only provides kinematic information (limb shape, size, connectivity, joint type, etc.) Both `D` and `K` context vectors are read without modification from the robots' URDF

files as described in Section 2. The D+K context concatenates D and K, which corresponds exactly to Xiong et al. (2023). This makes for 75 policies total, each trained under the regime described in Section 6. In Appendix B.1, to confirm that contextual modulation is vital for universal control, we also provide results for entirely absent context.

We report the robot's final success rate during training (the average episodic fraction of timesteps at which the running speed exceeds the target). The shaded area is the standard error. When the task is relatively easy (e.g. 0.5m/s), the contents of the context is less consequential. As the target velocity increases, context becomes increasingly important. We find that K tracks D+K's performance curve much more closely than D context can, with their shaded areas overlapping throughout. This is a strong signal that the shape of the robot is a more critical contributor of overall performance than the dynamics context. From the perspective of classical robotics, the dominance of kinematic context over dynamic context is intuitive: *dynamics* is built upon *kinematics*. The kinematic structure (the rigid bodies and their joint connectivity) must be established first. Only then can the system's dynamic properties (e.g. mass, inertia, and actuation strength) be applied to solve the equations of motion and compute resulting accelerations. In this work, we only modify the **body shape/connectivity portion** of the **kinematics portion** of the **overall context vector**. Yet, as demonstrated in Section 7, this is sufficient to produce substantially different downstream performance.

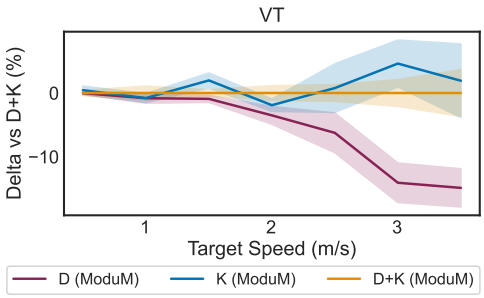


Figure 2: **Importance of Kinematics Context**: Context ablation on the variable terrain (VT) task (see Figure 3). K denotes kinematics context, D denotes dynamics, and their concatenation D+K is exactly ModuMorph (Xiong et al., 2023). K context alone noisily tracks the full D+K context for most target velocities (shaded areas overlap). We show how our proposed method performs on this experiment in Figure 11.

5 Volumetric Morphology Encodings

In Section 4, we obtained evidence that the kinematic context k_r is a critical piece of the overall context vector. To provide k_r, prior work has posed the strong assumption that the robots are composed of primitive shapes (spheres, capsules, etc.), and since these shapes are expressible as descriptor vectors, these vectors were fed directly to the UC controller as context (Wang et al., 2018; Huang et al., 2020; Kurin et al., 2021; Gupta et al., 2022; Xiong et al., 2024). While these descriptors contain the robot's ground truth physical structure, we argue from the perspective of representation learning that two context tokens containing identical information might not be equivalent for a learner: the format of the data matters. For instance, in the domain of Atari RL (Bellemare et al., 2013), high-dimensional visual representations can sometimes support better policy learning than compact, ground-truth RAM states because visual inputs allow models to to identify emergent patterns (e.g., CNN translational equivariance (LeCun et al., 2010)). A central hypothesis to our work is that learning from 3D body shape data could lead to more value for UC than providing ground truth shape descriptors. While many approaches could be undertaken to achieve 3D-aware morphology embeddings, here we propose to learn them from 3D scans of robot bodies are a representative example of such methods.

We require a 3D dataset of robot body shapes and body plans, but there is no existing standard for this beyond the URDF files that underpin robotic simulators. It is important to note that in existing UC methods the controller receives one context token per limb, and so each element of the dataset needs to describe one limb in relation to the rest of the robot. Consequently, we must design the data such that it allows us produce the limb-specific context k_r^i. Furthermore, the data should be sparse and/or compact enough to admit low-dimensional latent vectors suitable for UC learning, yet still remain general enough to express arbitrary robot geometries. Occupancy grids (voxel grids) (Reijgwart et al., 2026) are a natural choice of representation: they can identify different limbs through the use of different cell values, are sparse and thus highly compressible by neural networks, and they expose their representational fidelity as a grid-size hyperparameter ν. Furthermore, voxel grids encode neighbourhood information through grid adjacency, enabling them to capture the area of 3D space where the robot's limbs connect to the rest of the body, i.e. the robot's joints. We collect a dataset of voxel grids $D = \{\langle r, \{v_r^i\}_{i=1}^{N_m}\rangle\}_{r \in R}$. Each voxel grid's values are null everywhere save

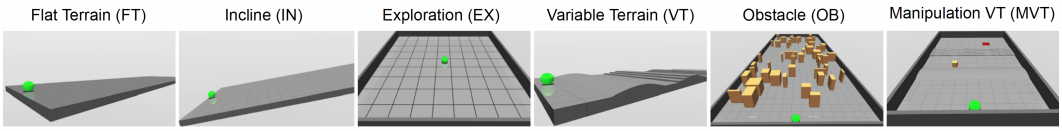


Figure 3: **Training Environments:** We use six environments of variable complexity and difficulty proposed by Gupta et al. (2021). The green ball represents the starting position. Image credits to Gupta et al. (2021). "ModularRL" (Huang et al., 2020) results shown in Appendix B.3 and Figure 17.

for where the robot's shape lies, where we set three distinct class values: the limb of interest i, the rest of the robot body, and the limb-body joint. This 3D representation, shown in Figure 1, captures much of the ground-truth kinematic parameters: limb shape, size, orientation, position, joints, etc.

The remaining piece of this representation design problem is the process by which we produce the body shape embeddings. We propose a volumetric morphology encoder (VME), which is capable of compressing body shape voxel grids v_r^i into learned vectors. Given a robot dataset R, a VME compresses each $v_r^i \ \forall i \in N_m \ \forall k \in R$ into a single shared latent space L. We use an autoencoder training regime composed of the encoder function $\text{VME}_\phi : \{0,1\}^{\nu^3 \times 4} \to \mathbb{R}^L$, which takes a one-hot encoded v_r^i as input, and the decoder function $D_\phi : \mathbb{R}^L \to \{0,1\}^{\nu^3 \times 4}$. The autoencoder is trained to minimize a class-weighted cross-entropy classification loss, where w_c are weights and c are classes and σ is the softmax function:

$$\mathcal{L}_{\text{rec}}(\phi) = -\mathbb{E}_{v_r^i \sim \mathcal{D}}\left[\sum_{x,y,z=1}^{\nu}\sum_{c=1}^{3} w_c, v_{r;xyzc}^i, \log\Big(\sigma(D_\phi(\text{VME}_\phi(v_r^i)))_{xyzc}\Big)\right] \tag{3}$$

The voxel classification problem that we propose avoids the limiting assumptions of some prior work (Trabucco et al., 2022), which assigned a distinct integer class label to **each** semantic limb type (e.g., a humanoid forearm or a cheetah thigh). This reliance on predefined labels restricted transferability to *ablations* of seen morphological graphs, as policies could not generalize to entirely novel robots featuring previously unseen integer labels. Instead, by operating over volumetric data, we apply the **same** class value to **all** limbs, allowing its physical *shape* to serve as its identifying feature. Furthermore, because our learned k_r representations utilize 3D input data, our approach elegantly accommodates both unseen variations of seen training morphologies as well as entirely unseen robots from unseen morphological graphs (to obtain their k_r^i representation, one must simply encode them as v_r^i and pass them through the VME model). Ultimately, the physical *shape* of the robot bodies dictates their body-shape context vectors (a subset of k_r), bypassing the need for rigid semantic labels.

6 Experiments

To evaluate our approach, we require specialized benchmarks where improving morphological context representations prior to UC training leads to tangible downstream performance benefits. We require a simulation with a large quantity of different morphological graphs M and robots R, as well as an unseen test set. Not many benchmarks of this type exist. We opt to leverage the challenging UNIMAL dataset of procedurally generated robots introduced by Gupta et al. (2021), and subsequently used by state-of-the-art UC methods (Gupta et al., 2022; Xiong et al., 2023; 2024). This dataset offers R_{train} (100 different morphological graphs, one concrete robot each), R_{unseen} (≈ 1150 unseen variant robots of the training graphs), and R_{test} (≈ 1250 robots for 100 unseen morphological graphs). We illustrate six tasks in Figure 3. (1) **Flat Terrain (FT):** The reward is the robot's forward (x-axis) velocity on a flat floor; (2) **Incline (IN):** The same reward as FT, but the terrain is a more challenging inclined plane; (3) **Exploration (EX):** The reward is proportional to the number of novel grid cells visited in a square arena; (4) **Variable Terrain (VT):** The same reward as FT, but each episode features a randomly sampled sequence of terrain types (e.g., stairs, bumps); (5) **Obstacles (OB):** The same reward as FT, but with a large number of obstacles randomly generated at every environment reset; and (6) **Manipulation in Variable Terrain (MVT):** A particularly challenging task originally proposed in Gupta et al. (2021) but omitted in subsequent work where the reward consists of an xy-plane displacement term that encourages the robot to approach an object, and an additional xy-plane displacement term that rewards moving the object toward a goal location. For more details, see Appendix A. We also provide results for ModularRL (Huang et al., 2020) in Appendix B.3. We now make precise the modification of the originally unbounded reward signal that we used in Section 4:

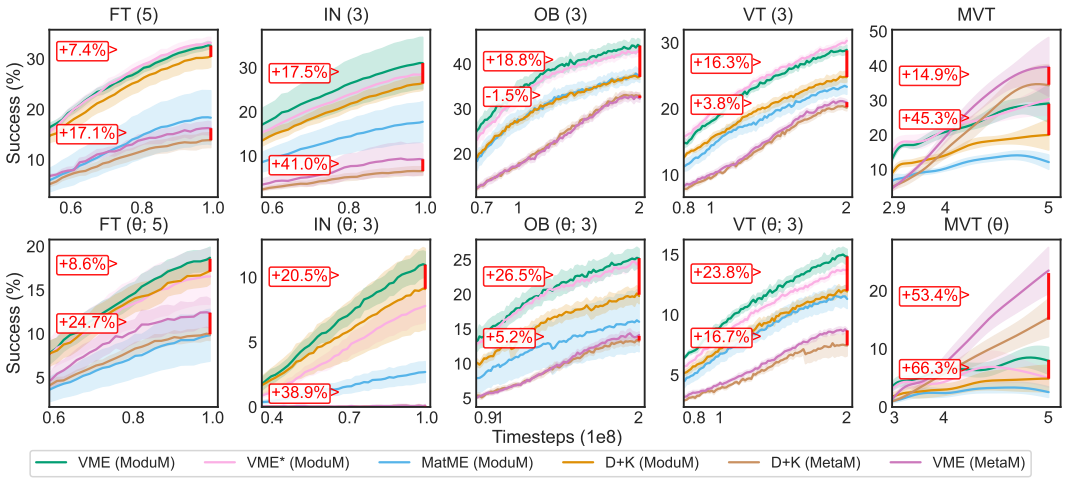


Figure 4: **UNIMAL Benchmark (Hard):** VME's context vectors improve or match performance in all tasks (or do not cause notable performance decreases). This is achieved by providing better body shape/connectivity context, a part of the kinematics context k_r^i, with absolutely no change to the base architecture. We observe that VME provide higher improvements in harder tasks (roughly left to right). The red labels and lines correspond to the improvement delta between VME and the unaltered baselines ModuMorph and MetaMorph (Xiong et al., 2023; Gupta et al., 2022).

$$r(s_t, a_t) = \begin{cases} s_t(v_x) & \text{if } s_t(v_x) < \xi \\ 10\xi & \text{otherwise} \end{cases} \tag{4}$$

where $s_t(v_x)$ is current state's x-axis velocity and ξ is the target velocity (note that Gupta et al. (2021) has unbounded ξ). Where applicable, we use a high target velocity (chosen by computing the top 25% of velocities in the unaltered tasks), which we report in parenthesis beside the task abbreviations. We then introduce another compositional level of difficulty in the from of random initial rotations around the vertical axis, which we mark by θ in the task names. Results for unaltered tasks are in Appendix B.

Baselines: We train policies using D+K context for both ModuMorph and MetaMorph. This corresponds exactly to the algorithms respectively proposed by Xiong et al. (2023) and Gupta et al. (2022). **Ablations:** We denote our proposed learned representation as VME. Unlike prior works in UC where k_r^i contains only limb-specific features, VME also encodes "rest of the body" and local "joint" information (see Figure 1). To account for this, we introduce VME$\star$, where each v_r^i only encodes the shape of limb i. This still communicates the same information as VME, given that the policy can aggregate information across limbs. **Finally,** to strenghten our claim that learning representations directly from 3D shapes is more effective than learning from ground-truth shape descriptor vectors, we introduce MatME. Here, we construct a 2D matrix where each row j corresponds to the ground-truth descriptor k_r^j for limb $j \in N_r$, sourced from the URDF. In other words, the row j of the MatME input matrix m_r^i correspond exactly to the body shape section of the kinematics vector k_r^j utilized in D+K. To obtain a distinct matrix m_k^i for each limb of interest i, we invert the sign of the i-th row. We modify the VME autoencoder architecture to use 1D limb-wise convolutions instead of 3D convolutions. Further architectural details are provided in Appendix A.3. **Importantly,** contrarily to the motivating experiment in Section 4, in Section 7 all methods (as well as D+K) share the same contextual dynamics vectors d_r and mostly the same kinematics vectors k_r: **we only alter the *body shape/connectivity* context** (a portion of k_r). Yet, we obtain very large differences in terms of final performance, supporting our claim that 3D body shape representations are a critical piece of the online universal control problem.

Training budget: We follow the established training budget for the UNIMAL tasks. FT, EX, and IN tasks: 100M interactions; VT, OB: 200M interactions. For the MVT task that we reintroduce, we find that a much larger budget of 500M interactions is necessary for the policy to have any traction over solving the task. We follow common practice in online universal control Gupta et al. (2022); Xiong et al. (2023; 2024) and we use PPO Schulman et al. (2017) to train the controllers. Contrary to prior work, we hold hyperparameters fixed across tasks. We train 5 random seeds for each method. More details are given in Appendix A.3 and Figure 12. **Training details:** For universal control training

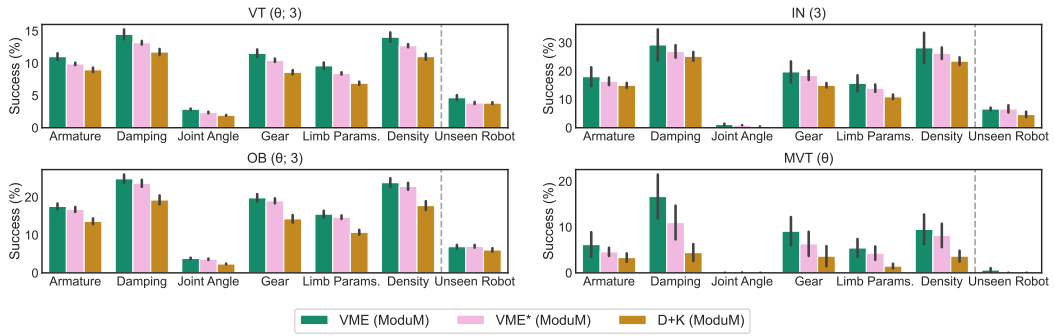


Figure 6: **Policy Transfer:** We additionally deploy trained controllers on both entirely unseen robots (R_{test}, right) and on domain-randomized Tobin et al. (2017) variants of training robots (R_{unseen}, left). In general, policies trained with VME transfer more reliably than other context vectors, but all methods struggle with R_{test}. Appendix B.4 shows results for the task variants that are not shown here.

we use a mixture of A100, L40s, and RTX8000 GPUs. On average, the easy tasks required 1 day of training, the harder tasks required 2 days, and the MVT task required 5 days.

7 Results

Figure 4 reports success as the fraction of timesteps the agent exceeds the target velocity, alongside manipulation success rates for the highly challenging MVT task. VME demonstrates a substantial improvement over the baseline D+K representations, except for a slight degradation with MetaMorph in OB (3). Furthermore, the extra information incorporated into VME proves highly effective: VME consistently matches or outperforms VME$\star$, with only minor underperformance in isolated cases. While VME$\star$ alone generally outperforms D+K, it does so less reliably than VME. Notably, the performance gap between VME and D+K widens when random initial rotations are introduced (second row of the figure). This reinforces our rationale for increasing the difficulty of the baseline UNIMAL tasks: as task complexity rises, morphological context becomes increasingly vital.

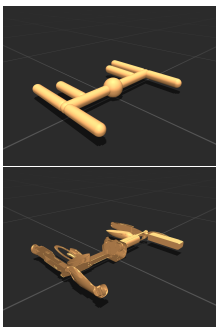


Figure 5: UNIMAL (top) and Mesh-UNIMAL (bottom).

In the MVT results (to which we had apply heaving smoothing due to the sparse nature of the success signal), we observe a substantial improvement over the strongest baseline (MetaMorph). Intuitively, VME's ability to ingest 3D limb shapes should enable the policy to better reason about interactions between the robot and the manipulable object, leading to more effective manipulation behaviour. This intuition is reflected both qualitatively and quantitatively. Indeed, policies trained with VME context tend to exhibit behaviours that seem to be more purposeful in their interactions with their world. Figure 7 shows how VME policies kick the MVT object to quickly move the object towards the goal. Appendix B.2.1 shows that VME policies quantitatively achieve more object air time than alternatives.

The policies in Figure 4 are trained on $|M| = |K| = 100$, but there are many more robots in the UNIMAL dataset. In Figure 6's left-hand side, we perform policy transfer to R_{unseen}, which contains unseen robots for seen morphological graphs (unseen robots for M_{train}). This R_{unseen} set offers both *kinematics* variants where the robot shapes are altered ("joint angle", "limb params") and *dynamics* variants where dynamic parameters are altered (all other). In Figure 6's right-hand side, we show

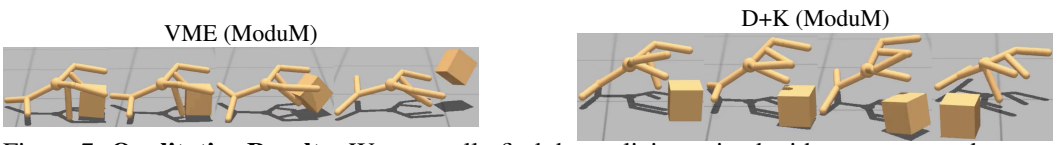


Figure 7: **Qualitative Results:** We generally find that policies trained with VME seem to be more "aware" of their bodies. They exhibit emergent behaviours such as the ability to "kick" the object in the MVT task. The standard ModuMorph policy seems clumsier, mistakenly moving the box out of the way. For videos see https://anonymizedprofile.github.io/vme/.

results for transfers to wholly unseen robots ($|M| = 100$ test set, $|R_{\text{test}}| \approx 1250$). In all cases, VME and VME$\star$ improves or matches the baseline D+K (which corresponds exactly to ModuMorph). In particular, even if VME only alters the body-shape portion of the kinematics portion of the overall context vector, we still observe stronger transfer capabilities to *dynamics* variants of R_{unseen}. While the exploration-exploitation dynamics of online RL are inherently difficult to interpret, one plausible explanation for this phenomenon is that the VME tokens lessen the burden of learning "what body am I controlling?" for the policies, allowing policies to more effectively learn other aspects of the task.

The UNIMAL dataset was selected for its great diversity of morphological graphs, unparalleled by any other alternative. This, however, limits our experiments to robots composed of primitive shapes. Yet, one of the core features of VME is its ability to ingest and describe any input shape, including arbitrary meshes (governed by the voxel grid size hyperparameter). To showcase this feature of our proposed method, we have constructed a synthetic version of UNIMAL, where for each limb, for each robot, we sample a random mesh from Mujoco Menagerie (Zakka et al., 2022), and we (greedily) resize, scale, and shear the mesh to fit the limb. More details are given in Appendix A.1. Here, the kinematics context available to be read from the definition files is much more limited: elements such as the connectivity and link axes are available, but the ground-truth body shape descriptors are not. To account for this, in addition to VME (ModuM) and D (ModuM), we train D+Capsule (ModuM). This is to provide a fair comparison to a strategy commonly used in RL for locomotion tasks, wherein capsule approximations are used to represent meshes. In other words, the context used in D+Capsule (ModuM) on Mesh-UNIMAL corresponds exactly to the D+K (ModuM) context on UNIMAL. Due to the very high computational cost of training controllers on Mesh-UNIMAL (we found simulation to be much costlier than UNIMAL), we only report ModuM on the three harder tasks from UNIMAL: OB, VT, and MVT. The results are shown in Figure 8.

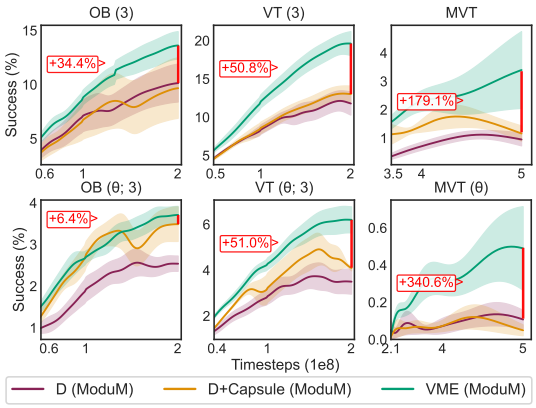


Figure 8: **Mesh-UNIMAL Benchmark (Harder):** We observe that the tasks based on Mesh-UNIMAL lead to much more unstable training curves, and a generalized loss of performance compared to equivalent UNIMAL tasks. Here, VME seems to enable much smoother training here, in addition to the massively improved performance we observed on other task domains. A Mesh-UNIMAL robot is shown in Figure 5.

8 Conclusion & Future Work

Contributions. In Section 4, we identify kinematic context as a critical driver of high-performance universal control. We then hypothesize that a large component of the kinematic context (the body shape) can be better captured using its native 3D domain rather than 1D descriptors. To capitalize on this insight, in Section 5 we introduce VME, a principled design to represent this vital portion of the morphological context vector. By varying only the portion of the kinematic context that encodes body shapes, we obtain widely different final online universal control performance. Empirically, our plug-and-play representation leads to often-substantial performance improvements, with higher gains in harder tasks that require precise reasoning about contact and embodiment. **Limitations.** Our work finds great performance gains by improving but a small part of the kinematics part of the full context vector. We believe that there is significant value in future work tackling the problem of suitably representing the rest of the context token. Furthermore, due to the limited availability of suitable simulable robot datasets with many different robot morphologies M, we are limited to the challenging but unrealistic UNIMAL setting. **Future work.** While *online* universal control promises to accelerate the development of embodied AI, real-world deployments of online UC policies trained on large M remain extremely rare. Successful cross-embodiment deployments have instead come from the *offline* universal control setting, which operates under very different assumptions. In particular, training a full-scale VLA with VME was impossible for us to do: large-scale datasets like Open-X Embodiment (O'Neill et al., 2024) only partially include the URDF information necessary to train a VME. While much neural architecture and data collection work would be required to enable VME on VLAs, if VME could be applied to this offline setting, we could potentially be looking at substantial generalization gains for the low cost of a one-time VME pretraining phase.

AI USE STATEMENT

In this work, we used generative AI tools (coding agents) to write SLURM launchers and monitor SLURM jobs. We also used coding agents to extend our hand-coded VME implementation and (4) from UNIMAL to the ModularRL domain, and for help with data aggregation to create our plots. We also used it for help with making the Mesh-UNIMAL dataset. In the manuscript, LLMs were used to augment writing and/or fix typos. We have reviewed all AI-assisted work (we ensured that the ModularRL extensions were valid). We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

There is no particular ethics concern related to our proposed method.

REPRODUCIBILITY STATEMENT

Our experiments are based on several open-source codebases: ModuMorph (Xiong et al., 2023) packages MetaMorph (Gupta et al., 2022), UNIMAL (Gupta et al., 2021), and ModularRL (Huang et al., 2020). We also use Mujoco Menagerie (Zakka et al., 2022) to source limb meshes for Mesh-UNIMAL. As for the VME body-part scanning process, the VME implementation, and the extension of ModuMorph and MetaMorph to consume it: we shall make our code open-source upon acceptance into ICLR. We will also open-source the Mesh-UNIMAL dataset to help drive research into 3D morphology representations.

REFERENCES

Bo Ai, Liu Dai, Nico Bohlinger, Dichen Li, Tongzhou Mu, Zhanxin Wu, K. Fay, Henrik I Christensen, Jan Peters, and Hao Su. Towards embodiment scaling laws in robot locomotion. In 1st Workshop on Robot Hardware-Aware Intelligence, 2025. URL https://openreview.net/forum?id=9LMG2hQ3jq.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279, jun 2013.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164, 2024.

Nico Bohlinger, Grzegorz Czechmanowski, Maciej Piotr 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. In 8th Annual Conference on Robot Learning, 2024. URL https://openreview.net/forum?id=PbQOZntuXO.

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.

Hiroki Furuta, Yusuke Iwasawa, Yutaka Matsuo, and Shixiang Shane Gu. A system for morphology-task generalization via unified representation and behavior distillation. In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=HcUf-QwZeFh.

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

Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. Metamorph: Learning universal controllers with transformers. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=Opmqtk_GvYL.

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

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes. arXiv preprint arXiv:1502.02259, 2015.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. CoRR, abs/1604.00772, 2016. URL http://arxiv.org/abs/1604.00772.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Jiaheng Hu, Julian Whitman, and Howie Choset. GLSO: Grammar-guided latent space optimization for sample-efficient robot design automation. In 6th Annual Conference on Robot Learning, 2022. URL https://openreview.net/forum?id=vyR0C7zvozS.

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, 2020. URL https://api.semanticscholar.org/CorpusID:220424751.

Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson. My body is a cage: the role of morphology in graph-based incompatible control. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=N3zUDGN5lO.

Yann LeCun, Koray Kavukcuoglu, and Clement Farabet. Convolutional networks and applications in vision. In Proceedings of 2010 IEEE International Symposium on Circuits and Systems, pp. 253–256, 2010. doi: 10.1109/ISCAS.2010.5537907.

Muhan Li, Lingji Kong, and Sam Kriegman. Generating freeform endoskeletal robots. In The Thirteenth International Conference on Learning Representations, 2025. URL https://openreview.net/forum?id=awvJBtB2op.

Melissa Mozian, Juan Camilo Gamboa Higuera, David Meger, and Gregory Dudek. Learning domain randomization distributions for training robust locomotion policies. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6112–6117. IEEE, 2020.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 6892–6903. IEEE, 2024.

Austin Patel and Shuran Song. Get-zero: Graph embodiment transformer for zero-shot embodiment generalization. In 2025 IEEE International Conference on Robotics and Automation (ICRA), pp. 14262–14269. IEEE, 2025.

Victor Reijgwart, Jens Behley, Teresa Vidal-Calleja, Helen Oleynikova, Lionel Ott, Cyrill Stachniss, and Ayoung Kim. Dense map representation. In Luca Carlone, Ayoung Kim, Timothy Barfoot, Daniel Cremers, and Frank Dellaert (eds.), SLAM Handbook. From Localization and Mapping to Spatial Intelligence. Cambridge University Press, 2026.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. CoRR, abs/2109.11978, 2021. URL https://arxiv.org/abs/2109.11978.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. CoRR, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

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.

L. Shao, F. Ferreira, M. Jorda, V. Nambiar, J. Luo, E. Solowjow, J. A. Ojea, O. Khatib, and J. Bohg. Unigrasp: Learning a unified model to grasp with multifingered robotic hands. IEEE Robotics and Automation Letters, 5(2):2286–2293, April 2020. ISSN 2377-3774. doi: 10.1109/LRA.2020.2969946.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, 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, 2024. URL https://arxiv.org/abs/2405.12213.

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), pp. 23–30, 2017.

Brandon Trabucco, Mariano Phielipp, and Glen Berseth. AnyMorph: Learning transferable polices by inferring agent morphology. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), Proceedings of the 39th International Conference on Machine Learning, volume 162 of Proceedings of Machine Learning Research, pp. 21677–21691. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/trabucco22b.html.

Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE international conference on computer vision, pp. 4489–4497, 2015.

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.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In International Conference on Learning Representations, 2018. URL https://api.semanticscholar.org/CorpusID:65051725.

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

Zheng Xiong, Risto Vuorio, Jacob Beck, Matthieu Zimmer, Kun Shao, and Shimon Whiteson. Distilling morphology-conditioned hypernetworks for efficient universal morphology control. arXiv preprint arXiv:2402.06570, 2024.

Zhenjia Xu, Beichun Qi, Shubham Agrawal, and Shuran Song. Adagrasp: Learning an adaptive gripper-aware grasping policy. In Proceedings of the IEEE International Conference on Robotics and Automation, 2021.

Kevin Zakka, Yuval Tassa, and MuJoCo Menagerie Contributors. MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo, 2022. URL http://github.com/google-deepmind/mujoco_menagerie.

APPENDIX

A IMPLEMENTATION DETAILS

A.1 TASKS DETAILS:

In all tasks, an episode terminates when the robot's torso height falls below a predefined threshold; however, no negative terminal reward is assigned, following common practice to avoid inducing overly conservative behaviors (Mozian et al., 2020; Rudin et al., 2021). The first three tasks (FT, IN, EX) rely solely on proprioceptive observations, as the terrain is locally invariant and does not change across environment resets. In contrast, for VT, OB, and MVT, the terrain varies both locally and across resets, making proprioceptive information alone insufficient to infer the surrounding geometry. Consequently, these tasks require exteroceptive observations, and we provide the agent with a height map of its surroundings. For MVT, the observation space is further augmented with the object's position and velocity relative to the robot, its rotational velocity, and the distance between the object and the goal.

To show how VME impacts learning on robots composed of non-primitive shapes, we design Mesh-UNIMAL. In Mesh-UNIMAL, for each UNIMAL robot, for each limb, we randomly fit a mesh sampled from Mujoco Menagerie (Zakka et al., 2022). This allows us to evaluate online universal control on a dataset of robots with very diverse limbs. An example is shown in Figure 5.

A.2 DATA COLLECTION DETAILS:

While constructing the voxel grids, we normalize the robots' scale with respect to the robot with the largest bounding box. We also apply an aggressive data augmentation strategy based on random 3D rotations. We disable this augmentation when producing the final latents that we save to disk for use by downstream universal controllers.

A.3 VME DETAILS:

We use an autoencoder based on a 3D CNN (Tran et al., 2015) with residual blocks inspired by the ResNet (He et al., 2016) architecture. The autoencoder consists of three main components: an input embedding, an encoder, and a decoder.

The *input embedding* applies a per-voxel linear projection from the single input channel to 64 feature channels. Given an input voxel grid of shape $(1, 32, 32, 32)$, this produces a feature volume of shape $(64, 32, 32, 32)$.

The *encoder* compresses the feature volume using 3D convolutions and residual blocks. For grid resolution $\nu = 32$, the spatial resolution is reduced as $32 \to 16 \to 8 \to 4 \to 2 \to 1$, while the channel dimension increases from 64 to 128 and then 256. Residual blocks consist of two $3 \times 3 \times 3$ convolutions with batch normalization and LeakyReLU activations, with a projection shortcut when the stride or number of channels changes. The final projection maps the $(256, 2, 2, 2)$ representation to a latent tensor of shape $(L, 1, 1, 1)$ using a convolution with kernel size 2. No activation is applied at the latent bottleneck.

The *decoder* upsamples the latent tensor using transpose convolutions, increasing spatial resolution as $1 \to 2 \to 4 \to 8 \to 16 \to 32$. The channel dimensions decrease as $L \to 256 \to 128 \to 64 \to 32 \to 32$. The first four upsampling stages are followed by residual blocks; the final upsampling stage is followed directly by a $3 \times 3 \times 3$ convolution producing C output logits, with no final activation.

A.4 MATME DETAILS:

The MatME model learns over matrices in which each row represents a robot component token b_r^i, following the morphology representations used in ModuMorph (Xiong et al., 2023) and MetaMorph (Gupta et al., 2022). For each training example, the row corresponding to the component of interest is multiplied by -1 and placed first, identifying the target component. Similar to the VME model, MatME uses a 1D convolutional autoencoder with residual blocks.

The *input embedding* applies a linear transformation across the component dimension, projecting the N component rows to 64 feature channels. The resulting representation has shape $(64, F)$, where F is the per-component feature dimension.

The *encoder* compresses the feature sequence using strided 1D convolutions and residual blocks. After an initial convolution, a strided convolution reduces the sequence length as $F \to F/2$. Four residual blocks then process the sequence, with two of them performing downsampling: $F/2 \to F/4 \to F/8$. A final strided convolution reduces the sequence to $F/16$, followed by a 1×1 convolution that maps to the latent dimension L. Each residual block contains two 3×1 convolutions with batch normalization and LeakyReLU activations. For the default feature dimension $F = 16$, the latent representation has shape $(L, 1)$. No activation is applied at the latent bottleneck.

The *decoder* upsamples the latent representation using transpose convolutions, increasing the sequence length from $F/16$ back to F. The first two upsampling stages are followed by residual blocks. A final 3×1 convolution projects back to the original number of components, and a linear projection reconstructs the original feature dimension. No final activation is applied.

A.5 Universal Control Details:

In our universal control experiments, the proprioceptive robot states s_r^i are composed of the values of the following Mujoco elements: 'body_xpos', 'body_xvelp', 'body_xvelr', 'body_xquat', 'qpos', 'qvel'. The context token for the ablation or baseline at hand is also appended at the end of the proprioceptive state. The `D+K` context tokens c_r^i are composed of the values of the following Mujoco elements: 'body_pos', 'body_ipos', 'body_iquat', 'geom_quat', 'body_mass', 'body_shape', 'body_parent', 'jnt_pos', 'joint_range', 'joint_axis', 'gear'.

Contrarily to (Xiong et al., 2023; Gupta et al., 2022), we do not search the hyperparameter (HP) space for each task. Instead, we use the same hyperparameters across all tasks; these correspond to the default HPs from the ModuMorph code release, as listed on their GitHub. Proprioceptive observations are first partitioned into per-limb features. No changes were made to the ModuMorph and MetaMorph models: we only modify their input context vectors.

A.5.1 Transformer Policies For Universal Control: MetaMorph and ModuMorph

Our method is agnostic to the choice of universal controller, but in our experiments we leverage ModuMorph Xiong et al. (2023) and MetaMorph Gupta et al. (2022), two recent methods that have shown impressive results on online universal control tasks with low-level high-frequency control. In the MetaMorph architecture, the state information of a node i is augmented by the node's context such that $\hat{s}_k^i = [s_k^i, c_k^i]$. A shared embedding layer then transforms $\hat{s}_k^i$ into an embedding e_k^i. A transformer encoder then creates node features f_k^i from the node embeddings using key $k_k^i = W_k e^i$, query $q_k^i = W_q e^i$ and value $v_k^i = W_v e^i$, where $W_{\{k,q,v\}}$ are learned. In addition to the node-level proprioceptive features f_k^i that result from $\hat{s}_k^i$, in some tasks, it might also be necessary to have access to exteroceptive information such as a height map of challenging terrain. In this case, a learned MLP embeds the exteroception into a vector, which is concatenated to each f_k^i. A transformer decoder then decodes actions $a^i \in A_k^i$ from each f_k^i.

ModuMorph improves upon MetaMorph by improving the policy's ability to leverage contextual information. MetaMorph enforces that all nodes share network parameters, while in ModuMorph, a hypernetwork (HN) Ha et al. (2016) layer is introduced to allow the context c_k^i to alter the embedding function for node $i \in N_r$. In ModuMorph the weights W and biases b are derived from the context of the node c_k^i such that $e_k^i = \texttt{HN}_{\phi,W}(c_k^i)\hat{s}_k^i + \texttt{HN}_{\phi,b}(c_k^i)$, where ϕ is the weights of the HN (shared across all nodes and robots). Furthermore, ModuMorph imposes an inductive bias upon the policy in the form of *fixed attention*: each node pays static attention to the others, where these attentions are learned from the morphology context $\{c_k^i\}_{i=1}^{N_r}$ alone. In this work, we leverage the ModuMorph universal control *architecture*, however we modify the way in which the morphology context is *encoded* (see Figure 1 for details).

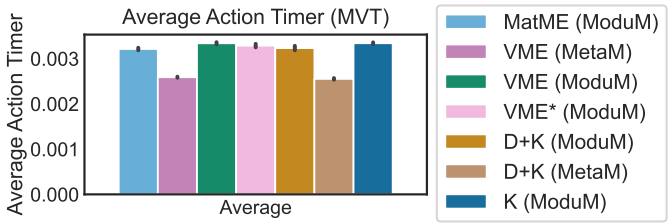


Figure 9: **Time needed to generate an action** (averaged over batch size and all training timesteps). We notice that all methods take approximately 0.003 seconds to generate an action, with only small differences. The timer was set up to capture the inference time for the exact line in the code where the model was called.

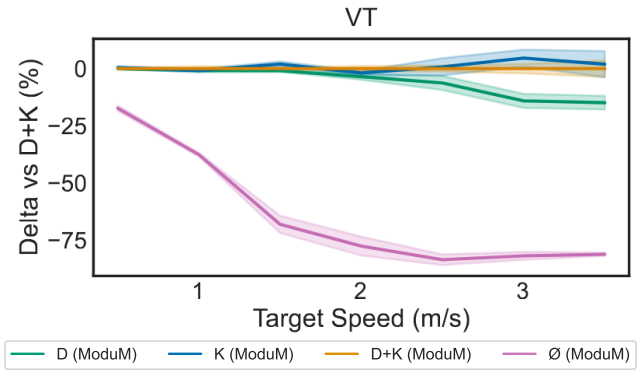


Figure 10: **Importance of Body Shape Context**: Companion figure for Figure 2. In the easiest regimes, ∅ context is still capable of reaching some level of success. However, its performance quickly drops in the harder regimes, before plateauing at a $\sim 80\%$ performance decrease compared to the D+K baseline. Even if our results suggest that K context is more important than D context, D context alone still enables the controller to reach a much higher level of success than ∅.

A.6 Computational Cost:

To train a VME, a suitable dataset must first be collected. Using a single-threaded program, ~ 30 hours were necessary to process all robots in the UNIMAL dataset. Then, training one VME model required ~ 5 days on a L40S Nvidia GPU. After training, the final learned context vectors are cached to disk for later use. We note that it would not be desirable to recompute the latents at each step to be reflective of the robot's limb's current orientations and positions due to the high computational cost of scanning the robots. Because the VME models are never active during the universal control phase, no additional computational cost is incurred in the universal control phase. Training ModuMorph and MetaMorph took ~ 20 hours per 100M timesteps, e.g. the OB task required around 40 hours of training per individual method and seed.

The cached VME context can reused many times for training new universal controllers. In essence, assuming that the VME model was trained on a suitably large robot dataset, it can be seen as a foundation model for body plan representation, and treated as a pretrained backbone.

In Figure 9, we show that all methods require approximately the same time to generate actions. Any differences can be attributed to random noise or hardware differences.

B Supplementary Results

B.1 Motivating Experiment

In Table 1, we observe that K context contributes more to universal control performance than the dynamics context. As the task gets harder, the performance of dynamics (D) context fails to track D+K context, while K doesn't exhibit large performance deltas. The seemingly positive performance delta exhibited by K over D+K can be interpreted as expected random noise over the training process.

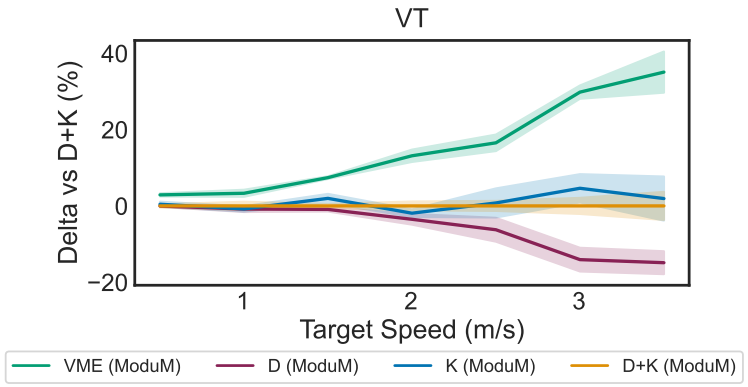


Figure 11: Accompanying figure for Figure 2. VME outperforms other context vectors at all target velocities.

In Figure 10, we report results for ∅ context. In this context vector, no useful contextual information is provided, and so the ∅ universal controller's poor performance reflects the critical value of proper contextual conditioning for universal control. In Figure 11, we show how VME performs extremely well on this sweep of the target velocity.

Table 1: Accompanying table for Figure 10. Performance improvement (%) over ModuMorph (D+K).

Task	Type	**D+K (ModuM)**	**D (ModuM)**	**K (ModuM)**	**∅ (ModuM)**
VT	(0.5) (%)	+0.00%	-0.07%	**+0.28%**	-17.3%
	(1.0) (%)	**+0.00%**	-0.59%	-0.95%	-37.6%
	(1.5) (%)	+0.00%	-0.09%	**+2.48%**	-68.2%
	(2.0) (%)	**+0.00%**	-3.51%	-1.76%	-77.5%
	(2.5) (%)	+0.00%	-6.85%	**+0.33%**	-83.6%
	(3.0) (%)	+0.00%	-14.3%	**+2.90%**	-81.9%
	(3.5) (%)	+0.00%	-10.9%	**+2.34%**	-81.1%

B.2 VME Training

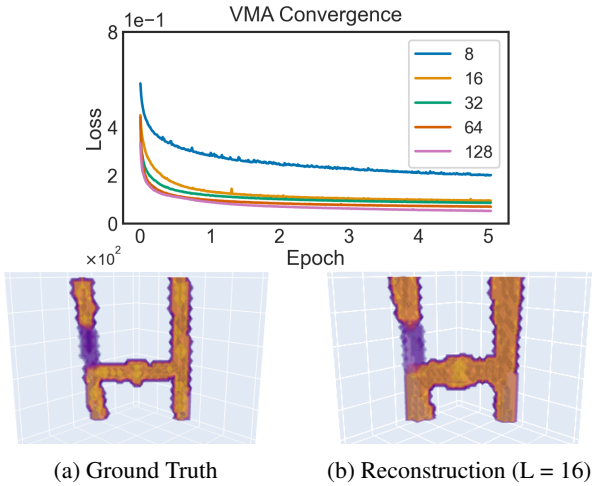
To determine a suitable dimensionality L for the VME's shared latent space, we report VME convergence profiles for a suite of L values in Figure 12. We observe that an overly constricted latent space ($L = 8$) results in a significant information bottleneck, preventing the model from effectively capturing the morphological details of the robot modules. Conversely, increasing L beyond 16 yields diminishing returns, offering negligible improvements. Consequently, we employ $L = 16$ for all of our universal control experiments, as it strikes the most effective balance between representation compactness and fidelity.

In Figure 13, we show how the weight term in Eq. (3) impacts reconstruction accuracy.

B.2.1 MVT

We now offer supplementary results for the MVT task. In Figure 15, we show how VME-trained policies seem to be better at reasoning about robot-object contacts, allowing policies to kick the MVT object more reliably.

The average episodic return for the MVT task shown in Figure 16 is misleading: with very small percentage differences in terms of returns, one might expect very little changes between the behaviours of policies trained with the different context vectors. Yet, the small $\approx \%$ improvement in returns for VME (MetaM) policies actually results in a massive in terms of success rate, as shown in Figure 4.



(a) Ground Truth (b) Reconstruction (L = 16)

Figure 12: Averaged VME training loss (Equation (3)) over the course of training. We ablate over the latent size and find that $L > 16$ only offers diminishing returns. We thus select $L = 16$ for use in our universal control experiments. Regarding the robot volumetric plots: the left column shows shows the ground truth module-segmented robot volume while the right column shows its VME reconstruction. We observe that the reconstruction shows grosser features, but that most details were successfully captured.

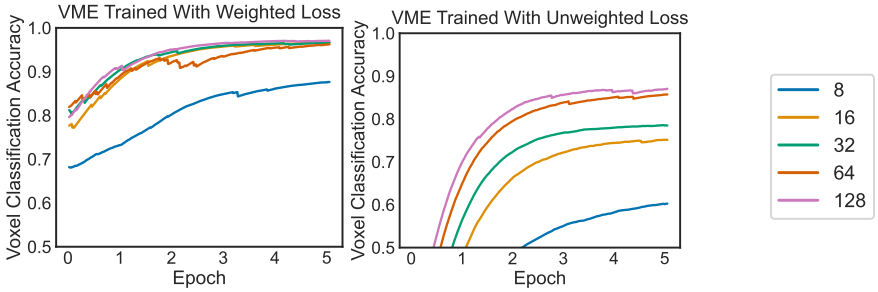


Figure 13: Average VME reconstruction accuracy (which corresponds to a voxel-level classification accuracy) during training. We only report the accuracy for voxels of non-zero class label; otherwise, the very large number of pixels (with $\nu = 32$, this is 32768 voxels) inflates the accuracy even when the non-zero pixels are wrong.

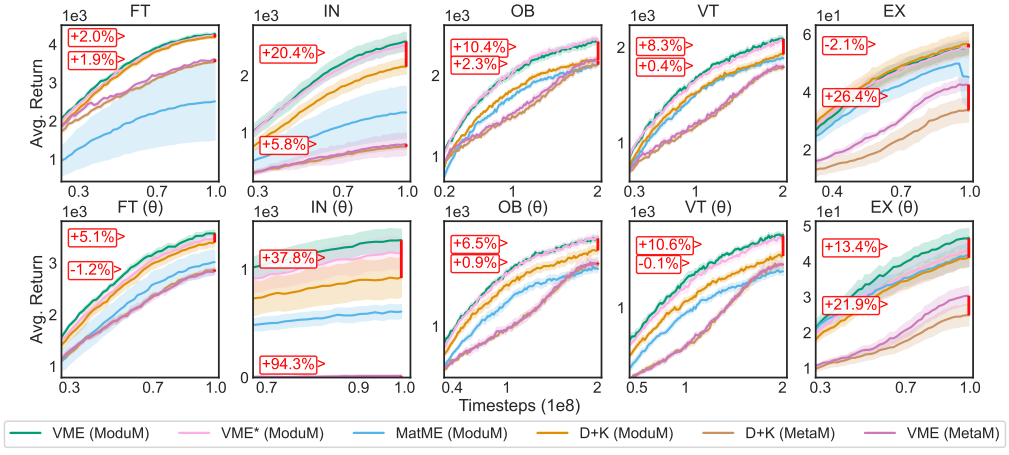


Figure 14: **UNIMAL Benchmark (Easier):** VME context outperforms ModuMorph (Xiong et al., 2023) as well as MetaMorph Gupta et al. (2022) in most tasks. The red labels and lines correspond to the improvement delta between VME and the unaltered baselines ModuMorph and MetaMorph (Xiong et al., 2023; Gupta et al., 2022).

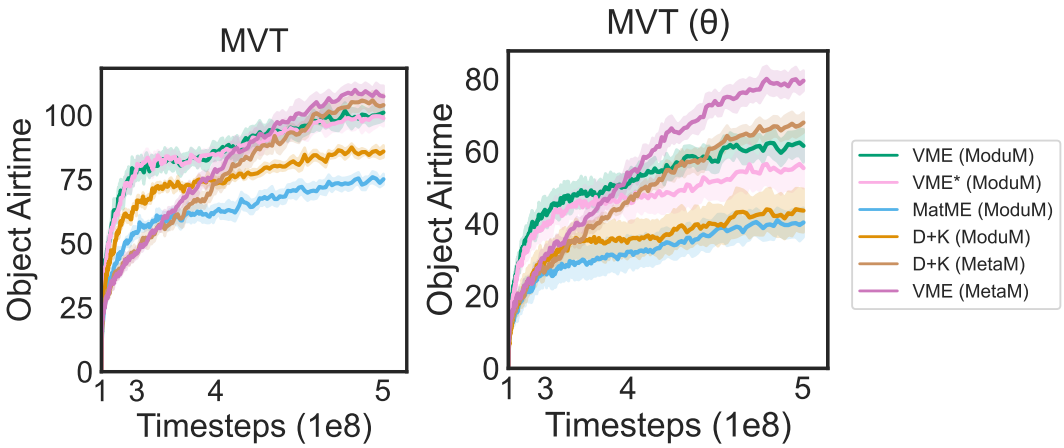


Figure 15: **Average Episodic Airtime For MVT:** We find that more successful policies (see Figure 4 and Figure 7) tend to have higher airtime (in timesteps) for their manipulable object; i.e., that "kicking" the object forward is a great strategy.

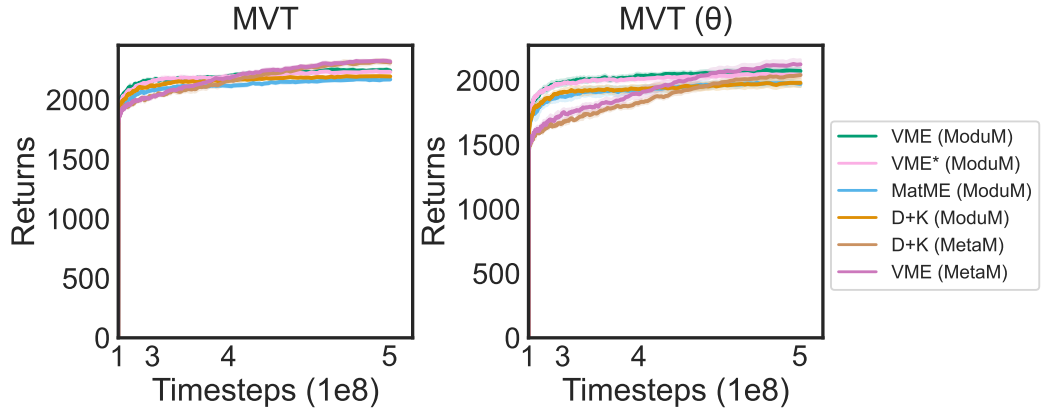


Figure 16: **Average Episodic Returns For MVT:** Accompanying figure for Figure 4.

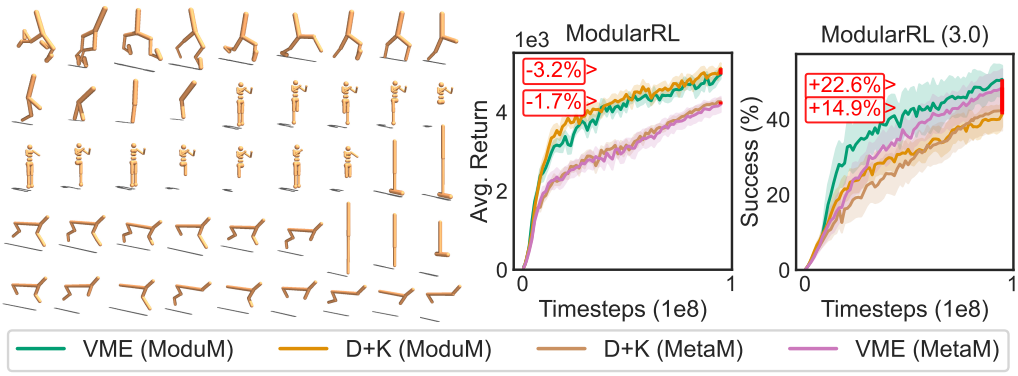


Figure 17: **Left:** a sample of the ModularRL robots Huang et al. (2020). **Middle:** the unbounded task (no Equation (4)). **Right:** the task with a target velocity of 3.0 m/s. Here, we again find that using a more difficult reward signal (Equation (4)) leads to a more pronounced stratification of different context vectors. We find that VME provides substantial performance improvements of up to $\approx 20\%$ in the harder regime.

B.3 ModularRL Environment

The ModularRL environment has historically been used extensively by the online universal control literature before UNIMAL's release Huang et al. (2020); Trabucco et al. (2022). In this environment, limbs are systematically added/removed from classical Mujoco robots, creating a dataset of multiple morphological graphs M. The task is to run forward as fast as possible. Here, we again apply the Equation (4) reward signal. While these robots are composed of primitive shapes, just like UNIMAL, we still find that VME is able to provide substantial performance improvements, as shown in Figure 17.

B.4 Evaluation & Generalization Results

In Table 2, we present the evaluation results for all tasks and variants. Bold cells represent the best method (as according to its average performance), while underlined and italicized cells represent methods whose standard error overlaps the best method's.

Table 2: **Evaluation Table:** Accompanying Table For Figure 6. MVT and tasks with target speeds are reported in terms of success rate, while other tasks are in terms of returns.

Task	Type	VME (MM)	VME* (MM)	D (MM)	D+K (MM)
EX	Armature	*44.3 ± 1.4*	*44.1 ± 2.6*	40.0 ± 2.0	**45.8 ± 3.0**
	Damping	*54.2 ± 2.0*	*54.7 ± 3.4*	49.1 ± 2.7	**56.3 ± 4.0**
	Density	*52.7 ± 2.0*	*52.9 ± 3.4*	45.6 ± 2.5	**53.7 ± 3.9**
	Gear	*43.9 ± 1.6*	**44.8 ± 2.8**	36.1 ± 2.4	*43.6 ± 3.5*
	Joint Angle	*16.9 ± 0.54*	**17.4 ± 0.84**	13.8 ± 0.84	*16.7 ± 1.3*
	Limb Params.	**38.5 ± 1.2**	*38.2 ± 2.3*	33.9 ± 1.9	35.1 ± 1.9
	Unseen Robot	*20.0 ± 0.84*	**21.7 ± 1.8**	*20.3 ± 1.3*	*21.5 ± 1.8*
EX (θ)	Armature	**38.3 ± 1.9**	*35.3 ± 2.9*	29.0 ± 1.5	34.0 ± 2.0
	Damping	**46.0 ± 2.7**	*42.5 ± 4.0*	34.8 ± 2.0	*40.9 ± 2.7*
	Density	**44.9 ± 2.7**	*41.4 ± 4.0*	32.9 ± 1.9	39.0 ± 2.5
	Gear	**38.3 ± 2.5**	*35.5 ± 3.2*	26.5 ± 1.6	32.4 ± 2.5
	Joint Angle	**17.4 ± 1.0**	15.2 ± 1.0	11.2 ± 0.74	14.1 ± 1.2
	Limb Params.	**34.2 ± 1.9**	*30.9 ± 2.7*	25.1 ± 1.5	26.9 ± 1.4
	Unseen Robot	*18.1 ± 1.9*	**18.9 ± 1.6**	15.6 ± 0.65	*17.8 ± 1.2*
FT	Armature	**3170 ± 31**	3109 ± 15	3052 ± 16	*3126 ± 32*
	Damping	**4160 ± 41**	*4125 ± 15*	4073 ± 25	*4108 ± 27*
	Density	**4078 ± 37**	*4064 ± 10*	3813 ± 32	4007 ± 27
	Gear	**3464 ± 36**	*3461 ± 17*	3045 ± 38	3275 ± 32
	Joint Angle	*1130 ± 38*	**1165 ± 38**	753 ± 20	940 ± 56
	Limb Params.	**2813 ± 29**	*2784 ± 28*	2747 ± 20	2486 ± 29
	Unseen Robot	**1313 ± 56**	*1304 ± 45*	1137 ± 26	*1309 ± 26*
FT (θ)	Armature	**2661 ± 42**	*2593 ± 50*	2350 ± 57	2526 ± 65
	Damping	**3472 ± 48**	*3370 ± 71*	3071 ± 72	3328 ± 82
	Density	**3418 ± 53**	*3313 ± 69*	2856 ± 66	3212 ± 86
	Gear	**2915 ± 60**	*2850 ± 63*	2312 ± 50	2652 ± 69
	Joint Angle	**961 ± 28**	894 ± 37	584 ± 25	790 ± 34
	Limb Params.	**2427 ± 47**	2299 ± 52	2041 ± 53	1982 ± 63
	Unseen Robot	**1224 ± 22**	1098 ± 50	905 ± 36	1065 ± 52
FT (5)	Armature	*17.8 ± 0.66*	**18.2 ± 0.66**	13.8 ± 0.86	*17.2 ± 1.3*
	Damping	*30.2 ± 0.73*	**31.0 ± 1.0**	23.8 ± 1.3	*28.8 ± 2.1*
	Density	*29.8 ± 0.86*	**30.2 ± 0.86**	21.4 ± 1.1	*27.8 ± 1.9*
	Gear	**21.2 ± 0.66**	*21.2 ± 0.58*	13.8 ± 0.66	18.5 ± 1.3
	Joint Angle	**1.00 ± 0.00**	*1.00 ± 0.00*	0.00 ± 0.00	*0.83 ± 0.17*
	Limb Params.	**16.4 ± 0.81**	*16.2 ± 0.49*	12.2 ± 0.49	13.5 ± 1.1
	Unseen Robot	**7.00 ± 0.55**	5.20 ± 0.58	4.40 ± 0.24	*6.33 ± 0.42*
FT (θ; 5)	Armature	**10.4 ± 0.75**	*9.60 ± 1.7*	*9.40 ± 0.81*	*10.0 ± 1.1*
	Damping	**16.8 ± 1.1**	*15.6 ± 3.0*	*15.6 ± 1.5*	*16.2 ± 1.5*
	Density	**16.6 ± 1.0**	*15.2 ± 2.9*	13.8 ± 1.3	*15.2 ± 1.5*
	Gear	**12.2 ± 1.1**	*10.8 ± 2.3*	9.80 ± 1.2	*11.4 ± 1.1*
	Joint Angle	**0.80 ± 0.20**	*0.80 ± 0.20*	0.00 ± 0.00	*0.40 ± 0.24*
	Limb Params.	**8.60 ± 0.75**	*8.20 ± 1.6*	*7.40 ± 0.81*	*7.40 ± 0.75*
	Unseen Robot	**3.80 ± 0.73**	*2.80 ± 0.86*	*3.20 ± 0.66*	*3.80 ± 0.66*
IN	Armature	**1746 ± 107**	*1656 ± 78*	1495 ± 115	1449 ± 102
	Damping	**2488 ± 144**	*2430 ± 104*	2153 ± 148	2102 ± 131
	Density	**2446 ± 149**	*2377 ± 98*	1997 ± 137	2007 ± 127
	Gear	**1957 ± 113**	*1878 ± 82*	1478 ± 105	1477 ± 91
	Joint Angle	**379 ± 38**	309 ± 15	153 ± 25	180 ± 27
	Limb Params.	**1591 ± 100**	*1459 ± 84*	1260 ± 98	1095 ± 82
	Unseen Robot	**591 ± 64**	*496 ± 32*	368 ± 51	461 ± 29
IN (θ)	Armature	**851 ± 77**	*793 ± 98*	600 ± 129	625 ± 125
	Damping	**1210 ± 100**	*1129 ± 133*	858 ± 178	907 ± 172
	Density	**1176 ± 101**	*1086 ± 129*	803 ± 166	862 ± 161

Continued on next page

Task	Type	VME (MM)	VME* (MM)	D (MM)	D+K (MM)
	Gear	**998 ± 80**	*911 ± 110*	652 ± 134	711 ± 130
	Joint Angle	**171 ± 20**	*158 ± 22*	72.8 ± 20	110 ± 32
	Limb Params.	**799 ± 68**	*711 ± 84*	534 ± 116	494 ± 95
	Unseen Robot	**357 ± 40**	*297 ± 67*	202 ± 60	*251 ± 70*
IN (3)	Armature	**18.0 ± 3.4**	*16.4 ± 1.3*	11.2 ± 0.73	*14.8 ± 0.97*
	Damping	**29.2 ± 5.3**	*26.8 ± 2.3*	19.0 ± 1.1	*25.2 ± 1.6*
	Density	**28.2 ± 5.3**	*26.4 ± 2.1*	17.0 ± 1.0	*23.6 ± 1.4*
	Gear	**19.6 ± 3.6**	*18.6 ± 1.6*	10.8 ± 0.58	15.2 ± 0.73
	Joint Angle	**1.20 ± 0.37**	*1.00 ± 0.00*	0.00 ± 0.00	0.40 ± 0.24
	Limb Params.	**15.6 ± 2.9**	*14.0 ± 1.3*	9.40 ± 0.40	10.6 ± 0.87
	Unseen Robot	**6.80 ± 0.49**	*6.60 ± 1.4*	4.60 ± 0.40	4.80 ± 0.97
IN (θ; 3)	Armature	**7.00 ± 0.63**	4.40 ± 0.68	4.40 ± 0.68	*5.40 ± 2.0*
	Damping	**10.8 ± 0.97**	7.40 ± 1.1	7.00 ± 0.84	*8.80 ± 3.0*
	Density	**10.0 ± 0.89**	6.80 ± 1.0	6.00 ± 0.84	*8.00 ± 2.7*
	Gear	**7.80 ± 0.58**	5.00 ± 0.95	4.80 ± 0.49	*6.00 ± 2.0*
	Joint Angle	**0.60 ± 0.24**	0.00 ± 0.00	0.00 ± 0.00	*0.20 ± 0.20*
	Limb Params.	**6.20 ± 0.58**	4.00 ± 0.71	3.80 ± 0.49	3.80 ± 1.4
	Unseen Robot	**4.60 ± 0.98**	*4.00 ± 0.84*	2.80 ± 0.66	*3.80 ± 1.2*
VT	Armature	**1694 ± 22**	*1634 ± 41*	1436 ± 18	1584 ± 22
	Damping	**2063 ± 25**	*2014 ± 53*	1769 ± 18	1935 ± 26
	Density	**2024 ± 29**	*1973 ± 54*	1687 ± 17	1875 ± 21
	Gear	**1824 ± 29**	*1778 ± 54*	1416 ± 25	1611 ± 32
	Joint Angle	**722 ± 33**	*683 ± 20*	452 ± 12	580 ± 25
	Limb Params.	**1572 ± 27**	1495 ± 36	1266 ± 19	1324 ± 22
	Unseen Robot	**806 ± 41**	*771 ± 36*	629 ± 26	727 ± 25
VT (θ)	Armature	**1235 ± 15**	*1217 ± 23*	1034 ± 27	1137 ± 24
	Damping	**1493 ± 17**	*1473 ± 27*	1251 ± 32	1382 ± 29
	Density	**1456 ± 17**	*1433 ± 26*	1173 ± 29	1319 ± 25
	Gear	**1301 ± 10**	*1281 ± 17*	972 ± 32	1144 ± 21
	Joint Angle	**498 ± 16**	*475 ± 20*	328 ± 19	408 ± 16
	Limb Params.	**1124 ± 11**	1074 ± 22	863 ± 28	917 ± 24
	Unseen Robot	**615 ± 15**	576 ± 17	478 ± 23	*580 ± 21*
VT (3)	Armature	*20.2 ± 0.49*	**21.0 ± 0.45**	16.2 ± 0.73	17.6 ± 0.68
	Damping	28.0 ± 0.71	**29.2 ± 0.37**	22.2 ± 0.97	24.6 ± 0.98
	Density	27.4 ± 0.51	**28.8 ± 0.37**	20.4 ± 0.81	23.2 ± 1.0
	Gear	22.2 ± 0.49	**23.2 ± 0.37**	15.4 ± 0.60	18.0 ± 0.84
	Joint Angle	**5.00 ± 0.00**	4.60 ± 0.24	2.20 ± 0.20	3.20 ± 0.20
	Limb Params.	*18.6 ± 0.40*	**19.0 ± 0.45**	14.0 ± 0.63	14.6 ± 0.51
	Unseen Robot	*7.40 ± 0.51*	**7.80 ± 0.73**	5.00 ± 0.32	5.80 ± 0.58
VT (θ; 3)	Armature	**11.2 ± 0.66**	9.80 ± 0.20	7.50 ± 0.50	9.00 ± 0.45
	Damping	**14.4 ± 0.81**	13.2 ± 0.37	9.50 ± 0.87	11.8 ± 0.49
	Density	**14.2 ± 0.86**	12.8 ± 0.20	9.00 ± 0.71	10.8 ± 0.49
	Gear	**11.6 ± 0.68**	10.4 ± 0.40	7.25 ± 0.48	8.60 ± 0.40
	Joint Angle	**3.00 ± 0.00**	2.20 ± 0.20	1.50 ± 0.29	2.00 ± 0.00
	Limb Params.	**9.40 ± 0.51**	*8.80 ± 0.20*	6.00 ± 0.41	6.60 ± 0.40
	Unseen Robot	**4.80 ± 0.37**	3.80 ± 0.20	3.00 ± 0.00	3.80 ± 0.20
OB	Armature	*1894 ± 25*	**1902 ± 29**	1644 ± 24	1732 ± 24
	Damping	*2385 ± 32*	**2405 ± 37**	2072 ± 29	2188 ± 29
	Density	*2353 ± 31*	**2360 ± 38**	1901 ± 28	2124 ± 34
	Gear	*2114 ± 29*	**2116 ± 32**	1690 ± 14	1840 ± 21
	Joint Angle	**839 ± 25**	*825 ± 20*	566 ± 8.4	643 ± 23
	Limb Params.	**1757 ± 21**	*1738 ± 25*	1435 ± 28	1486 ± 17
	Unseen Robot	**1002 ± 33**	945 ± 17	780 ± 19	851 ± 25
OB (θ)	Armature	*1427 ± 23*	**1443 ± 34**	1225 ± 26	1337 ± 33
	Damping	*1765 ± 25*	**1792 ± 46**	1516 ± 34	1653 ± 42

Continued on next page

Task	Type	VME (MM)	VME* (MM)	D (MM)	D+K (MM)
	Density	*1729 ± 28*	**1750 ± 42**	1411 ± 32	1586 ± 41
	Gear	*1576 ± 27*	**1593 ± 38**	1245 ± 31	1388 ± 46
	Joint Angle	**582 ± 9.2**	*566 ± 18*	417 ± 15	483 ± 17
	Limb Params.	**1319 ± 22**	*1304 ± 27*	1058 ± 17	1115 ± 37
	Unseen Robot	*726 ± 26*	**754 ± 15**	607 ± 23	712 ± 20
OB (3)	Armature	**30.4 ± 1.6**	*29.0 ± 0.63*	23.0 ± 1.1	25.8 ± 0.58
	Damping	**43.4 ± 1.9**	*41.8 ± 0.92*	32.8 ± 1.4	37.0 ± 0.63
	Density	**42.4 ± 1.9**	*41.2 ± 0.97*	29.2 ± 1.2	35.2 ± 0.73
	Gear	**36.0 ± 1.7**	*34.6 ± 0.60*	23.4 ± 0.98	28.0 ± 0.84
	Joint Angle	**7.20 ± 0.37**	*7.00 ± 0.45*	2.80 ± 0.20	4.60 ± 0.40
	Limb Params.	**28.2 ± 1.4**	*26.6 ± 0.60*	19.4 ± 0.68	21.4 ± 0.60
	Unseen Robot	10.6 ± 0.68	**12.0 ± 0.71**	8.60 ± 0.40	9.40 ± 0.40
OB (θ; 3)	Armature	**17.6 ± 0.81**	*16.8 ± 0.58*	12.8 ± 0.49	13.6 ± 0.68
	Damping	**24.8 ± 1.2**	*23.6 ± 0.93*	18.2 ± 0.97	19.2 ± 1.3
	Density	**24.0 ± 1.3**	*22.8 ± 0.80*	16.2 ± 0.86	17.6 ± 1.1
	Gear	**19.6 ± 1.0**	*19.0 ± 0.71*	12.8 ± 0.73	14.4 ± 1.0
	Joint Angle	**3.80 ± 0.20**	*3.60 ± 0.24*	2.20 ± 0.20	2.20 ± 0.20
	Limb Params.	**15.6 ± 0.81**	*14.6 ± 0.51*	10.4 ± 0.51	10.6 ± 0.68
	Unseen Robot	**7.00 ± 0.55**	*7.00 ± 0.45*	5.80 ± 0.37	6.00 ± 0.45
MVT	Armature	**18.7 ± 9.4**	*13.1 ± 3.3*	3.72 ± 1.0	*12.2 ± 4.6*
	Damping	**37.3 ± 13**	*26.7 ± 6.3*	6.62 ± 1.9	*20.7 ± 7.5*
	Density	**32.0 ± 10**	*21.4 ± 6.1*	4.85 ± 1.6	*22.4 ± 12*
	Gear	**19.8 ± 9.3**	*12.9 ± 3.4*	4.02 ± 1.5	*12.1 ± 4.9*
	Joint Angle	**1.81 ± 1.1**	*0.91 ± 0.40*	0.00 ± 0.00	0.35 ± 0.20
	Limb Params.	**11.9 ± 6.7**	*10.1 ± 3.1*	*4.17 ± 2.0*	*10.2 ± 3.4*
	Unseen Robot	*0.64 ± 0.64*	*0.61 ± 0.53*	*0.60 ± 0.38*	**4.15 ± 4.0**
MVT (θ)	Armature	**6.17 ± 2.7**	*4.59 ± 0.91*	0.95 ± 0.55	*3.34 ± 0.92*
	Damping	**16.7 ± 4.8**	*11.0 ± 3.7*	1.61 ± 0.11	4.43 ± 1.9
	Density	**9.49 ± 3.3**	*8.16 ± 2.6*	1.21 ± 0.56	3.64 ± 1.2
	Gear	**9.08 ± 3.1**	*6.33 ± 2.7*	0.99 ± 0.32	3.64 ± 2.2
	Joint Angle	0.00 ± 0.00	**0.06 ± 0.04**	0.00 ± 0.00	0.00 ± 0.00
	Limb Params.	**5.43 ± 2.0**	*4.29 ± 1.5*	1.68 ± 0.96	1.47 ± 0.52
	Unseen Robot	**0.61 ± 0.44**	0.03 ± 0.03	*0.20 ± 0.20*	0.00 ± 0.00

C LICENSES

ModuMorph Xiong et al. (2023), MetaMorph Gupta et al. (2022), UNIMAL Gupta et al. (2021) all have unknown/no license listed on GitHub. We plan to release our own code under an appropriate but not-yet-selected license after acceptance to the conference.