2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Motion Generation for Modular Robots Using Hierarchical Policies

Kenjiro Minamikawa*[1], Satoshi Yamamori[1], Satoshi Yagi[1], Sho Takeda[1], Kazuya Yoshida[2] and Jun Morimoto[1,3]

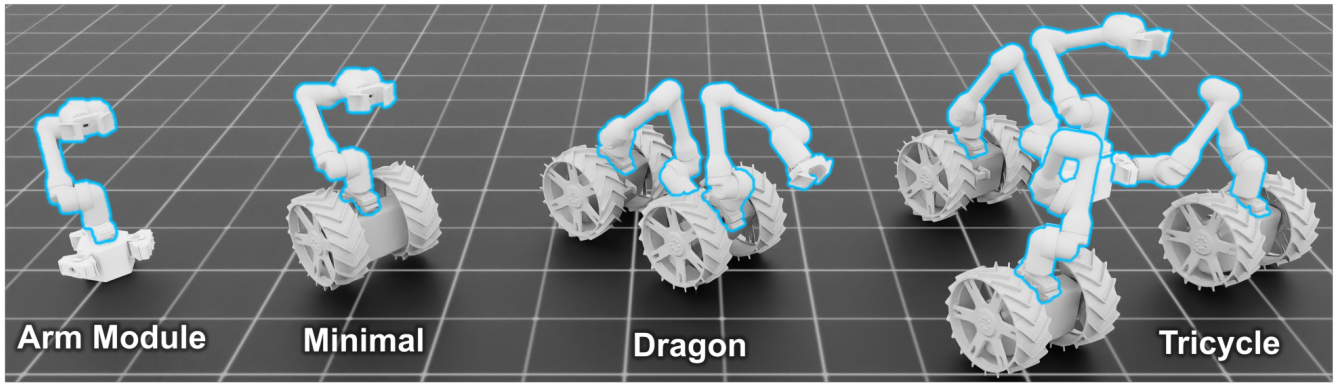


Fig. 1: Modular robot *MoonBot*, consisting of a 7-DoF arm module and three morphologies (*Minimal*, *Dragon*, and *Tricycle*) assembled from arm and wheel modules.

***Abstract*— Modular robots can be reconfigured into multiple morphologies, offering high adaptability for diverse tasks. However, reinforcement learning (RL)-based motion generation typically requires separate policy training for each morphology, and end-to-end training often fails to exploit module-specific roles. This paper proposes a hierarchical policy framework that explicitly separates control at the module level, learning reusable motion skills for each module and coordinating them with an upper-level policy for whole-body control. A single lower-level reaching policy, shared across all arm modules, is trained once and reused across morphologies, ensuring that module-specific functions are preserved even as complexity increases. The method is evaluated on the modular robot *MoonBot* in simulation, demonstrating scalable control of diverse morphologies and improved learning efficiency and interpretability over non-hierarchical baselines.**

I. Introduction

A modular robot is composed of multiple detachable modules that can be assembled in various morphologies. By altering the number and arrangement of these modules, the robot's structure can be flexibly adapted to diverse environments, enabling it to perform a wide range of tasks. Applying deep reinforcement learning (DRL) to modular robots has the potential to fully exploit this adaptability, allowing robots to autonomously acquire complex control policies. However, a key challenge is that separate policies are often trained for each robot morphology and task, leading to high computational cost and poor sample efficiency. Hierarchical reinforcement learning (HRL) has been shown to improve learning efficiency in scenarios with varying morphologies and tasks [1]–[3], but most prior approaches are limited to fixed structures or task-specific solutions.

Prior work on robotic arms has primarily focused on fixed morphologies equipped with different grippers, where the arm and gripper form a single rigid structure [4]. If the number and arrangement of arm and wheel modules could be dynamically reconfigured according to operating environments and tasks, the inherent adaptability of modular robots could be leveraged more effectively [5]. To achieve this, it is essential to develop policy construction methods that systematically reuse motion knowledge at the module level and generalize to new robot morphologies [6].

In this work, a hierarchical policy framework is proposed in which each module of a modular robot acquires reusable motion skills through lower-level training, and an upper-level policy coordinates these module-level skills to achieve scalable whole-body control. The proposed method is applied to the modular robot *MoonBot* [7], as shown in Fig. 1, and motion generation is demonstrated on multiple robot morphologies. A single, shared lower-level policy trained for the arm module's end-effector reaching is reused across all arm modules, enabling scalable control of multiple robot morphologies with varying numbers of arm and wheel modules.

The key contributions of this work are as follows:

1) A hierarchical reinforcement learning framework is proposed that decomposes control at the module level,

*Corresponding author.
(e-mail: minamikawa.kenjiro@lm.sys.i.kyoto-u.ac.jp)
[1]Graduate School of Informatics, Kyoto University, Kyoto, Japan
[2]Graduate School of Engineering, Tohoku University, Sendai, Japan
[3]Dept. of Brain Robot Interface, Computational Neuroscience Labs, ATR, Kyoto, Japan

enabling reusable, skill-oriented lower-level policies.
2) The scalability of this approach is demonstrated by applying the same trained lower-level policies to multiple robot morphologies with different numbers and arrangements of modules, without retraining the low-level skills.
3) The method is validated through extensive simulation experiments, including comparisons with non-hierarchical baselines, showing improved learning efficiency and interpretability of robot behavior.

The remainder of this paper is organized as follows. Section II reviews related work. Section III introduces the proposed HRL framework. Section IV outlines the experimental setup, and Section V reports the results. Section VI provides a discussion of the findings, and Section VII concludes the paper.

II. Related Work

A. Hierarchical Reinforcement Learning (HRL)

Recent advances in hierarchical reinforcement learning (HRL) have leveraged deep learning to scale policies to high-dimensional observation and action spaces, enabling more complex and expressive decision-making [8]–[10]. For example, HIRO [8] introduced an efficient off-policy algorithm for training hierarchical policies with raw state goal representations, whereas the Option-Critic architecture [10] jointly learns both high-level and lower-level policies using an on-policy approach. Surveys such as [3], [11] have highlighted HRL ability to improve exploration, reuse learned skills, and address long-horizon credit assignment challenges. HRL has also been applied in robotics, including locomotion [12], [13], manipulation [14], [15], multi-task learning [16], [17].

Previous studies have primarily focused on policy acquisition for robots with a single morphology, whereas our HRL method introduces a modular policy that can be applied across multiple morphologies. In the problem setting considered in this work, the low-level policy is responsible for goal-conditioned reaching. Although inverse kinematics (IK) can provide an analytical solution for such control, we instead adopt a reinforcement learning–based formulation to promote generalization across different modules and to facilitate extension to more complex control objectives.

B. Skill Composition and Transfer

A key challenge in scaling reinforcement learning (RL) to complex robotic systems is the ability to reuse and compose skills learned in simpler settings. Early approaches such as the Options framework [18], [19] introduced temporal abstractions that enable policies to be expressed as reusable sub-policies, laying the foundation for skill-based RL. Subsequent works have explored learning skill embeddings to facilitate transfer and composition, where low-level skills are represented in a latent space that allows a higher-level policy to adaptively combine them for downstream tasks [20]–[22].

While conventional approaches often rely on latent representations or sub-policies to acquire motion skills, our results suggest that the modular structure of the robot's body itself can be leveraged as a basis for skill acquisition.

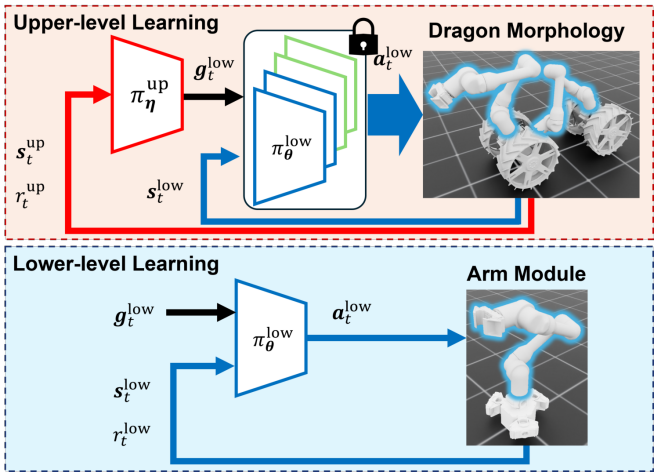


Fig. 2: Hierarchical policy learning architecture with reusable module skills: the lower policy $\pi_{\boldsymbol{\theta}}^{\text{low}}(\boldsymbol{a}^{\text{low}} \mid \boldsymbol{s}^{\text{low}}, \boldsymbol{g}^{\text{low}})$ is trained via GCRL on modular components, and the upper policy $\pi_{\boldsymbol{\eta}}^{\text{up}}(\boldsymbol{a}^{\text{up}} \mid \boldsymbol{s}^{\text{up}}, \boldsymbol{g}^{\text{up}})$ issues subgoals $(\boldsymbol{a}^{\text{up}}{=}\boldsymbol{g}^{\text{low}})$ to coordinate whole-body motion; lower-level policy parameter $\boldsymbol{\theta}$ is kept fixed during upper-level training.

C. Modular Robots

Modular robots are composed of multiple detachable modules that can be reconfigured into a variety of morphologies to perform diverse tasks. Early research focused on self-reconfigurable hardware platforms, such as lattice-type and chain-type modular robots, demonstrating autonomous shape-shifting for navigation and manipulation [23], [24]. Later studies introduced distributed planning and control methods for adaptive reconfiguration [25], [26]. With advances in deep learning and simulation, reinforcement learning (RL) has been increasingly applied to modular robots, enabling end-to-end training of locomotion and manipulation behaviors without handcrafted controllers [6], [27], [28].

This study extends previous modular approaches to enable coordinated control across reconfigurable high-DoF morphologies. Our method introduces a hierarchical RL framework that treats multi-DoF arms as modular building blocks. This design allows combining these modules into diverse morphologies while ensuring coordinated motion control.

III. Methods

A. Goal-conditioned Reinforcement Learning

Goal-conditioned reinforcement learning (GCRL) [29] is formalized in this section to enable hierarchical policy construction. A goal-augmented Markov Decision Process (GA-MDP), which extends the standard MDP framework [30], is defined as: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, r, \mathcal{P}, p_g, \gamma)$. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, and $\mathcal{G}$ the goal space. The reward function is defined as $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$, $\mathcal{P}$ represents the state transition probability, p_g is the goal distribution, and $\gamma \in [0, 1)$ is the discount factor. At each

timestep t, the agent observes a state $\boldsymbol{s}_t \in \mathcal{S}$ and selects an action $\boldsymbol{a}_t \in \mathcal{A}$ conditioned on a goal $\boldsymbol{g} \in \mathcal{G}$, according to a parameterized policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t, \boldsymbol{g})$. A scalar reward r_t is then received, and the environment transitions to a new state $\boldsymbol{s}_{t+1} \sim \mathcal{P}(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t)$. The objective is to find an optimal policy $\pi^*_{\boldsymbol{\theta}}$ that maximizes the expected discounted return:

$$J(\theta) = \mathbb{E}_{\tau|\pi_{\boldsymbol{\theta}}, p_{\boldsymbol{g}}} \left[\sum_{t=0} \gamma^t r_t \right], \tag{1}$$

where $\tau = (\boldsymbol{g}, \boldsymbol{s}_0, \boldsymbol{a}_0, \dots)$ denotes a trajectory.

The proposed framework incorporates the concept of Universal Value Function Approximators (UVFA [31]) and extends the Proximal Policy Optimization (PPO [32]) algorithm to be conditioned on the goal space. The advantage function A_π under policy π is defined in Eq. 2 using the value function V_π and the action-value function Q_π:

$$A_\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{g}) = Q_\pi(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{g}) - V_\pi(\boldsymbol{s}, \boldsymbol{g}). \tag{2}$$

In PPO, the policy $\pi_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ is optimized to maximize the following clipped surrogate objective:

$$L(\boldsymbol{\theta}) = \mathbb{E}_t \left[L_t^{CLIP}(\boldsymbol{\theta}) - c_1 L_t^{VF}(\boldsymbol{\theta}) + c_2 S[\pi_{\boldsymbol{\theta}}](\boldsymbol{s}_t) \right], \tag{3}$$

where c_1 and c_2 are coefficients, S denotes the entropy bonus, L^{VF} is the squared error loss of the value function, and L^{CLIP} is the clipped surrogate objective in PPO, which is computed using the advantage function defined in Eq. (2).

B. Hierarchical Policy Construction

An overview of the proposed hierarchical policy architecture, consisting of two levels, is illustrated in Fig. 2. This architecture is designed to construct a lower-level policy around modular components and an upper-level policy that reuses these modular policies across different morphologies. The lower-level policy training $\pi^{\text{low}}_{\boldsymbol{\theta}}(\boldsymbol{a}^{\text{low}}|\boldsymbol{s}^{\text{low}}, \boldsymbol{g}^{\text{low}})$ uses GCRL at the module level (e.g., arm module), acquiring required motion skills in the upper level. The upper-level policy $\pi^{\text{up}}_{\boldsymbol{\eta}}(\boldsymbol{a}^{\text{up}}|\boldsymbol{s}^{\text{up}}, \boldsymbol{g}^{\text{up}})$ outputs goal states $\boldsymbol{g}^{\text{low}}$ for the lower-level policy as its actions $\boldsymbol{a}^{\text{up}}$, enabling coordinated whole-body motion by integrating the behaviors of individual modules. During upper-level training, the parameters $\boldsymbol{\theta}$ of the lower-level policies are kept fixed to ensure training stability and facilitate skill reusability. Goal information plays a crucial role in transmitting low-level skills to the upper level. While the goal state remains nearly time-invariant at the lower level, it is dynamically generated at each time step by the upper-level policy. This module-centric decomposition allows each module to acquire natural and consistent local behaviors, which can be leveraged for full-body control without re-training the low-level dynamic controllers. Furthermore, safety constraints and physical limitations can be encoded at the module level, reducing the need for explicit reasoning about them in high-level planning. By using goals with explicit parameters such as positions and orientations, the upper-level policy can explore the motion skill space, avoid infeasible regions, and improve training efficiency.

TABLE I: State and action space of the lower-level policy.

Type	Component	Set
State	Joint angles	$\mathbb{R}^7$
	Joint angular velocities	$\mathbb{R}^7$
	End-effector goal position	$\mathbb{R}^3$
	End-effector goal orientation	$\mathbb{H}$
Action	Joint angle increments	$\mathbb{R}^3 \times \mathbb{H}$

TABLE II: Network and training hyperparameters.

Parameter	Value
Hidden layers	256, 128
Learning rate	0.001
Discount factor γ	0.99
GAE parameter λ	0.95
Clipping range ϵ	0.2
Entropy coefficient	0.01
Episode length	360 steps
Control frequency	30 Hz
Total training steps	480,000

IV. Experiments

This study evaluates the proposed hierarchical policy construction through simulation experiments on *MoonBot*, which can be reconfigured into three morphologies, *Minimal*, *Dragon*, and *Tricycle*, by attaching or detaching arm and wheel modules (Fig. 1). These diverse morphologies enable the robot to adapt to a surrounding environment and various tasks. The experiments focused on a reaching task, in which the robot moves its arm toward a target position and orientation. To successfully complete the task, the robot must coordinate the behavior of each module while implicitly satisfying task constraints, such as maintaining leg contact with the ground and avoiding falls. To address this challenge, a hierarchical reinforcement learning approach is proposed, in which the lower-level policy is trained to control the arm module, allowing the upper-level policy to reuse module-level motion skills across different reconfigured morphologies. The proposed method is compared with a non-hierarchical baseline that directly outputs joint-level actions without modular decomposition. All simulation experiments are conducted using Nvidia IsaacLab [33].

A. Lower-level Policy Learning

The lower-level policy is trained on an arm reaching task to move the end-effector to target poses, through which it acquires reusable motion skills leveraged by the upper-level policy. The arm module's state and action spaces are summarized in Table I. The policy is given the goal state $\boldsymbol{g}^{\text{low}}$, which is defined as the target position ($\mathbb{R}^3$) and orientation ($\mathbb{H}$, represented as a quaternion) of the end-effector. The target is resampled every 4 seconds from the specified position range (x, y, z) from $(-0.1, -0.3, 0.6)$ to $(0.6, 0.3, 1.3)$ [m] and orientation range $(\text{roll}, \text{pitch}, \text{yaw})$ from $(-\pi, -\pi/3, 0)$ to $(\pi, \pi/3, 0)$, with the sampled orientation converted to a

TABLE III: State and action space of the upper-level policy.

Type	Component	Set
State	Base linear velocity	$\mathbb{R}^3$
	Projected gravity vector	$\mathbb{R}^3$
	End-effector–goal relative pose	$\mathbb{R}^4$
	Wheel angular velocities	$\mathbb{R}^{2n_{\text{wheel}}}$
	Previous action	$\mathbb{R}^{\dim(\mathcal{A})}$
Action	Desired end-effector pose	$\mathbb{R}^{d_{\text{arm}}}$
	Wheel velocity	$\mathbb{R}^{n_{\text{wheel}}}$

quaternion. The target pose is defined in a fixed global, right-handed coordinate frame centered at the arm base.

To emulate inputs from the upper level and improve robustness to dynamically changing targets, small random noise is added to the goal state at each timestep. The base position and orientation are fixed throughout training. The initial joint position is uniformly sampled from the range of $(-0.5, 0.5)$ [rad] for each joint.

The reward function is defined in Eq. 4. Here, $\boldsymbol{p}_{\text{ref}}$ and $\boldsymbol{\omega}_{\text{ref}}$ denote the target position and orientation of the end-effector, while $\boldsymbol{p}_{\text{ee}}$ and $\boldsymbol{\omega}_{\text{ee}}$ represent its current position and orientation. $\dot{\boldsymbol{q}}_t$ denotes the vector of joint angular velocities at timestep t. The reward penalizes the Euclidean distance to the target position, orientation error (via a quaternion logarithmic map), and adds regularization on action smoothness and joint velocities. The weights w_p, w_o, w_a, w_v and scaling factors α_p, α_o are chosen empirically to balance accuracy and smooth control.

$$\begin{aligned} r_t^{\text{low}} = w_p \exp\big(-\alpha_p \|\boldsymbol{p}_{\text{ref}} - \boldsymbol{p}_{\text{ee}}\|_2\big) \\ - w_o \exp\big(-\alpha_o \|\log(\boldsymbol{\omega}_{\text{ref}}^{-1}\boldsymbol{\omega}_{\text{ee}})\|_2\big) \\ - w_a \|\boldsymbol{a}_t - \boldsymbol{a}_{t-1}\|_2^2 - w_v \|\dot{\boldsymbol{q}}_t\|_2^2. \end{aligned} \tag{4}$$

The network architecture and training parameters are summarized in Table II.

B. Upper-level Motion Generation

This experiment evaluates whether the proposed framework can leverage module-level policies to achieve coordinated locomotion and manipulation across the three morphologies. To this end, the modular robot is assigned a reaching task using an arm module attached to each morphology for manipulation. Targets are placed at varying distances, often requiring body locomotion before the arm can reach them. Each *MoonBot* morphology utilizes multiple modules, leading to a vast number of possible policies for a single reaching task. To obtain natural behaviors from this redundancy, implicit task constraints must be incorporated during learning. This experiment demonstrates that the hierarchical framework enables the acquisition of whole-body motion skills from limited reward signals at the upper level by inheriting the motion skills learned at the lower level.

Since *MoonBot* is composed of arm and wheel modules, the upper-level policy manages whole-body control by outputting goal states for each module: a desired position and orientation for each arm module, and a desired velocity for each wheel module. The module composition of each morphology is as follows: the *Minimal* consists of one arm and one wheel, the *Dragon* has two arms and two wheels, and the *Tricycle* has four arms and three wheels, as illustrated in Fig. 1.

TABLE IV: Reward terms for the upper-level training.

Term	Expression
Position reward	$w_{p1}\left(1 - \tanh\left(\frac{d_p}{2.0}\right)\right)$
Close-range position reward	$w_{p2}\left(1 - \tanh\left(\frac{d_p}{0.2}\right)\right)$
Orientation error penalty	$-w_o\|\theta_h\|_2$
Joint velocity penalty	$-w_v\|\dot{\boldsymbol{q}}\|_2^2$
Action smoothness penalty	$-w_a\|\boldsymbol{a}_t - \boldsymbol{a}_{t-1}\|_2^2$

TABLE V: Morphology-specific arm module configuration

Morphology	Num. Modules	d_{arm}	Threshold
Minimal	1	6	0.6 [rad]
Dragon	2	10	1.0 [rad]
Tricycle	4	18	0.785 [rad]

The observation and action spaces are defined for each morphology, with their dimensions scaling according to the number of arm and wheel modules, as shown in Table III. The *End-effector-goal relative pose* represents the relative position and the relative yaw angle between the current and target poses of the end-effector. d_{arm} denotes the number of dimensions of the goal state specified by the upper level for each arm module, and this value varies depending on the robot morphology, as detailed later in this section. Similarly, n_{wheel} represents the number of wheel modules, and the action space includes wheel angular velocity commands for each wheel module.

The target position and yaw orientation are sampled at the start of each episode within predefined ranges: the position sampled in (x, y, z) from $(-3.0, -3.0, z_{\text{min}}^M)$ to $(3.0, 3.0, z_{\text{max}}^M)$ [m] and the yaw angle in $[-\pi, \pi]$, where z_{min}^M and z_{max}^M are morphology-specific height limits. The robot's initial base pose is also sampled, with the position (x, y) from $(-0.5, -0.5)$ to $(0.5, 0.5)$ [m] and the yaw angle in $[-\pi, \pi]$.

The reward function is composed of the terms summarized in Table IV. Here, d_p denotes the Euclidean distance between the target position and the end-effector position, and θ_h represents the difference in yaw angle between the target orientation and the current end-effector orientation.

Both the upper-level policy and value networks are implemented as multilayer perceptrons (MLPs) with two hidden layers of 512 and 256 nodes, respectively. The same set of PPO hyperparameters is used as in the lower layer. During the upper-level training, the parameters of the lower-level module policies remain fixed. The control frequency is also set to 30 Hz, and training is conducted for a total of 480,000 control steps, identical to the lower-level training setup.

Several configuration parameters are specified for each morphology, and their detailed setups are described below

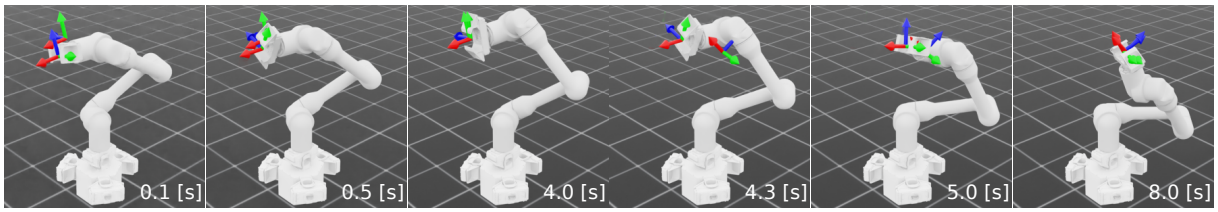


Fig. 3: Snapshots of the trained lower-level policy executing on the arm module. The goal pose (target) and the end-effector pose are visualized as coordinate frames. The task is to align the end-effector's position and orientation with the target pose. The target is resampled every 4 seconds, and the arm consistently tracks and reaches the updated targets.

and illustrated in Table V. Each morphology uses a combination of manipulation and leg-part arm modules. Arm-modules are controlled via a 7D goal space (position and orientation), while leg-part modules use a simplified 4D space (position and roll only). Episodes terminate when the robot base tilts beyond a morphology-specific threshold. In the *Minimal* morphology, an alive reward is added to encourage upright posture throughout the episode: $r_{\text{alive}} = w_{\text{alive}}$, where $w_{\text{alive}} = 0.2$ is a constant positive reward applied at each timestep the robot remains upright.

C. Baseline: Non-hierarchical Policy

To demonstrate the effectiveness of the proposed hierarchical approach, a non-hierarchical policy is trained as a baseline for each morphology under the same experimental setup. In this baseline, the action vector directly specifies all joint positions of the arm modules and the velocities of the wheel modules, without any hierarchical modular decomposition. The observation space of the non-hierarchical policy includes both the upper-level states and the low-level arm module states of all modules. The network has the same architecture as the upper-level policy (512- and 256-unit layers), and training hyperparameters match those of the hierarchical policy.

Additionally, in supplementary experiments with the non-hierarchical policy using the *Tricycle* morphology, two reward terms are introduced to promote stable wheel–ground interaction and to prevent undesirable lower-arm contacts.

An orientation reward term is introduced to encourage the lower wheel-equipped arms to orient downward, promoting stable wheel–ground contact:

$$r_{\text{ori}} = w_{\text{ori}} \exp\left(-\sum_{i=1}^{3} \theta_i^g\right). \tag{5}$$

Here, $\theta_i^g \in [0, \pi]$ denotes the angle between the local x-axis of the i-th leg end-effector and the gravity vector, and w_{ori} is a weighting coefficient. This term encourages each leg end-effector to orient its x-axis downward, thereby improving wheel–ground contact stability.

The contact penalty term is defined as a binary penalty applied when undesired collisions involving the lower arm modules are detected. Specifically, the reward is set to $-w_c$ if such contact occurs and to zero otherwise:

$$r_{\text{contact}} = \begin{cases} -w_c, & \text{if lower-arm contact occurs,} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

This penalty discourages configurations that lead to unintended contacts and helps maintain stable support and locomotion.

V. Results

A. Lower-level Policy Performance

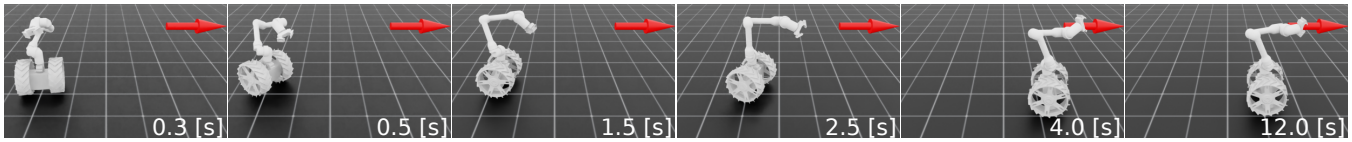
Fig. 3 presents a sequence of images demonstrating the trained policy. The target is resampled every 4 seconds, and the arm successfully tracks and reaches each updated target. Upon reaching the target site, the end-effector exhibits a position error of 0.045 ± 0.035 m and an orientation error of 0.43 ± 0.35 rad. This low-level policy serves as a reusable skill for training higher-level policies across the three robot morphologies.

B. Upper-level Policy Performance

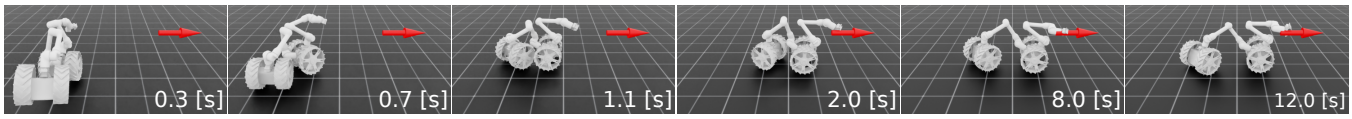
The upper-level training results show that the robot can be effectively controlled in all three morphologies (Fig. 4). This is achieved by leveraging the lower-level module policies to control end-effector poses. In the *Minimal* morphology, the wheel module is used to rotate and reposition, after which the arm aligns its end-effector to the target. In the *Dragon* morphology, the robot achieves locomotion and reaching by steering through active reorientation of the rear wheel module. Because the wheel modules receive identical velocity commands for both wheels, body rotation relies on adjusting the orientation of the rear wheel module, indicating that the lower arm module is effectively utilized for locomotion control. In the *Tricycle* morphology, the three wheel-equipped arm modules are skillfully reoriented to change the robot's heading while approaching the target. Near the target, the entire body is used to fine-tune the end-effector pose. All images illustrate the state at the end of the 12-second episode, confirming that the robot is able to maintain stability and accurately reach the target pose.

C. Comparison with Non-hierarchical Policy

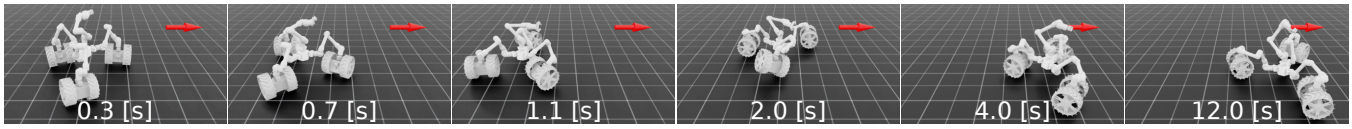
The proposed hierarchical policy was compared against a non-hierarchical baseline trained under identical task settings. Fig. 6 presents box plots of the end-effector position and orientation errors at the end of each episode, computed over 1000 rollouts. For the *Minimal* morphology, the non-hierarchical policy achieves slightly better, though still comparable, accuracy in both position and orientation. This outcome is expected, as the simple body structure allows the



(a) *Minimal*: The wheel module is used to rotate the body before reaching the target.

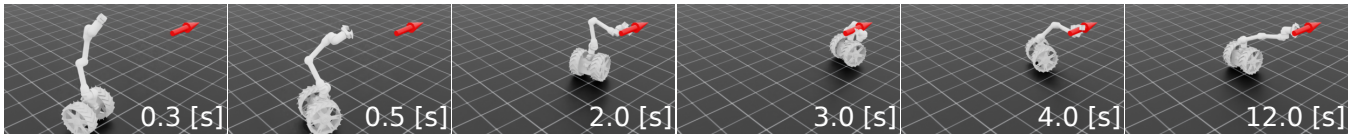


(b) *Dragon*: Steering through the rear wheel module enables precise body orientation control and reaching.

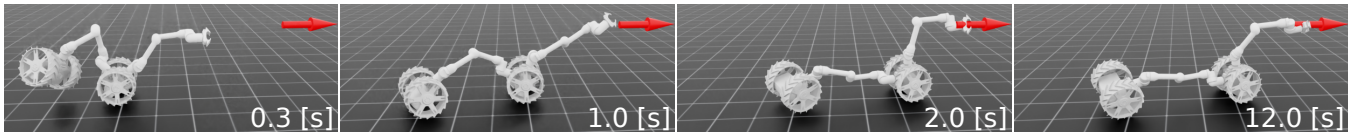


(c) *Tricycle*: The three wheel-equipped arm modules are used for locomotion and fine positioning.

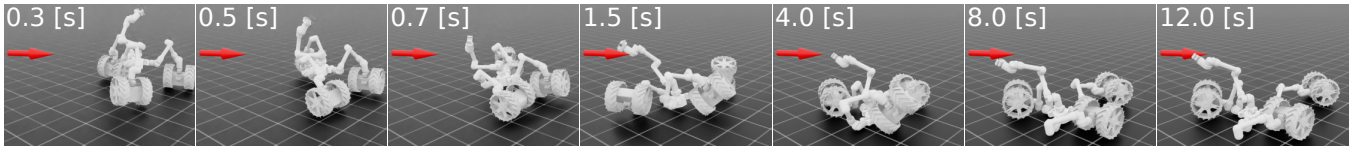
Fig. 4: Snapshots of execution sequences across the three robot morphologies. Each sequence illustrates how the robot moves and reaches the target using different configurations. The task is to align the end-effector of the designated manipulation arm module with the target's position and yaw orientation, shown as the red arrow in the figures.



(a) *Minimal*



(b) *Dragon*



(c) *Tricycle*

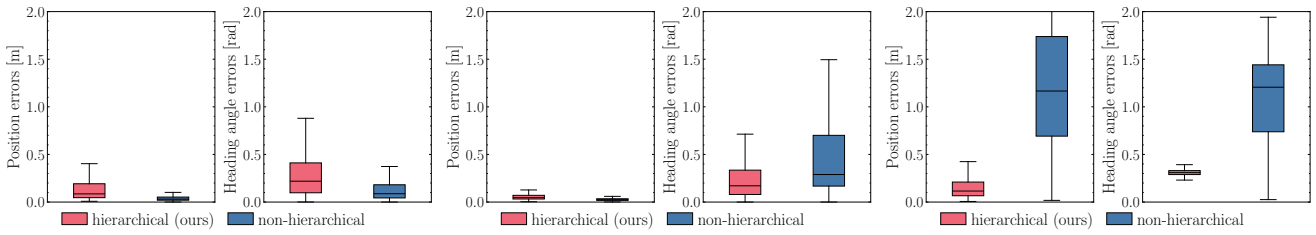
Fig. 5: Unintended postures of *Minimal*, *Dragon*, and *Tricycle* morphologies generated by a non-hierarchical policy.

non-hierarchical policy to issue direct joint commands without relying on a lower-level controller, thereby providing a minor advantage. For the *Dragon* morphology, the proposed hierarchical and non-hierarchical policies exhibited comparable accuracy in position reaching. However, the hierarchical policy achieved substantially better orientation accuracy. This improvement can be attributed to the relatively complex body structure, where the lower-level policy progressively demonstrates its effectiveness. For the *Tricycle* morphology, the reaching performance of the non-hierarchical policy degraded markedly. These results suggest that the advantage of the hierarchical policy becomes more pronounced as the number of modules increases.

Furthermore, under the non-hierarchical setup, the policies frequently produced unnatural and unintended postures. As shown in Fig. 5, the robot extends its end-effector rigidly rather than leveraging the kinematic flexibility of its individual modules.

We further examined how much additional prior knowledge needs to be embedded into the non-hierarchical policy through reward engineering to achieve performance comparable to the proposed hierarchical control policy. This analysis highlights the advantage of employing a lower-level policy that naturally constrains joint movements. Concretely, we augmented the original reaching-task reward with additional penalty terms.

In the *Tricycle* morphology, performance degradation was primarily caused by frequent failures to maintain proper posture, which often led to arm-ground contact and rendered the wheel modules unusable, as illustrated in Fig. 5c. The large



(a) *Minimal* (b) *Dragon* (c) *Tricycle*

Fig. 6: Comparison of hierarchical and non-hierarchical policies on the reaching task across different morphologies, computed over 1000 rollouts.

number of degrees of freedom also resulted in frequent inter-module contacts. To mitigate these issues, two additional reward terms were introduced in the non-hierarchical policy experiments to encourage stable posture and to discourage undesirable contacts, as defined in Eqs. (5) and (6).

A parameter search was conducted over eight different combinations of reward weights for end-effector pose accuracy, lower-arm orientation reward w_{ori}, and contact penalties w_c. Table VI reports the best-performing configuration together with additional evaluation metrics for the *Tricycle* morphology. The *Leg EE orientation* shows the orientation deviations of the lower wheel-equipped arm modules, computed at the end of each episode using the same formulation as Eq. 5. The *Contact count* shows the total number of undesired contacts detected throughout each episode, and the *Termination count* indicates the number of episodes (out of 1000 rollouts) that ended prematurely due to a termination condition. Together, these metrics provide insight into posture stability, collision avoidance, and overall robustness of the learned policies. The results demonstrate that carefully designed penalties can improve overall performance, bringing it closer to that of the hierarchical policy. However, the hierarchical approach achieves this level of performance without explicitly tuning such penalties, indicating that it effectively leverages the intended roles of individual modules. It is worth noting that several parameter settings for the non-hierarchical policy led to training collapse, with the robot failing to perform the reaching task altogether, highlighting the difficulty of manual reward tuning.

TABLE VI: Comparison of hierarchical and penalty policies in the *Tricycle* morphology.

Metric	Hierarchical	Penalty
Position [m]	0.17 ± 0.18	$\mathbf{0.08 \pm 0.18}$
Orientation [rad]	$\mathbf{0.31 \pm 0.03}$	$0.38 \pm 0.0.29$
Leg EE orientaion [m]	$\mathbf{0.44 \pm 0.24}$	0.62 ± 0.37
Contact count	$\mathbf{117.14 \pm 88.95}$	194.18 ± 121.13
Termination count	$\mathbf{2.80 \pm 1.54}$	6.00 ± 2.79

VI. Disucussion

As shown in the Fig. 5, the non-hierarchical policy tends to minimize whole-body movements across morphologies. While the main task is accomplished, the individual modules are not utilized and exhibit inefficient behaviors. Reward engineering, which involves designing additional sub-rewards, has traditionally been used to mitigate such behaviors. However, as the morphology becomes more complex, the cost of tuning trade-offs between rewards increases. In contrast, the hierarchical framework demonstrates scalability for morphologies with many modules by reusing the motion skills acquired in lower-level policies at the higher level.

This decomposition not only simplifies reward design but also naturally enforces module-specific roles (e.g., manipulation modules specialize in precise manipulation tasks, while leg modules focus on locomotion and orientation control). By embedding constraints and desired behaviors into the lower-level policies, the upper level can achieve effective whole-body coordination without extensive lower-level fine-tuning. As a result, the hierarchical approach yields simpler, more interpretable policies and demonstrates improved scalability for robots with many modules.

VII. Conclusion

This study presented a motion construction framework for modular robots based on hierarchical reinforcement learning. Reusable motion skills were acquired at the module level, and an upper-level policy coordinated these module-specific behaviors to achieve whole-body motion control. The proposed modular hierarchy preserves the functional roles of individual modules, resulting in scalable and interpretable control even as the number of modules increases. Simulation experiments on multiple *MoonBot* morphologies demonstrated that a single lower-level reaching policy can be reused across all arm modules, eliminating the need for retraining low-level skills. Comparisons with non-hierarchical baselines showed improved learning efficiency and reduced reliance on extensive reward engineering. Although this study is evaluated only in simulation, existing sim-to-real frameworks can be applied to the proposed hierarchical structure. Extending hierarchical policy learning across diverse morphologies is a promising direction for future work.

Acknowledgements

We would like to express our sincere gratitude to Assistant Professor Kentaro Uno at Tohoku University and the development team for providing the MoonBot model. This work was supported by Japan Science and Technology Agency's Moonshot R&D Program (Grant No. JPMJMS223B-03), and JSPS KAKENHI Grant Number: 22H04998 and 23K24925.

REFERENCES

[1] S. Takeda, S. Yamamori, S. Yagi, and J. Morimoto, "Hierarchically Connecting Modularly-Learned Policies to Generate a Controller for a Combined Robot System," in *IEEE 21th International Conference on Automation Science and Engineering*, 2025.

[2] J. Morimoto and K. Doya, "Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning," *Robotics and Autonomous Systems*, vol. 36, no. 1, pp. 37–51, Jul. 2001.

[3] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, Jun. 2021.

[4] S. Takeda, S. Yamamori, S. Yagi, and J. Morimoto, "An empirical evaluation of a hierarchical reinforcement learning method towards modular robot control," *Artificial Life and Robotics*, Jan. 2025.

[5] G. Liang, D. Wu, Y. Tu, and T. L. Lam, "Decoding modular reconfigurable robots: A survey on mechanisms and design," *The International Journal of Robotics Research*, vol. 44, no. 5, pp. 740–767, 2025.

[6] J. Whitman, M. Travers, and H. Choset, "Learning modular robot control policies," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 4095–4113, 2023.

[7] K. Uno, E. Neppel, G. H. Diaz, A. Mishra, S. Karimov, A. S. Jain, A. Habib, P. Pama, H. Gozbasi, S. Santra *et al.*, "Moonbot: Modular and on-demand reconfigurable robot towards moon base construction," *IEEE Transactions on Field Robotics*, 2025.

[8] O. Nachum, S. S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[9] A. Levy, G. Konidaris, R. Platt, and K. Saenko, "Learning multi-level hierarchies with hindsight," *arXiv preprint arXiv:1712.00948*, 2017.

[10] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[11] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, pp. 41–77, 2003.

[12] D. Jain, A. Iscen, and K. Caluwaerts, "Hierarchical reinforcement learning for quadruped locomotion," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 7551–7557.

[13] T. Li, N. Lambert, R. Calandra, F. Meier, and A. Rai, "Learning generalizable locomotion skills with hierarchical reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 413–419.

[14] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller, "Learning an embedding space for transferable robot skills," in *International Conference on Learning Representations*, 2018.

[15] X. Yang, Z. Ji, J. Wu, Y.-K. Lai, C. Wei, G. Liu, and R. Setchi, "Hierarchical reinforcement learning with universal policies for multistep robotic manipulation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 9, pp. 4727–4741, Sep. 2022.

[16] E. Triantafyllidis, F. Acero, Z. Liu, and Z. Li, "Hybrid hierarchical learning for solving complex sequential tasks using the robotic manipulation network roman," *Nature Machine Intelligence*, vol. 5, 09 2023.

[17] C. Hao, C. Weaver, C. Tang, K. Kawamoto, M. Tomizuka, and W. Zhan, "Skill-critic: Refining learned skills for hierarchical reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3625–3632, 2024.

[18] M. Stolle and D. Precup, "Learning options in reinforcement learning," in *Abstraction, Reformulation, and Approximation*, S. Koenig and R. C. Holte, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 212–223.

[19] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, 1999.

[20] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Transactions On Graphics (TOG)*, vol. 41, no. 4, pp. 1–17, 2022.

[21] Y. Kuang, H. Geng, A. Elhafsi, T.-D. Do, P. Abbeel, J. Malik, M. Pavone, and Y. Wang, "Skillblender: Towards versatile humanoid whole-body loco-manipulation via skill blending," *arXiv preprint arXiv:2506.09366*, 2025.

[22] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum, "OPAL: Offline primitive discovery for accelerating offline reinforcement learning," in *International Conference on Learning Representations*, 2021.

[23] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss, "Structure decision method for self organising robots based on cell structures-cebot," in *Proceedings, 1989 International Conference on Robotics and Automation*, 1989, pp. 695–700 vol.2.

[24] M. Yim, W.-m. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[25] Z. Butler, S. Byrnes, and D. Rus, "Distributed motion planning for modular robots with unit-compressible modules," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, Oct 2001, pp. 790–796 vol.2.

[26] R. Fitch, Z. Butler, and D. Rus, "Reconfiguration planning for heterogeneous self-reconfiguring robots," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2003, pp. 2460–2467 vol.3.

[27] J. Wang, C. Hu, and Y. Zhu, "CPG-based hierarchical locomotion control for modular quadrupedal robots using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7193–7200, 2021.

[28] A. Mishra, S. Santra, E. Neppel, E. M. R. Lombardi, S. Karimov, K. Uno, and K. Yoshida, "Multi-modal decentralized reinforcement learning for modular reconfigurable lunar robots," *arXiv preprint arXiv:2510.20347*, 2025.

[29] M. Liu, M. Zhu, and W. Zhang, "Goal-conditioned reinforcement learning: Problems and solutions," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, L. D. Raedt, Ed., 7 2022, pp. 5502–5511, survey track.

[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[31] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1312–1320.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[33] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit - A Unified Simulation Framework for Interactive Robot Learning Environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, 2023.