IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Decentralized Motor Skill Learning for Complex Robotic Systems

Yanjiang Guo[1*], Zheyuan Jiang[1*], Yen-Jen Wang[1], Jingyue Gao[1], Jianyu Chen[1,2]

***Abstract*—Reinforcement learning (RL) has achieved remarkable success in complex robotic systems (eg. quadruped locomotion). In previous works, the RL-based controller was typically implemented as a single neural network with concatenated observation input. However, the corresponding learned policy is highly task-specific. Since all motors are controlled in a centralized way, out-of-distribution local observations can impact global motors through the single coupled neural network policy. In contrast, animals and humans can control their limbs separately. Inspired by this biological phenomenon, we propose a Decentralized motor skill (DEMOS) learning algorithm to automatically discover motor groups that can be decoupled from each other while preserving essential connections and then learn a decentralized motor control policy. Our method improves the robustness and generalization of the policy without sacrificing performance. Experiments on quadruped and humanoid robots demonstrate that the learned policy is robust against local motor malfunctions and can be transferred to new tasks.**

***Index Terms*—Robot learning, Reinforcement learning**

I. INTRODUCTION

RECENTLY, many complex robotic systems have been developed and demonstrated. Quadruped robots, for instance, have shown their ability to traverse diverse challenging terrains [1], [2], [3]. Furthermore, attaching an additional arm to a quadruped robot allows it to perform various manipulation tasks [4]. Some impressive biped and humanoid robots also show great improvements in their locomotion and manipulation skills [5], [6], [7]. However, controlling these complex robotic systems is a challenging task. Due to complex robot dynamics and environments, model-based control methods might fail when the model largely deviates from the ground truth. Reinforcement learning provides an alternative way to learn control policies without the need to model the dynamics and environments. Such mechanism helps improve robustness and generalization [1], [2], [3], [8].

A typical reinforcement learning controller for robot motor control is implemented as a centralized neural network policy.

Manuscript received: March 23, 2023; Revised June 20, 2023; Accepted July 15, 2023.

This paper was recommended for publication by Editor A. Faust upon evaluation of the Reviewers' comments. This work was supported by the Ministry of Science and Technology of the People's Republic of China, the 2030 Innovation Megaprojects "Program on New Generation Artificial Intelligence" (Grant No. 2021AAA0150000).

*Yanjiang Guo and Zheyuan Jiang contributed equally to this work.

[1]Yanjiang Guo, Zheyuan Jiang, Yen-Jen Wang, Jingyue Gao, Jianyu Chen are with Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China. guoyj22@mails.tsinghua.edu.cn

[2]Jianyu Chen is also with Shanghai Qi Zhi Institute, Shanghai, China (Corresponding author). jianyuchen@tsinghua.edu.cn

Digital Object Identifier (DOI): see top of this page.

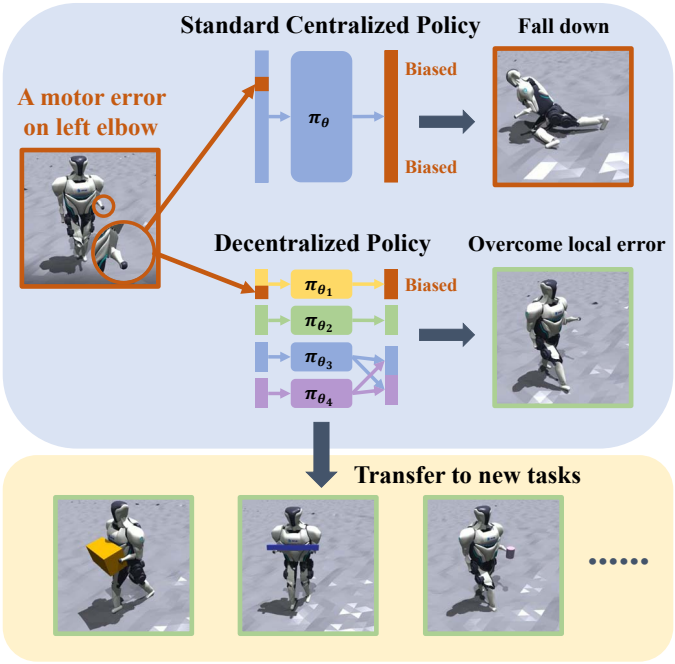


Fig. 1: Illustration of our motivation. Standard centralized policies are highly task-specific and local motor malfunction can influence actions for all motors. **DEMOS** can automatically decouple some motor groups, making the policy more robust and transferable.

The input to the policy is some important observations, such as root pose, joint pose, joint velocity, etc. All these observations are concatenated into a single vector, which is directly fed to the neural network, and then the neural network will directly output the actions such as target poses or the torques of the joints. Such an approach has some important drawbacks: 1) The policy lacks robustness. Since all motors are controlled in a centralized way, out-of-distribution local observations can impact global motors through the single coupled neural network policy, as shown in Figure 1. 2) The converged policy is highly specific to the training task and will be failed when transferred to new tasks.

In order to address the challenges of controlling complex robotic systems, previous work has employed a decentralized approach. De et al. manually partitioned the robot motors into multiple groups and treated each group as a single agent [9]. These agents used their local observations to control their corresponding motors. However, this manual partitioning could potentially disrupt the coordination between motor groups and sacrifice overall performance. In contrast, animals and humans have adopted a gated mechanism [10], [11] that automatically

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

decides which modules need to cooperate with each other and which can function independently. For instance, humans can still walk naturally with a broken arm, as it is independent from their legs, but humans would struggle to walk with a broken leg, which requires coordination with the other leg.

Inspired by this biological phenomenon, we propose a **De**centralized **mo**tor **s**kill learning (**DEMOS**) method under the RL framework. DEMOS automatically discovers motor groups that can be decoupled from each other while preserving essential connections and then learn a decentralized motor control policy, as shown in Figure 1. Specifically, DEMOS first divide robots into potentially decoupled motor groups and then introduces a decentralized objective to encourage each group to influence others as little as possible. Finally, DEMOS decouple groups with weak connections and obtain a decentralized policy. Further experiments on quadruped robots and humanoid robots show the robustness and generalization of the learned decentralized policy.

Our main contributions can be summarized as follows:

- We propose a distributed motor skill learning (DEMOS) method inspired by biological phenomenons, which can automatically decouple motor groups for better robustness and generalization.
- We apply the DEMOS algorithm to both a quadruped robot and a humanoid robot and demonstrate that DEMOS can learn a robust policy against motor malfunctions.
- We demonstrate that the learned decentralized policy can be directly transferred to new tasks on the humanoid robot.

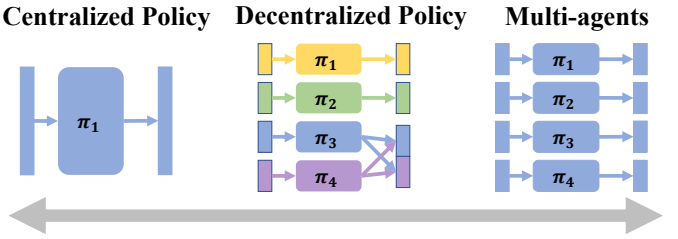


Fig. 2: Single neural network policy is highly centralized and task-specific while manually dividing the robot into multi-agents may disrupt useful coordination. **DEMOS** partially decouples modules and keeps essential connections during skill learning.

II. RELATED WORKS

A. Reinforcement Learning for Legged Robots

Legged robots have the potential to traverse various terrains that are inaccessible by wheeled robots. Quadruped robots can learn to locomote on diverse terrains through reinforcement learning with manually designed rewards. Additionally, quadruped robots can generate more diverse and natural gaits through imitation learning on animal mocap data, as demonstrated by [12], [13], [14]. To perform additional manipulation tasks, a robotic arm can be attached to the quadruped robot. Ma et al. used reinforcement learning for the locomotion task and a model-based controller for robotic arm manipulation [15]. Meanwhile, Fu et al. used reinforcement learning to jointly optimize the control of the quadruped and attached arm [4]. Aside from quadruped robots, reinforcement learning with hand-designed rewards is also applied to biped and humanoid robots, resulting in a wide variety of robust gaits [6], [16].

B. Decentralized Robot Motor Control

The most common reinforcement learning controller is a single centralized neural network with concatenated observation input. This centralized policy is highly specific to the training task. Moreover, all robot modules are coupled together and can not function independently.

To address these limitations, several works have adopted a decentralized approach that partitions the robot into modules and utilizes multiple policies to control different modules [17], [18]. For instance, Ma et al. [15] separate the quadruped-arm robot into the locomotion part and the manipulation part, controlling them via a combination of data-driven and model-based methods. Decentralized control can be viewed as a multi-agent collaboration problem [9], [19], where each module is treated as an agent that observes local information and controls its own motors. Huang et al. [20] propose the most extreme form of decentralization, where each joint is treated as a separate module, and only receives information from neighboring joints. However, all the above-mentioned decentralized control methods typically require manual partitioning before training, which can affect the overall performance. In contrast, our method automatically decouples motor groups while still maintaining the essential connections between groups, which are necessary for whole-body performance, thus avoiding any compromise on overall performance. A comparison of our method with other approaches can be found in Figure 2.

III. PRELIMINARY

Instead of controlling the robot with a single neural network policy, we divide the robot into n potentially decoupled branches and control them in a decentralized way. This setting can be constructed under the decentralized partially observable Markov decision processes (Dec-POMDP) [21] defined by $\langle \mathbb{D}, \mathbb{S}, \mathbb{A}, \mathbb{O}, O, P, R, \gamma \rangle$. $\mathbb{D}$ is the set of n branches and $\mathbb{S}$ is the state space. $\mathbb{O}_i$ is the set of observations available to the i^{th} branche with elements $o_i = O(s; i)$. $\mathbb{M}_i$ is the motor group on i^{th} branch. Straightforwardly, the global observation space $\mathbb{O}$ and global action space $\mathbb{A}$ are the aggregation of branch's sub-spaces:

$$\mathbb{O} = \mathbb{O}_1 \cup \mathbb{O}_2 \cup ... \cup \mathbb{O}_n \quad \mathbb{A} = \mathbb{M}_1 \cup \mathbb{M}_2 \cup ... \cup \mathbb{M}_n \tag{1}$$

Specifically in our setting, the i^{th} branch holds a policy π_{θ_i} which **inputs local observations o_i but outputs global actions a_i**. This indicates that each branch's local observations can contribute to whole-body actions, and the final action a is the sum of all branch actions:

$$a = a_1 + a_2 + ... + a_n. \tag{2}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

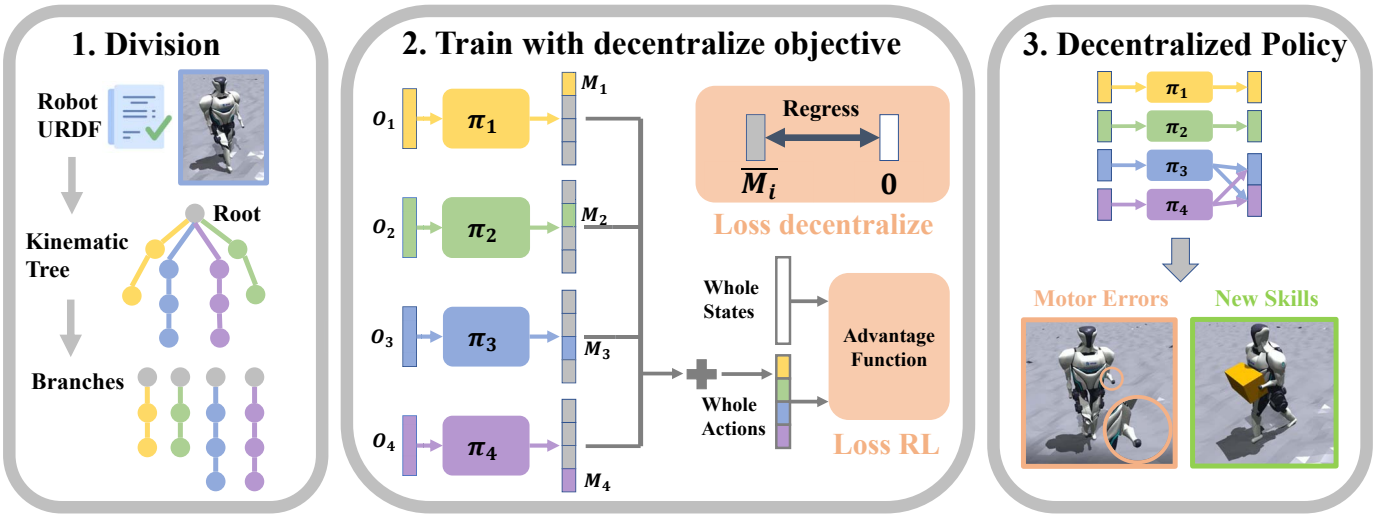


Fig. 3: Three steps of the decentralized motor skill learning pipeline: 1) Divide the robot into branches according to the kinematic tree described in its URDF. 2) O_i and M_i are local observations and actions for branch B_i. Actions out of M_i (colored grey) are encouraged to maintain zero. 3) Decouple branches with weak connections and obtain decentralized policies.

$P(s'|s,a)$ denotes the transition probability from s to s' given the whole joint action a. $R(s,a)$ denotes the shared reward function and γ is the discount factor. All branches cooperate with each other to maximize the discounted accumulated reward:

$$J(\theta_1,\theta_2,..,\theta_n) = \mathbb{E}_{a^t,s^t}\left[\sum_t \gamma^t R(s^t,a^t)\right] \tag{3}$$

After initialization, all branches remain coupled with each other since all local observations contribute to global actions. During the training procedure, our proposed method discourages local observations o_i from contributing to motors outside of M_i, and only keeps essential connections between modules. The detailed pipeline can be found in Section IV or Figure 3.

Our setting differs significantly from the settings that treat decentralized control problems as multi-agent problems. In the multi-agent setting, each agent observes locally and only controls its own motor group, which may disrupt coordination between motor groups. Moreover, since different modules are not homogeneous agents (i.e. agents have different observation spaces), some important technical designs such as policy parameter sharing [22], [23] can not be adopted.

IV. METHODS

In this section, we introduce the proposed **De**centralized **mo**tor **s**kill learning (**DEMOS**) algorithm for complex robot systems. Our pipeline can be summarized in the following steps:

1) Pre-training stage (Sec. IV-A): Divide the robot into potentially decoupled modules according to the URDF file.
2) Training stage (Sec. IV-B): Introduce a decentralized control objective to diminish the connections between modules without sacrificing performance.
3) Post-training stage (Sec. IV-C): Obtain decentralized policy by decoupling branches with weak connections and keeping the essential connections.

The overall algorithm can be found in algo. 1.

A. Division Rules

We begin by constructing a kinematic tree from the corresponding Universal Robot Description Format file (URDF). In this context, a branch refers to a path within the tree that originates from the root node and terminates at a leaf node. Multiple branches are present in every tree structure. Conceptually, joints and links located on the same branch of the tree are inherently interconnected in a sequential manner, thereby directly influencing their kinematics and dynamics. On the other hand, joints located on distinct branches of the tree exhibit relatively weak connections, as their forces or torques are conveyed through the root node, and these influences can be reflected in root velocity, acceleration, and other related parameters.

Based on the aforementioned analysis, the robot components are categorized into n branches with each branch originating from the root node and extending to a leaf node. Joints that belong to the same branch form a group or branch denoted as B_i. For instance, in the case of a humanoid robot with four limbs, there will be four branches. It's worth mentioning that branches may overlap and the same joint or link can exist in multiple branches . This process is illustrated in Figure 3 (1).

To represent the local information related to branch B_i, we employ the notation $\mathbb{O}_i$. This encompasses the local joint pose, joint velocity, root projected gravity, and central periodic clock signal. Similarly, $\mathbb{M}_i$ represents the local motor situated on branch B_i. Consequently, the global observation space $\mathbb{O}$ and global action space $\mathbb{A}$ can be obtained using Equation 1.Then we initiate policies $\pi_{\theta_i}(1 \le i \le n)$ for branch B_i, which **input local observation o_i and output global action a_i**. All branches are coupled with each other after initialization since all branches contribute to global actions.

B. Decentralized Control Objective

During the training process, in addition to the reinforcement learning (RL) objective, we also introduce a decentralized

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

control objective that encourages each branch to affect other branches as little as possible. We denote motors not on branch B_i as a complementary set $\overline{\mathbb{M}_i}$, and we have the following relationships:

$$\mathbb{A} = \mathbb{M}_i \cup \overline{\mathbb{M}_i},\ 1 \leq i \leq n \tag{4}$$

We minimize influences between branches through regressing actions in $\overline{\mathbb{M}_i}$ to 0 through L-P norm. Formally, for policies with global outputs and a sampled transition batch D consists of tuples $(o_1, ..., o_n, s, a)$:

$$\mathcal{J}_{de} = -\frac{1}{|D|} \sum_{o_i \sim D} \left[\sum_{m \in \overline{\mathbb{M}_i}} (\pi_{\theta_i}(o_i)[m] - 0)^p \right]^{1/p} \tag{5}$$

m is the motor sampled from $\overline{\mathbb{M}_i}$ and $\pi_{\theta_i}(o_i)[m]$ stands for one dimensional action that controls motor m. s represents all state information in this paper, including all local joint states and root states.

The reinforcement learning objective can be written as follows:

$$\begin{aligned} \mu &= \sum_{i=1}^{n} \pi_{\theta_i}(o_i), \quad a \sim \mathcal{N}(\mu, \sigma^2) \\ \mathcal{J}_{RL} &= \frac{1}{|D|} \sum_{(o_1,..,o_n,s,a) \sim D} \log P(a|s) \cdot A(s,a) \end{aligned} \tag{6}$$

Here $\mathcal{N}(\mu, \sigma^2)$ stands for a diagonal Gaussian distribution with mean μ and standard deviation σ. $A(s, a)$ stands for the advantage function.

Finally, the overall objective function is composed of both the RL objective and the decentralized control objective with learnable parameters $\theta_1, \theta_2, ..., \theta_n, \sigma$. The proportion of the two objectives is controlled by a hyper-parameter λ:

$$\mathcal{J}(\theta_1, \theta_2 ..., \theta_n, \sigma) = \mathcal{J}_{RL} + \lambda \mathcal{J}_{de} \tag{7}$$

However, these two objectives may conflict with each other. Reinforcing learning may encourage branches to cooperate with each other to achieve higher performance, while the decentralized control objective discourages such cooperation. In practice, a small value of lambda, such as 0.01, is typically chosen. The primary objective remains to achieve high performance in the training task, while simultaneously minimizing the connection between branches without compromising performance.

C. Decentralized Policy

After the learning process finishes, we can evaluate how much contribution local observations o_i have on B_j's motors. Formally, for branches B_i and $B_j (1 \leq i \leq n, 1 \leq j \leq n)$, we calculate the connection strength C_{ij} with L-P norm:

$$C_{ij} = \frac{1}{|D|} \sum_{o_i \sim D} \left[\sum_{m \in \mathbb{M}_j} (\pi_{\theta_i}(o_i)[m] - 0)^p \right]^{1/p} \tag{8}$$

Specifically, C_{jj} represents B_j's contributions to itself. If connections between B_i and B_j are essential to overall performance, the reinforcement learning objective will keep C_{ij}/C_{jj} at high levels. If the relative connection strength C_{ij}/C_{jj} is smaller than a small threshold: $C_{ij}/C_{jj} < \eta$, we can remove the connection between B_i and B_j:

$$\pi_{\theta_i}(o_i)[m] = 0, \quad m \in \mathbb{M}_j, \quad if\ \frac{C_{ij}}{C_{jj}} < \eta \tag{9}$$

In addition, it is possible to implement decentralization at the motor level, particularly in cases where branches overlap. From another perspective, each motor receives influences from all branches. We can calculate the connection strength S_{ij} between branch B_i and an individual motor m_j, and then remove weak connections:

$$S_{ij} = \frac{1}{|D|} \sum_{o_i \sim D} |\pi_{\theta_i}(o_i)[m_j]| \tag{10}$$

$$\pi_{\theta_i}(o_i)[m] = 0, \quad m \in \mathbb{M}_j, \quad if\ \frac{S_{ij}}{S_j} < \eta' \tag{11}$$

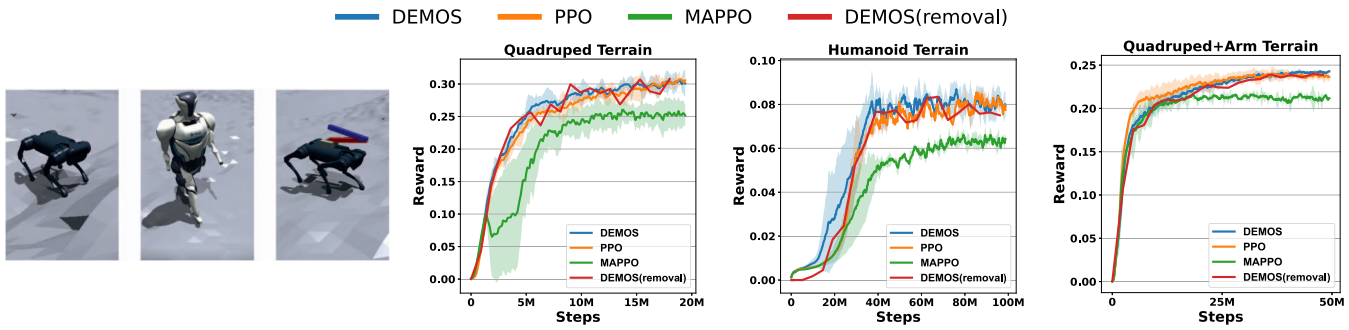
Here, S_j represents the connection strength from the motor's own branches: $S_j = \sum_{m_j \in B_i} S_{ij}$.

After we perform decentralization, only essential connections remained while others are removed. Our approach ensures that all policies are optimized jointly and coordinated under the same central clock. As a result, each local observation has the potential to contribute to global motor actions and all policies can work together under equivalence by preserving essential connections. For instance, if a particular branch detects a collision, it can transmit signals to other branches and modify their original actions to prepare for the collision.

Algorithm 1 **De**centralized **Mo**tor **S**kill Learning (**DEMOS**)

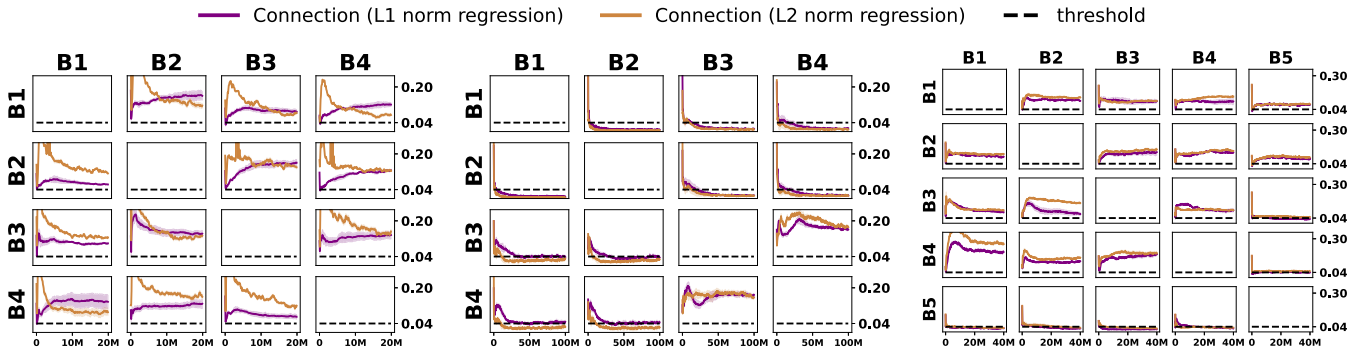
```
# Pre-training stage
Divide robot into n branches B_1 ~ B_n with observation space O_1 ~ O_n and motor sets M_1 ~ M_n. Global observation space O and global action space A are defined by equation 1.
Initialize policies π_θ1 ~ π_θn with local observations o_1 ~ o_n and global actions a_1 ~ a_n.
# Training stage
Input: Number of episodes n_e, steps of policy updates per epoch n_p, policies π_θ1 ~ π_θn, replay buffer D, proportion coefficient λ.
for k in {0, 1, 2, ...,n_e} do
  Interact with the environment a = Σ_{i=1}^{n} a_n, full fill the buffer D with (o_1, ...o_n, s, a).
  for i in {0, 1, 2, ...,n_p} do
    Update θ_1 ~ θ_n with objective J = J_RL + λJ_de (7)
  end for
end for
# Post-training stage
for 1 ≤ i ≤ n, 1 ≤ j ≤ n, i ≠ j do
  Calculate connection strength C_ij between branches B_i and B_j with equation 8.
  if C_ij < η then
    Decouple influences from B_i to B_j with equation 11.
  end if
end for
Obtain decentralized motor policy π : {π_θ1, π_θ2, ..., π_θn}.
```

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.



(a) Original locomotion tasks on terrains (b) Quadruped locomotion (c) Humanoid locomotion (d) Quadruped+Arm locomotion

Fig. 4: Locomotion tasks on challenging terrain for various types of robot. X-axis stands for environment interaction steps and Y-axis stands for the average reward. DEMOS(removal) denote performance after removing the weak connections.



(a) Quadruped branch connections (b) Humanoid branch connections (c) Quadruped+Arm branch connections

Fig. 5: Visualization of normalized connection strength between branches. X-axis stands for the environment interaction step and Y-axis stands for the relative connection strength C_{ij}/C_{jj}. Subfigure (i, j) stands for influence from B_i to B_j during the training, and the threshold is set to 0.04. Connections in subfigure (i, i) are always equal to 1 and we do not plot it.

V. EXPERIMENTS

In this section, we present experimental results aimed at addressing the following questions:

1) Does decentralized policy sacrifice performance on the original learning task when compared to a centralized policy? (Sec. V-B)
2) Is decentralized policy more robust to local motor malfunctions? (Sec. V-C)
3) How can we transfer a decentralized policy to new tasks and rapidly acquire new skills? (Sec. V-D)

A. Setups and baselines

We choose different types of robots in our experiments, which include a quadruped robot with 12 motors, a humanoid robot with 16 motors, and a quadruped with an attached arm, powered by 15 motors. The complete set of observation dimensions can be found in Table I.Our neural network policy output the PD target of all motors and runs at 50 Hz, and the low-level PD controllers run at 1000Hz with $k_p = 40$ and $k_d = 1$.

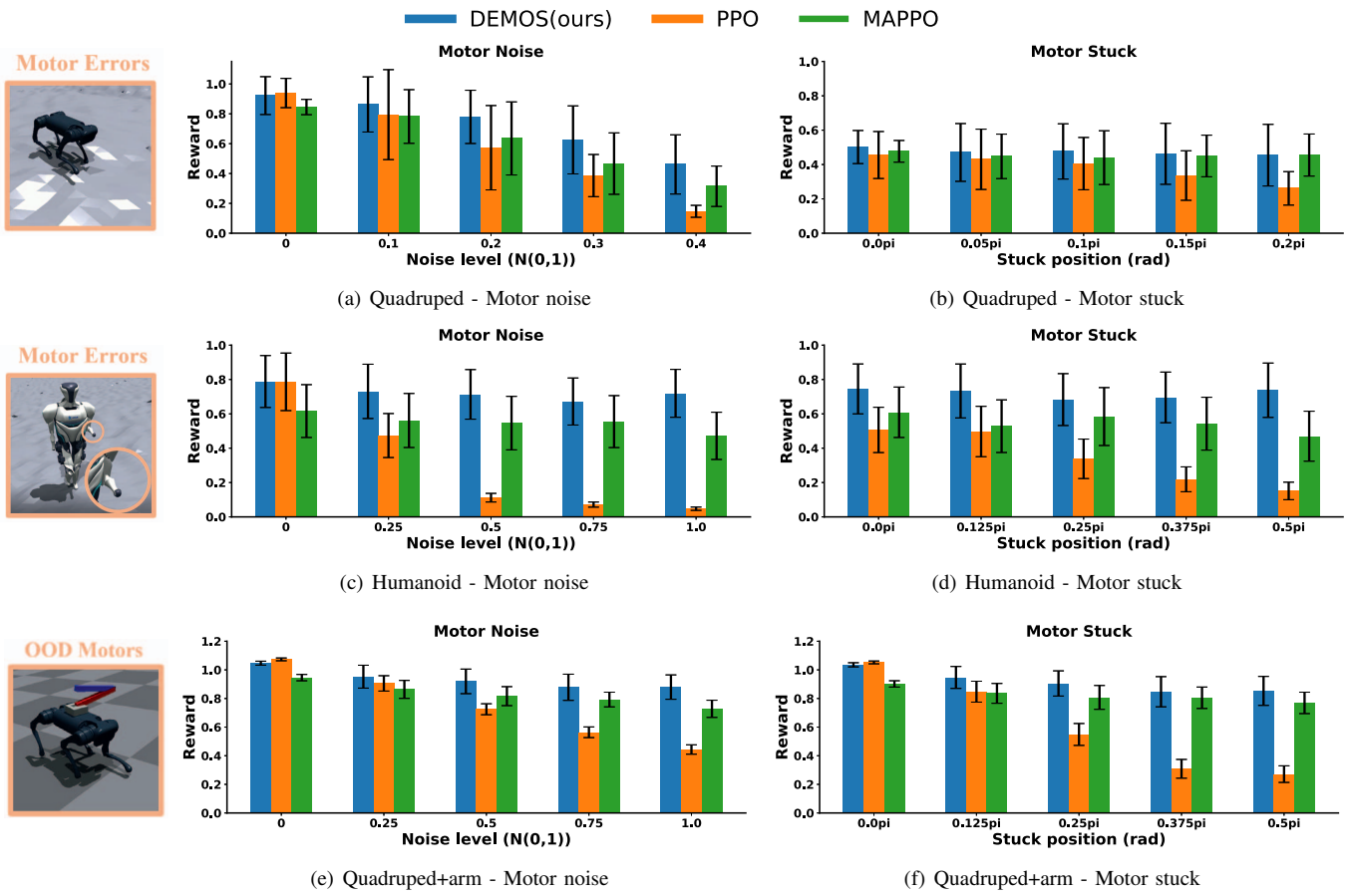
The proposed decentralized motor skill learning pipeline is compatible with various on-policy or off-policy reinforcement learning algorithms and we choose PPO [24] as our backbone RL algorithm. However, training robots with manually designed rewards can lead to unnatural behaviors. To mitigate this, we integrated animal and human motion capture (Mocap) data into the learning process. This Mocap data was used as a form of regularization to produce natural robot behaviors. The overall reward r is the combination of task reward r_{task} and style reward r_{style} similar to Deepmimic [13]: $r = r_{task} + r_{style}$.

Besides our proposed **De**centralized **mo**tor **s**kill learning (**DEMOS**) Method, we compared our algorithms to the cen-

	Quadruped	Humanoid	Quadruped +Arm
Projection of gravity	3	3	3
Clock inputs	2	2	2
Joint positions	12	16	15
Joint velocities	12	16	15
Last actions	12	16	15
Overall	41	53	50

TABLE I: Observation space for the various types of robots used in our experiments. The projection of gravity (in robot root coordinates) reflects the orientation of the root link. Clock inputs are periodic sin and cos signals used for periodic walking.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.



(a) Quadruped - Motor noise

(b) Quadruped - Motor stuck

(c) Humanoid - Motor noise

(d) Humanoid - Motor stuck

(e) Quadruped+arm - Motor noise

(f) Quadruped+arm - Motor stuck

Fig. 6: Evaluations on various types of robots with motor observation noises and motor stuck errors. We can find that local motor errors have less influence on decentralized policy since DEMOS minimizes the influences between branches.

tralized approach and the multi-agent approach:

- **PPO [24]**: The most common approach in the previous works, typically using a single neural network as a centralized controller.
- **MAPPO [25]**: A popular multi-agent reinforcement learning algorithm shown to have state-of-the-art performance in cooperation games. Under this setting, each module has its own actor that observes locally and acts locally.

Our simulated environment is built with Isaac Gym [26]. All the experiments are run over 5 random seeds.

B. Performance on original task

In this part, we want to demonstrate that our proposed DEMOS algorithm will not sacrifice performance on original learning tasks compared to the centralized approach. The quadruped robot and humanoid robot are required to walk on various types of terrains such as up-slope, down-slope, and random up-down terrains. Before the training stage, the quadruped robot is divided into 4 branches $\{B_1, B_2, B_3, B_4\}$ which stand for $\{$*Left-forward branch, Right-forward branch, Left-behind branch, Right-behind branch*$\}$. And humanoids robot is divided into 4 branches $\{B_1, B_2, B_3, B_4\}$ stand for $\{$*Left-arm branch, Right-arm branch, Left-leg branch, Right-leg branch*$\}$. Quadruped with arm is divided into 5 branches $\{B_1, B_2, B_3, B_4, B_5\}$ which stand for $\{$*Left-forward branch, Right-forward branch, Left-behind branch, Right-behind branch, Arm branch*$\}$.

Comparisons can be found in Figure 4. We notice that our decentralized policy achieves comparable performance with a single centralized PPO policy. However, the multi-agent approach reduces the performance since each agent only contributes to its own motors, making the cooperation between modules more difficult. Moreover, removing weak connections did not sacrifice performance, denoted as DEMOS(removal).

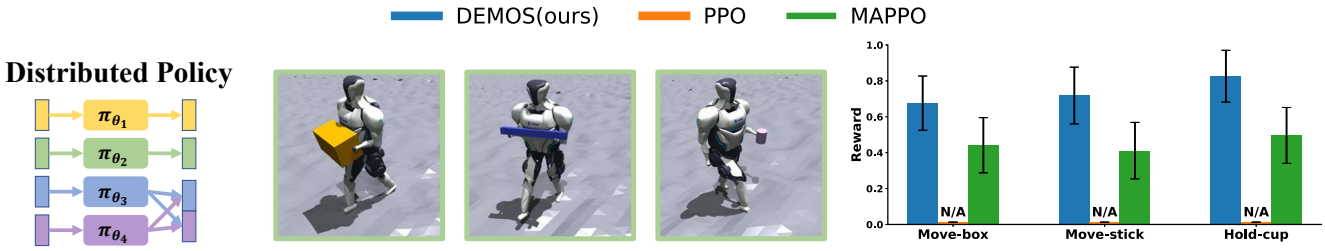
We also visualize the relative connection strength C_{ij}/C_{jj} between modules during the DEMOS training in Figure 5, calculated by equation 8. For the quadruped locomotion task, we find that all connections between modules are relatively high, and DEMOS maintains all connections between the four branches. This means that the four branches must work in cooperation with each other to achieve optimal performance. On the other hand, in the humanoid locomotion tasks, only B_3 (*left-foot branch*) and B_4 (*right-foot branch*) preserve relatively high connections. DEMOS has reduced all other connections without sacrificing performance. For the quadruped robot with an arm, we can remove the connection from the arm to the legs and preserve others.

C. Local motor malfunctions

In this part, we demonstrate that DEMOS learns policies that are more robust to motor malfunctions. DEMOS achieves

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.



(a) New tasks

(b) Performance on new tasks

Fig. 7: Transfer part of the decentralized policy to new tasks: move-box task, move-stick task, hold-cup task.

this by encouraging each branch to have minimal influence on others. As a result, errors on a single branch may also impact other branches less. Additionally, the DEMOS pipeline may decouple the connections between two branches, which means that errors in one branch will not affect the other branch at all.

We consider 2 kinds of malfunctions that widely exist in motors:

- **Motor noise**: a damaged motor may have a large motor observation noise and lead to out-of-distribution local observations.
- **Motor stuck**: a motor may lose the power to move or be stuck at a certain degree by external force or obstacle. In this setting, we assume that one motor is stuck at a certain degree and cannot move.

We assume that one motor on the quadruped left-forward leg and one motor on the humanoid left arm suffer these 2 types of motor malfunctions with different levels.

Experimental results are shown in Figure 6. The decentralized policy is more robust against local motor malfunctions since DEMOS minimizes the connections between branches, resulting in less performance drop for quadruped robots compared to the centralized policy. Furthermore, in the case of humanoid robots, motor malfunctions in the left arm have almost no effect on locomotion performance. This is reasonable since DEMOS has decoupled the arm branch from the leg branch, thus preventing errors from propagating to the legs. However, the centralized PPO policy experiences a rapid drop in performance as local motor error can impact global motors through a centralized neural network policy.

It is worth mentioning that decentralized policies are incapable of resisting large motor errors in the humanoid leg. This is reasonable since even humans struggle to walk with a broken leg. However, similar to human beings, humanoid robots are now capable of walking with a broken arm, thanks to the decentralized policies.

D. Transfer decentralized policy to new tasks

In this section, we demonstrate that it is possible to transfer subsets of the decentralized policy to new tasks and acquire new skills rapidly.

In the original task, we divide the humanoid robot into 4 branches $\{B_1, B_2, B_3, B_4\}$, and initialize policies $\pi_{\theta_1}, \pi_{\theta_2}, \pi_{\theta_3}, \pi_{\theta_4}$ respectively. During the training stage, branches are decoupled except for the *left-leg branch* and *right-leg branch*. This means we can divide policies into 3 subsets:

$$\{\pi_{\theta_1}\}, \{\pi_{\theta_2}\}, \{\pi_{\theta_3}, \pi_{\theta_4}\} \tag{12}$$

Connections between these subsets have vanished, and each subset can function independently. Specifically, $\{\pi_{\theta_1}\}$ controls left arm to swing naturally, $\{\pi_{\theta_2}\}$ controls right arm to swing naturally and $\{\pi_{\theta_3}, \pi_{\theta_4}\}$ jointly controls 2 legs to walk on terrains.

When faced with new tasks, we can reuse certain sets of sub-policies in 2 approaches: (1) Combination lead to new skills. For instance, in the move-box task, we can utilize the leg-walking policy set $\{\pi_{\theta_3}, \pi_{\theta_4}\}$ from the previous task. To complete the move-box task, we only need to substitute the original arm swing policy $\pi_{\theta_1}, \pi_{\theta_2}$ with the grasp policy π_{box}. The new sub-policy π_{box} can be either a model-based policy or a pre-trained neural network policy. (2) Provide a great starting point to learn new tasks. In the move-box case, we can initiate a new policy for the arm while retaining or freezing the policy parameters in the set $\{\pi_{\theta_3}, \pi_{\theta_4}\}$. Utilizing an agent who already possesses the knowledge of walking can serve as a strong foundation for the move-box task.

In our experiments, we combine pre-designed model-based arm policies and certain previously learned policies to handle new tasks:

- Move-box skill: $\{\pi_{box}\} + \{\pi_{\theta_3}, \pi_{\theta_4}\}$
- Move-stick skill: $\{\pi_{stick}\} + \{\pi_{\theta_3}, \pi_{\theta_4}\}$
- Hold-cup skill: $\{\pi_{cup}\} + \{\pi_{\theta_2}\} + \{\pi_{\theta_3}, \pi_{\theta_4}\}$

Here π_{box} is a model-based controller for 2 arms to hold a box. π_{stick} is a controller for 2 arms to hold a long stick. π_{cup} is a controller for a single arm to hold a cup.

Experimental results are shown in Figure 7. The decentralized policy can be successfully transferred to new tasks and combined into new skills. However, a typical centralized PPO policy is highly task-specific with all modules coupled together, thus can not transfer to new tasks. MAPPO policies lose performance when transferred to new tasks since some important cooperation between modules has been destroyed.

VI. CONCLUSIONS

In this work, we propose decentralized Motor skill learning (DEMOS), a decentralized approach for complex robotic systems control. Instead of one typical neural network policy that is highly centralized, we leverage multiple local-input-global-output policies to control the robot and decouple modules with

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

weak connections automatically during training. Experiment results demonstrated that DEMOS are effective for a wide range of robot types, including quadrupeds, humanoids, and quadrupeds with arms, etc. However, there also exist some limitations in our work. First, in the face of robots with largely overlapped branch structures, we may need other division rules in the pre-training stage to better decentralize the control policy. Second, in the face of new tasks acting significantly different from the original tasks, learned sub-skills may not contribute to new tasks. These are all possible research directions in the future.

APPENDIX

A. Implementation details

Our simulation environments are built in IsaacGym [26] and we implement our DEMOS algorithm and baselines in Pytorch based on opensource codebase https://github.com/leggedrobotics/legged_gym [27].

Hyperparameters of the backbone PPO algorithm can be found in table II.

Parameters	Value
Number of Environments	4096
Learning epochs	5
Steps per Environment	24
Minibatch Size	24576
Episode length	20 seconds
Discount Factor	0.99
Generalised Advantage Estimation(GAE)	0.95
PPO clip	0.2
Entropy coefficient	0.005
Desired KL	0.01
Learning Rate	5e-4
Weight decay	0.01

TABLE II: Hyperparameters of backbone PPO algorithm.

REFERENCES

[1] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[3] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[4] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: learning a unified policy for manipulation and locomotion," *arXiv preprint arXiv:2210.10044*, 2022.

[5] K. Kaneko, H. Kaminaga, T. Sakaguchi, S. Kajita, M. Morisawa, I. Kumagai, and F. Kanehiro, "Humanoid robot hrp-5p: An electrically actuated humanoid robot with high-power and wide-range joints," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1431–1438, 2019.

[6] J. Siekmann, Y. Godse, A. Fern, and J. Hurst, "Sim-to-real learning of all common bipedal gaits via periodic reward composition," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7309–7315.

[7] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous robots*, vol. 40, pp. 429–455, 2016.

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

[9] C. S. de Witt, B. Peng, P.-A. Kamienny, P. Torr, W. Böhmer, and S. Whiteson, "Deep multi-agent reinforcement learning for decentralized continuous cooperative control," *arXiv preprint arXiv:2003.06709*, vol. 19, 2020.

[10] R. Leiras, J. M. Cregg, and O. Kiehn, "Brainstem circuits for locomotion," *Annual review of neuroscience*, vol. 45, pp. 63–85, 2022.

[11] G. W. Lindsay, "Attention in psychology, neuroscience, and machine learning," *Frontiers in computational neuroscience*, vol. 14, p. 29, 2020.

[12] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 25–32.

[13] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[14] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

[15] Y. Ma, F. Farshidian, T. Miki, J. Lee, and M. Hutter, "Combining learning-based locomotion policy with model-based manipulation for legged mobile manipulators," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2377–2384, 2022.

[16] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Learning humanoid locomotion with transformers," *arXiv preprint arXiv:2303.03381*, 2023.

[17] J. Whitman, M. Travers, and H. Choset, "Learning modular robot control policies," *arXiv preprint arXiv:2105.10049*, 2021.

[18] T. Wang, R. Liao, J. Ba, and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," in *Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada*, vol. 30, 2018.

[19] B. Peng, T. Rashid, C. Schroeder de Witt, P.-A. Kamienny, P. Torr, W. Böhmer, and S. Whiteson, "Facmac: Factored multi-agent centralised policy gradients," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 208–12 221, 2021.

[20] W. Huang, I. Mordatch, and D. Pathak, "One policy to control them all: Shared modular policies for agent-agnostic control," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4455–4464.

[21] F. A. Oliehoek and C. Amato, *A concise introduction to decentralized POMDPs*. Springer, 2016.

[22] J. K. Terry, N. Grammel, A. Hari, L. Santos, and B. Black, "Revisiting parameter sharing in multi-agent deep reinforcement learning," *arXiv preprint arXiv:2005.13625*, 2020.

[23] F. Christianos, G. Papoudakis, M. A. Rahman, and S. V. Albrecht, "Scaling multi-agent reinforcement learning with selective parameter sharing," in *International Conference on Machine Learning*. PMLR, 2021, pp. 1989–1998.

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

[25] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative, multi-agent games," *arXiv preprint arXiv:2103.01955*, 2021.

[26] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[27] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.