DynSyn: Dynamical Synergistic Representation for Efficient Learning and Control in Overactuated Embodied Systems

Kaibo He
School of Aerospace Engineering
Tsinghua University
China
beanpow@gmail.com

Chenhui Zuo
School of Aerospace Engineering
Tsinghua University
China
zuoch22@mails.tsinghua.edu.cn

Chengtian Ma
School of Aerospace Engineering
Tsinghua University
China
mct23@mails.tsinghua.edu.cn

Yanan Sui
School of Aerospace Engineering
Tsinghua University
China
ysui@tsinghua.edu.cn

Abstract: Learning an effective policy to control high-dimensional, overactuated systems is a significant challenge for deep reinforcement learning algorithms. Such control scenarios are often observed in the neural control of vertebrate musculoskeletal systems. The study of these control mechanisms will provide insights into the control of high-dimensional, overactuated systems. The coordination of actuators, known as muscle synergies in neuromechanics, is considered a presumptive mechanism that simplifies the generation of motor commands. The dynamical structure of a system is the basis of its function, allowing us to derive a synergistic representation of actuators. Motivated by this theory, we propose the *Dynamical Synergistic Representation* (*DynSyn*) algorithm. *DynSyn* aims to generate synergistic representations from dynamical structures and perform task-specific, state-dependent adaptation to the representations to improve motor control. We demonstrate *DynSyn*'s efficiency across various tasks involving different musculoskeletal models, achieving state-of-the-art sample efficiency and robustness compared to baseline algorithms. *DynSyn* generates interpretable synergistic representations that capture the essential features of dynamical structures and demonstrates generalizability across diverse motor tasks.

Keywords: Reinforcement Learning, Motor Control, Overactuated Systems, Synergistic Representation

1 Introduction

In the evolution of embodied intelligence, researchers have used reinforcement learning (RL) algorithms to train controllers across diverse robotic platforms, yielding notable advancements in motor control. These agents can acquire robust and generalizable policies through iterative trial and error within large-scale simulations, subsequently deploying them onto real-world robots via sim-to-real methodologies [1, 2, 3]. Overactuation and redundancy can often enhance the safety and robustness of embodied intelligent systems, mitigating the risk of sudden control failures [4, 5, 6]. However, overactuation will increase the complexity of the controlled object, particularly by enlarging the

CoRL 2024 Workshop on Whole-Body Control and Bimanual Manipulation (CoRL 2024 WCBM).

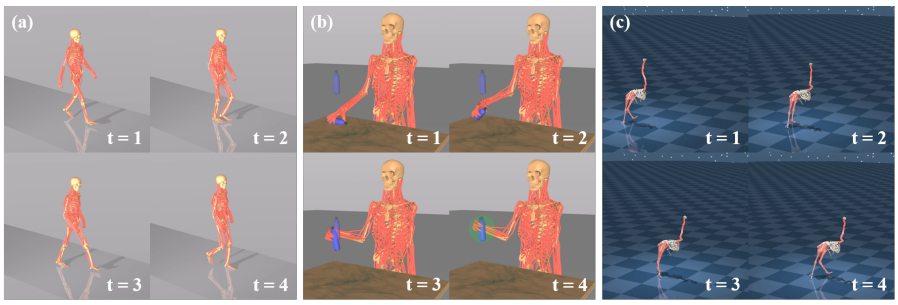


Figure 1: **Motor behaviors of overactuated musculoskeletal systems acquired by *DynSyn*.** (a) Gait of MS-HUMAN-700 model. (b) Manipulation of MS-HUMAN-700 Arm model. (c) Locomotion of *Ostrich* model. See our project website at https://sites.google.com/view/dynsyn.

dimensionality of the action space, making it difficult for the deep RL controllers to achieve motor control.

A common example of overactuated embodied systems is musculoskeletal systems in nature. In contrast to existing DRL agents, the motor control intelligence of vertebrates can control musculoskeletal systems through the central nervous system, exhibiting the ability to generalize across a variety of motor tasks while maintaining stability even under large disturbances, such as external force interference and drastic changes of environmental parameters. Exploring musculoskeletal motion control can help address the control challenges posed by high-dimensional, overactuated systems, thereby advancing our progress towards embodied intelligence. Nevertheless, a musculoskeletal model of human possesses characteristics that pose significant challenges for motor control by a RL agent. Firstly, the muscle control parameter space is high-dimensional, where over 600 skeletal muscles control hundreds of joints [7]. Secondly, the system is overactuated, as multiple muscles actuate one joint and multiple joints may be affected by one muscle [8]. Thirdly, the dynamic of neuro-muscular actuators is non-linear and inconstant [9, 10], and these actuators can only generate tension and no reverse forces.

How does motor control in vertebrates effectively address the challenge of redundant actuation? In neuroscience, there is a hypothesis known as muscle synergies. It proposes that coordinated recruitment of groups of muscles serves as a modular framework for biological motor control, simplifying the generation of motor commands. Studies have demonstrated that various motor behaviors can be reconstructed with high fidelity using a basic set of coordinated muscle activity patterns endowed with different weights [11, 12]. We conceptualize muscle synergies as representations of actuators' control strategy. According to our hypothesis, these representations should depend on the physical structure of the controlled system, as this structure determines the characteristics of actuators. These representations should also exhibit generalizability across diverse tasks and conditions, reflecting commonalities within motor systems. Furthermore, we assume that these synergies can be adaptively fine-tuned to suit specific demands of each movement and state, as the actual working conditions of actuators are not inherently identical when performing different movements. We explore to discover synergistic representations from dynamical structures and embed them into deep RL methods to achieve efficiency and generalization in physiological motor control tasks.

In this work, we propose *DynSyn*, a deep reinforcement learning algorithm that is capable of generating interpretable synergistic representations of dynamical structures and performing task-specific, state-dependent adaptation to the representations. The generation process is driven by random perturbations. A stable and interpretable representation of dynamical synergies can be obtained. Embedding the representation into the learning process improve the efficiency of the agent when learning motor control policies demonstrated in Figure 1.

Our work mainly achieves following contributions: **(1)** We propose a method to generate synergistic representations from dynamical structure (for the first time), and a learning algorithm for state-dependent, task-specific adaptation in motion control tasks. **(2)** Experiments show that this representation generation method can obtain stable and interpretable representations of different overactuated systems. The representations can be generalized across different tasks. **(3)** We demonstrate that our representation-embedded learning algorithm can make the training process more efficient.

(4) Our algorithm achieves control of ultra-high-dimensional musculoskeletal models, while other algorithms fail. *DynSyn* advances simulation control approaches in biomechanics, neuroscience and motor control communities.

2 Related Work

Over-redundant actuation control. Overactuated systems, often observed in the motor control of vertebrates, such as musculoskeletal systems, present a challenge for controllers trained by RL algorithms. A low-dimensional representation can be used to evaluate the quality of the control [13]. A series of attempts aimed on tackling the problem of learning control policies for locomotion and manipulation tasks with musculoskeletal models [14, 15, 16]. Leading solutions of these challenges include heavy curriculum training approaches, with reward shaping or demonstration imitation [17]. Recently, a hierarchical reinforcement learning algorithm combined with imitation learning was applied to a 346-muscle musculoskeletal model [18]. Generative models like variational autoencoders were utilized to control this musculoskeletal model to generate diverse behaviors [19, 20]. In addition to human models, a musculoskeletal model of ostrich was constructed using the MuJoCo physics engine and controlled by TD4 deep reinforcement learning algorithm [21]. Recent works, such as DEP-RL [22] and Lattice [23], have shown that employing better exploration techniques in reinforcement learning can help address the problem. Multi-task learning method is used in dexterous physiological control on a human hand model [24]. Bio-inspired approaches [25, 26] have demonstrated their effectiveness in motion control by applying synergistic representations in distinct parts of musculoskeletal bodies.

Synergies for motor control in neuroscience. For typical redundant actuation systems in nature, the coordination of actuators in vertebrates' neuromusculoskeletal motor control is known as muscle synergies. This can be defined as the coordinated recruitment of groups of muscles in the spatial, temporal, or spatiotemporal domains [27]. As a long-standing theory in neurophysiology, muscle synergies is widely considered a possible approach for the central nervous system to overcome the complexity of motor control by reducing the number of independent parameters to simplify the generation of motor commands [28, 29]. Researchers can generate the representation of muscle synergies through multidimensional matrix factorization from animals including rat, frog and human [30, 11, 31, 32]. In experiments where humans perform fast reaching movements, changes of the muscle contraction patterns among various conditions can be explained with a high degree of confidence by assigning a set of synergy coefficients [12]. The methodology of learning lower-dimensional action representations is being studied in the field of general robotics as well [33, 34, 35].

Our algorithm uses DRL to train control policies on high-dimensional, overactuated systems. Compared to existing methods, our representation generation method avoids training cost in an interpretable way. To the best of our knowledge, our work is the first algorithm to discover synergistic representations from dynamical structures and successfully use them to solve the motor control problem of high-dimensional overactuated systems.

3 System Dynamical Features

In this study, we aim to generate synergistic representations of actuators based on the dynamical characteristics of overactuated systems. Overactuation is common in natural musculoskeletal systems controlled by multi-articulation and pull-only actuations, making their motion control much harder than conventional torque-controlled robots [24, 26]. In this section, we will introduce the neuro-musculoskeletal control of a full-body model as an example and outline the problem formulation.

3.1 Physiological Neuro-Musculoskeletal Control

Full body musculoskeletal model. As shown in Figure 2(a), a full body musculoskeletal model MS-HUMAN-700 [36] is used. The model has 90 rigid body segments, 206 joints, and 700 muscle-tendon units, and is implemented in the MuJoCo physics simulator [37]. The base segment of the model is

(a) (b)

Figure 2: **Musculoskeletal model.** (a) Full body model MS-HUMAN-700, red lines represent muscle-tendon units. (b) Arm model with wrist and finger joints for dexterous manipulation. The top illustrates the the skeleton structure and the bottom illustrates muscles.

pelvis, which can translate and rotate in full degrees of freedom. The body parts can interact with the environment during simulation because the mesh files of their bones are used for collision calculation. The dynamics of the human musculoskeletal system can be formulated with the Euler-Lagrangian equations as

$$M(q)\ddot{q} + c(q, \dot{q}) = J_m^T f_m(act) + J_c^T f_c + \tau_{ext} \tag{1}$$

where q represents the generalized coordinates of joints, $M(q)$ represents the mass distribution matrix, and $c(q, \dot{q})$ stands for Coriolis and gravitational forces. $f_m(act)$ is the vector representing muscle forces generated by all muscle-tendon units, and is determined by muscle activations (act). f_c is the constraint force. J_m and J_c are Jacobian matrices that map forces to the space of generalized coordinates. τ_{ext} is external torque. The input control signal of muscle-tendon units is the neural excitations, which determine muscle activations. With the employment of the Hill-type muscle model [9], the activation-contraction dynamics of muscles exhibit non-linearity and temporal delay, thereby posing challenges to neuromuscular control (see Appendix A).

In Section 5, we apply *DynSyn* to several local models of human body (such as an arm model in Figure 2(b)) and a model of ostrich [21]. These local models of human body are part of the MS-HUMAN-700 model, with slight differences in the implementation of simulation. The details of human local models and the ostrich model are presented in Section 5.1.

3.2 Problem Formulation

A motor control task of musculoskeletal models and robots can be formulated as a Markov decision process, denoted by $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, r, p, \rho_0, \gamma \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^n$ represents the continuous space of all valid states, and $\mathcal{A} \subseteq \mathbb{R}^m$ represents the continuous space of all valid actions. $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. The state transfer probability function $p(s'|s, a)$ describes the probability of an agent taking an action a to transfer from the current state s to the state s'. ρ_0 is the probability distribution of the initial state with $\sum_{s_0 \in \mathcal{S}} \rho_0(s_0) = 1$ and $\gamma \in [0, 1)$ is the discount factor. In the reinforcement learning paradigm, the goal of the agent is to optimize the policy parameter θ that maps from states to a probability distribution over actions $\pi_\theta : \mathcal{S} \to P(\mathcal{A})$. The policy seeks to maximize the discounted returns, $\pi_\theta^*(a|s) = \arg\max_\theta [\sum_{t=0}^{T} \gamma^t r(s_t, a_t)]$. The details of our action space and state space are described in Appendix B.

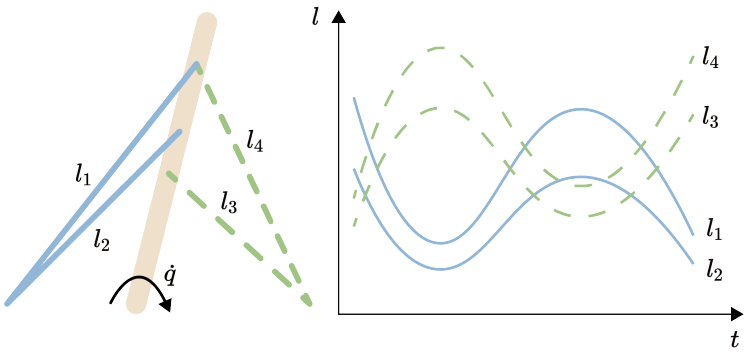


Figure 3: **Motivation of *DynSyn*.** The brown link represents a robot arm (or bone), while the blue and green lines represent the cable actuators (or muscles). By randomly controlling the joint velocity, the lengths of the four actuators are demonstrated on the right. Actuators with similar functions are categorized into the same group due to similar structures, based on the correlation of length changes.

Algorithm 1: [*DynSyn*] Dynamical Synergistic Representation

Input: model $\mathcal{M}$, total trajectory steps N_s, control frequency T, control amplitude A_c, number of groups N_g

Output: grouping bins G

Initialize trajectory buffer $\tau \leftarrow \emptyset$, $t \leftarrow 0$

`// Random trajectory generation`

while $t \leq N_s$ **do**

  if $t \bmod T = 0$ **then**

    Sample joint velocity $\dot{q}_t \sim \text{Unif}[-A_c, A_c]$

    Set model's joint velocity $\dot{q} \leftarrow \dot{q}_t$

  end

  Simulation step($\mathcal{M}$, zero action)

  $\tau \leftarrow \tau \bigcup l_t$ // Store the lengths of muscles

  $t \leftarrow t + 1$

end

`// Grouping based on the correlation`

Calculate correlation matrix R using τ with Equation 2

Grouping bins $G \leftarrow$ K-Medoids$(1 - R)$

4 Dynamical Synergistic Representation

As the action space enlarges, the sample efficiency of DRL algorithms sharply decreases. Researchers have explored various aspects of a typical example of these problems, human musculoskeletal system control, by means including refining exploration strategies [22, 23] and the utilization of hierarchical learning approaches [18]. Efforts has been made to learn synergistic action representations from trajectories in pre-training stage to expedite training, which is highly reliant on pre-training outcomes [26]. As shown in Figure 3, we observe that actuators with similar functions exhibit structural similarities. Hence, we employ a dynamics-based method which is able to rapidly generate interpretable synergistic representation. We then propose a novel algorithm to use these representations for further learning process.

4.1 Representation Generation

As illustrated in Algorithm 1, We employ a similarity-based grouping method for the dynamical synergistic representation generation. Firstly, we generate muscle length trajectories of length N_s through applying random control to the joint space of a musculoskeletal model $\mathcal{M}$. Here, the control signal is the joint velocity $\dot{q}$ of the musculoskeletal model, and this signal is sampled randomly from a uniform distribution Unif$[-A_c, A_c]$ every T time intervals. Upon obtaining the muscle

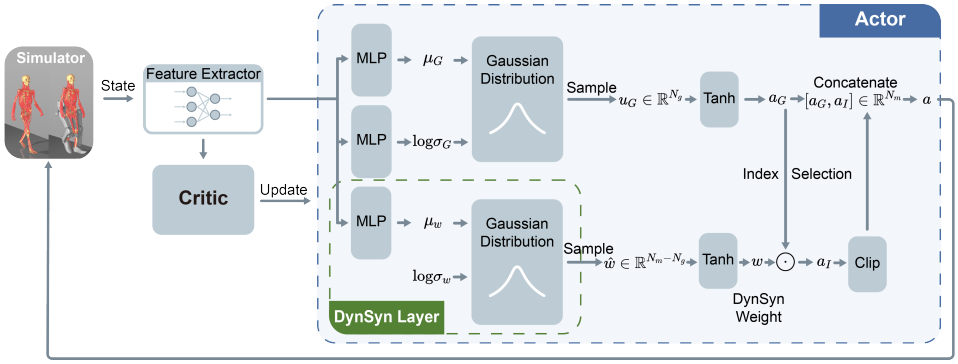


Figure 4: **Overview of *DynSyn*.** The algorithm generates a unified action a_G for each group of actuators, along with state-dependent correction weights w for each actuator on top of the unified action a_G. Separate MLPs are used to generate the parameters of two Gaussian distributions. We then sample from the Gaussian distributions and pass them through a squashing function Tanh to obtain a_G and w.

length trajectory $\tau \in \mathbb{R}^{N_s \times N_m}$, we calculate the correlation between length changes for each pair of muscles according to Equation 2. N_m is the dimension of actions (the number of muscles). Based on the correlation matrix $R \in \mathbb{R}^{N_m \times N_m}$, and a predetermined number of groups N_g, we employ the K-Medoids [38] clustering algorithm to generate the closest clustering results, forming grouping bins G.

$$R_{i,j} = \frac{1}{N} \sum_{k=0}^{N-1} S_c(\tau^i_{[\frac{N_s}{N} k : \frac{N_s}{N}(k+1)]}, \tau^j_{[\frac{N_s}{N} k : \frac{N_s}{N}(k+1)]}) \tag{2}$$

The correlation matrix is calculated by Equation 2. We divide the trajectories into N segments with respect to time and then average the similarity of each segmented trajectory. N represents the number of segmented trajectories, S_c is the cosine similarity, and i, j correspond to the i th and j th muscles, respectively. Subscript $[t_1 : t_2]$ represents the trajectory from time t_1 to time t_2.

4.2 State-dependent Representation

Using the above algorithm, functionally similar actuators will be categorized into a group, and assigned with same actions. This prevents DRL algorithms from assigning opposite actions to functionally similar actuators during the learning process, thereby enhancing effective exploration in high-dimensional action spaces. However, merely assigning same actions to all actuators within a group may result in unnatural movements. Therefore, we propose the algorithm illustrated in Figure 4. While the actuators within a group perform shared actions, state-dependent correction weights are produced for each actuator to facilitate fine-tuned adjustments.

Based on the SAC algorithm [39], our algorithm generates a unified action a_G for each group, along with state-dependent correction weights w for each actuator on top of the unified action a_G. a_G and w are written as

$$a_G = \tanh(u_G), u_G \sim \mathcal{N}(\mu_G, \sigma_G) \tag{3}$$
$$w = \tanh(\hat{w}), \hat{w} \sim \mathcal{N}(\mu_w, \sigma_w), \tag{4}$$

where μ_G, σ_G represent the mean and variance of the unified actions, μ_w is the mean of state-dependent correction weights and σ_w is the state-independent variance of the weights. By default, the first actuator in each group is assumed to have a correction weight of 1, and $N_m - N_g$ correction weights are to be determined. The final action is computed using the following equations:

$$a_I = \text{IS}(a_G) \odot \text{clip}(\kappa w, -c, c) \tag{5}$$
$$c = \min(\max(k_D t + a_D, 0), \kappa) \tag{6}$$
$$a = \text{clip}([a_G, a_I], -1, 1), \tag{7}$$

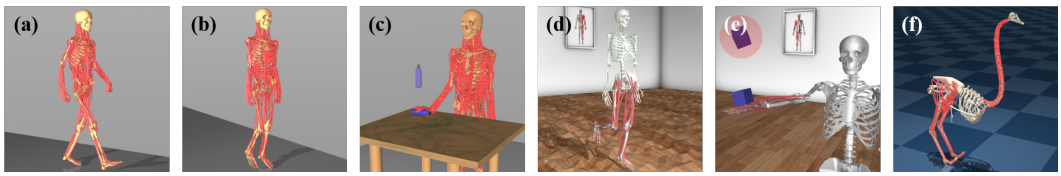


Figure 5: **Experiment environments.** (a) MS-HUMAN-700-Gait ($a \in \mathbb{R}^{700}$). (b) Legs-Walk ($a \in \mathbb{R}^{100}$). (c) Arm-Locate ($a \in \mathbb{R}^{81}$). (d) MyoLegs-Walk with rough terrain ($a \in \mathbb{R}^{80}$). (e) MyoHand-Reorient100 ($a \in \mathbb{R}^{39}$). (f) Ostrich-Run ($a \in \mathbb{R}^{120}$). The semitransparent objects in manipulation environments are the target indicators.

where clip(x, l, h) is a function that restricts the value of x to the interval $[l, h]$. The Index Selection function (IS function), selects corresponding unified actions a_G according to indices determined by grouping results. $\odot$ represents element-wise multiplication, and $[\cdot, \cdot]$ denotes concatenation. a_I is the individual action. k_D, a_D and κ are the hyperparameters of the weight boundary, which relaxes gradually as the training timesteps t increase. During training, the state-dependent adaptation of representations will start at a_D considering the stability of learning. Finally, the policy π will be updated according to the following formula:

$$\pi^* = \arg\max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi}[\sum_{t=0}^{\infty} \gamma^t (R(s_t, a_t) + \alpha H(\pi(a_G|s_t)))] \tag{8}$$

5 Experiments

We demonstrate our method's efficiency during learning and its generalization ability in overactuated motor control benchmarks built in MuJoCo. In this part, we will introduce the benchmarks, the learning process of *DynSyn* and the details of baselines.

5.1 Environments

We create reinforcement learning environments of various models and tasks to test our algorithm. Additionally, two tasks from MyoSuite [40] are taken into account. We use **Model-Task** pair to label the environments, as shown in Figure 5. A complete description including the action space, the state space and the reward function of each environment is given in Appendix B.

Human Motion Imitation: In **FullBody-Gait**, we expect the full body MS-HUMAN-700 model with 206 joints actuated by 700 muscles (described in Section 3.1) to mimic a motion-capture walking trajectory. During training, the model may be initialized at any time step throughout a trajectory cycle.

Human Locomotion: In **Legs-Walk**, a 20-DoF *Legs* model actuated by 100 muscles is used. In **MyoLegs-Walk**, the *MyoLegs* model in MyoSuite with 20 DoF and 80 muscles is used. Both models are expected to walk forward robustly, driven by biomechanically inspired reward functions.

Human Manipulation: In **Arm-Locate**, an *Arm* model of 28 DoF and 81 muscles, with wrist and fingers is used. The agent is trained to learn to grasp a bottle, relocate it to the random target position and orientation. In **MyoHand-Reorient100**, the *MyoHand* model in MyoSuite with 23 DoF and 39 muscles needs to rotate a set of 4 objects, each with 25 different geometries, to predetermined orientations.

Animal Locomotion: In **Ostrich-Run**, an *Ostrich* model [21] with 50 joints actuated by 120 muscles is used. The model is trained to run horizontally as fast as possible by rewarding its velocity.

Generation Tasks: We test various terrain conditions (**MyoLegs-Walk-Terrain**) and different walking targets (**Legs-Walk-Fast, Legs-Walk-Diagnal**) to demonstrate the generalization capability of the dynamical synergistic representation across various physical conditions, as well as the robustness of generated motor behaviors.

The *Legs* model and the *Arm* model are obtained by removing irrelevant degrees of freedom from the full body model. For the locomotion task and the manipulation task, two environments are tested for each task to demonstrate the robustness of the algorithm, given that there are several variations between the models and the environments.

5.2 Learning Dynamical Synergistic Representation

Before training, dynamical synergistic representations (i.e. the grouping of actuators) are generated for each model. We impose random control on joint velocities for $5e5$ simulation frames to collect sequences of actuators' features. For muscle-tendon units, the collected feature is length. The grouping of actuators is then obtained according to Section 4.1. We choose an appropriate number of groups where the difference between maximum and minimum distances among cluster centers are large enough. This process is further detailed in Appendix C.2. To demonstrate that the representation generated by our method can stably capture the dynamical features of the models, we repeat the generation for 10 times on each model and calculate the mean value and variance. The same representation of a single model are retained in tasks with changed conditions to verify the generalization ability of the algorithm. Furthermore, a series of ablation experiments are presented to prove that our choice of the number of groups is reasonable and helpful for the learning of motor control (see Section 6.2).

5.3 Baselines

We compared *DynSyn* with current DRL methods in overactuated systems: SAC [39], SAR [26] and DEP-RL [22]. SAR collects low-dimensional representations from a pre-training collection stage over M time steps and its training stage is over another N time steps. Other methods are directly trained over $M + N$ time steps. It should be noted that DEP-RL is an exploration method which can be integrated into our algorithm. *DynSyn* are based on the RL library Stable-Baselines 3 [41]. Hyperparameters and implementation details in the experiments are summarized in Appendix C.4. All results are averaged across 5 random seeds.

6 Results and Analysis

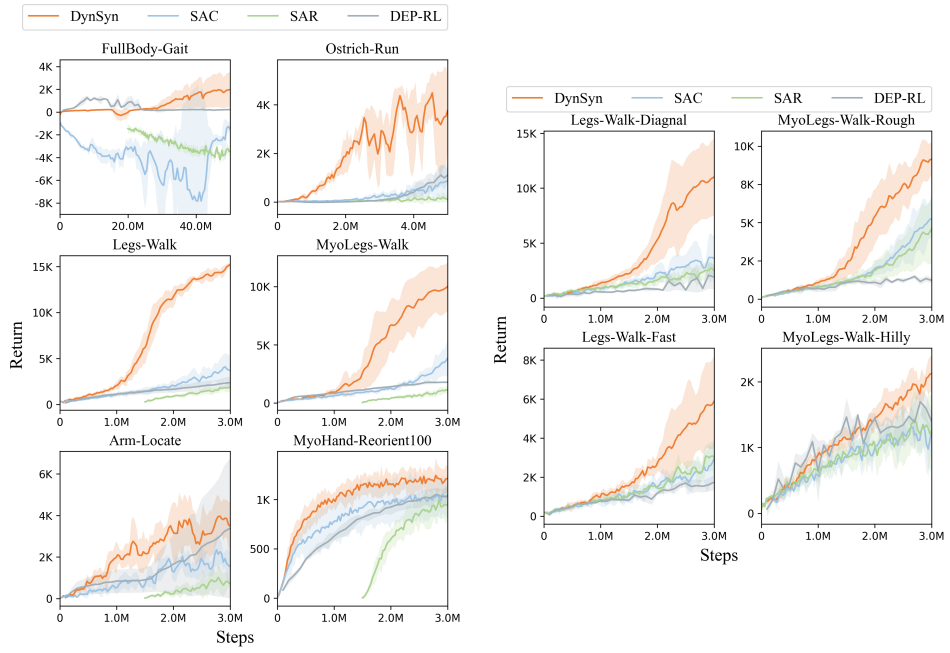
In this section, we present the experimental results, demonstrating that *DynSyn* effectively facilitates motor control across various tasks involving different models, exhibiting state-of-the-art sample efficiency and high stability. Additionally, we illustrate that the dynamical synergistic representations extracted from models exhibit good performance in terms of convergence and interpretability. This allows the model to leverage the representations in learning motor control across diverse tasks, even under varying conditions such as terrain and training targets.

6.1 Efficient Learning

Figure 6(a) illustrates that *DynSyn* achieves higher returns in fewer training steps across all standard experimental environments. This implies that *DynSyn* efficiently produces robust motor control (refer to Figure 1) in various overactuated models and motor tasks. Notably, the performance of baseline agents significantly deteriorates as the number of action dimensions increases, whereas *DynSyn* performs well even in a very high-dimensional action space of 700 dimensions.

6.2 Synergies Generalization

When the same representations are applied to tasks with additional environmental conditions or changed targets, such as rugged terrains and walking direction, *DynSyn* maintains good performance (see Figure 6(b)). This suggests that the generated synergistic representations of the same model can effectively generalize across different tasks. Figure 7(a) shows the average results of 10 preliminary group extractions for the *Legs* model, showing a high probability of obtaining the same grouping result (close to 1) and furthermore, the stability of the extraction process.



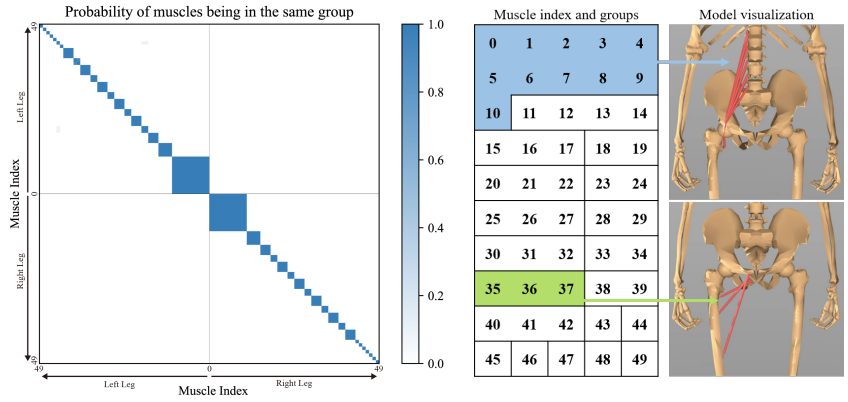
(a) (b)

Figure 6: **Experimental results.** Learning curves in the experimental environments. Mean $\pm$ SD across 5 random seeds for all the environments. (a) Standard experimental results. SAR are depicted to begin at M timesteps in order to account for the steps of pre-training. The return of baselines decreases as the number of action dimensions increases, while *DynSyn* is the only algorithm that performs well even in a very high-dimensional action space of 700 dimensions in the FullBody-Gait environment. (b) Generalization experimental results. Learning curves in the generalization environments in which the physical situations or targets are changed. Here, the M of SAR is 0 because the representations are generated from standard environments.

For ablation study, we utilize a random grouping approach to create 40 clusters for comparison. As shown in Figure 8, our method consistently yields performance improvements. However, randomly generated representations outperform the SAC algorithm, possibly due to the influence of our state-dependent algorithm. The high standard deviation of the learning curve of randomly generated representation shows a decrease in stability. We also attempt to generate representations for varying numbers of clusters. The results demonstrate that our cluster number selection scheme ensures performance stability with a lower standard deviation of the learning curve (see Figure 8). Each grouping mode is tried with 5 different random seeds.

6.3 Physiological Analysis

In accordance with the left-right symmetry in *Legs* modeling, 50 indices, from 0 to 49, are assigned to muscles of each leg symmetrically. In the grouping matrix in Figure 7(a), the upper left quadrant signifies the correlation between left-leg muscles, and the lower right quadrant represents the correlation between right-leg muscles. The remaining part of Figure 7(a) depicts correlation near 0 between pairs of muscles from both legs. Notably, our representation generation method identifies inherent symmetries in the musculoskeletal model. Only groupings within the same leg are observed. Furthermore, the groupings of muscles from the left and right legs are symmetrical. For improved visualization, the muscle grouping result of the right leg is expanded and depicted in Figure 7(b), detailing two representative muscle grouping examples in the musculoskeletal model (i.e., Psoas Major and Thigh Adductors). From a biomechanical perspective, this is evident that muscles grouped together exhibit similar effects, highlighting our method's ability to capture fundamental dynamical characteristics in the system.



(a) (b)

Figure 7: **Muscle grouping of *Legs* model.** (a) Grouping matrix P_n, where p_{ij} is the probability that muscle i and muscle j are in the same group (averaged over 10 seeds). For the *Legs* model where $n = 100$, the first 50 muscle indices represent muscles of the right leg, and the other 50 indices represent the left leg. The model is bilaterally symmetrical, and the muscle indices are reordered for better visualization. (b) Visualization of muscle groupings for the right leg. Each number represents a muscle, and different groups are delineated by borders.

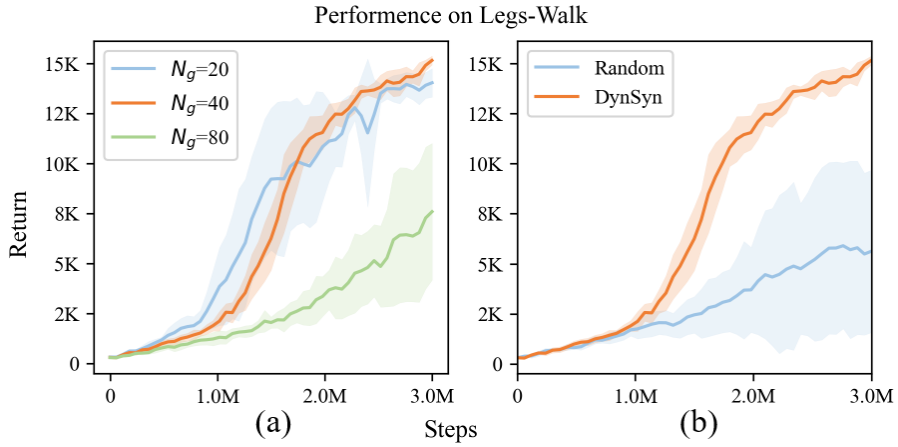


Figure 8: **Ablation study results.** Learning curves on the **Legs-Walk** environment. Mean $\pm$ SD across 5 random seeds. (a) Performance variation with different numbers of clusters, we apply $N_g = 40$ in our final control tasks. (b) Performance comparison between randomly generating 40 clusters and our method generating the dynamical synergistic representation of 40 clusters.

7 Conclusion and Discussion

We introduce *DynSyn*, a deep RL method that generates synergistic representations of actuators from dynamical structures of overactuated systems and make task-specific, state-dependent adaptation to the representations, thereby expediting and stabilizing motor control learning. Applying *DynSyn* to musculoskeletal locomotion and manipulation tasks, we demonstrate its superior learning efficiency compared to all baselines. Additionally, we illustrate the robust generalization ability of the extracted synergistic representations across various motor tasks with the same model. In conclusion, our work offers an efficient, generalizable, and interpretable approach to controlling high-dimensional redundant actuation systems. The generation method of synergistic representations can help deepen the understanding of motor intelligence. This research aims to facilitate the training of motor control policies for use in artificial intelligence, robotics and medicine, contributing to the development of a versatile embodied agent.

Despite the promising outcomes, distinctions persist between real embodied motor intelligence and the musculoskeletal model simulation employed in our study. For instance, current simulation methods primarily leverage proprioception (joint position and velocity), whereas in the real world, an animal receives additional sensory inputs, including vision and touch [42, 43, 44]. To enhance customization for specific applications, further work on biomechanically realistic simulations is essential. Other significant limitations include the multiple potential solutions in overactuated systems, and our method can only generate one of the numerous high-dimensional combinations to control the system. Future research may need to consider establishing a solution space of control patterns.

References

[1] 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*, pages 91–100. PMLR, 2022.

[2] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan. A survey of embodied ai: From simulators to research tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(2): 230–244, 2022.

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

[4] P. Hsu, J. Mauser, and S. Sastry. Dynamic control of redundant manipulators. *Journal of Robotic Systems*, 6(2):133–148, 1989.

[5] M. Schneiders, M. Van De Molengraft, and M. Steinbuch. Benefits of over-actuation in motion systems. In *Proceedings of the 2004 American control conference*, volume 1, pages 505–510. IEEE, 2004.

[6] S. S. Tohidi, Y. Yildiz, and I. Kolmanovsky. Fault tolerant control for over-actuated systems: An adaptive correction approach. In *2016 American control conference (ACC)*, pages 2530–2535. IEEE, 2016.

[7] N. Bernstein. The co-ordination and regulation of movements. *The co-ordination and regulation of movements*, 1966.

[8] L. H. Ting, S. A. Chvatal, S. A. Safavynia, and J. Lucas McKay. Review and perspective: neuromechanical considerations for predicting muscle activation patterns for movement. *International journal for numerical methods in biomedical engineering*, 28(10):1003–1014, 2012.

[9] F. E. Zajac. Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Critical reviews in biomedical engineering*, 17(4):359–411, 1989.

[10] D. M. Wolpert and Z. Ghahramani. Computational principles of movement neuroscience. *Nature neuroscience*, 3(11):1212–1217, 2000.

[11] A. d'Avella, P. Saltiel, and E. Bizzi. Combinations of muscle synergies in the construction of a natural motor behavior. *Nature neuroscience*, 6(3):300–308, 2003.

[12] A. d'Avella, A. Portone, L. Fernandez, and F. Lacquaniti. Control of fast-reaching movements by muscle synergy combinations. *Journal of Neuroscience*, 26(30):7791–7810, 2006.

[13] Y. Sui, K. ho Kim, and J. W. Burdick. Quantifying performance of bipedal standing with multi-channel emg. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3891–3896. IEEE, 2017.

[14] Ł. Kidziński, C. Ong, S. P. Mohanty, J. Hicks, S. Carroll, B. Zhou, H. Zeng, F. Wang, R. Lian, H. Tian, et al. Artificial intelligence for prosthetics: Challenge solutions. In *The NeurIPS'18 Competition: From Machine Learning to Intelligent Conversations*, pages 69–128. Springer, 2020.

[15] V. Caggiano, H. Wang, G. Durandau, S. Song, Y. Tassa, M. Sartori, and V. Kumar. Myochallenge: Learning contact-rich manipulation using a musculoskeletal hand, 2022.

[16] V. Caggiano, H. Wang, G. Durandau, S. Song, C. K. Tan, C. Berg, P. Schumacher, M. Sartori, and V. Kumar. Myochallenge 23: Towards human-level dexterity and agility, 2023.

[17] S. Song, Ł. Kidziński, X. B. Peng, C. Ong, J. Hicks, S. Levine, C. G. Atkeson, and S. L. Delp. Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation. *Journal of neuroengineering and rehabilitation*, 18:1–17, 2021.

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

[19] J. Park, S. Min, P. S. Chang, J. Lee, M. S. Park, and J. Lee. Generative gaitnet. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.

[20] Y. Feng, X. Xu, and L. Liu. Musclevae: Model-based controllers of muscle-actuated characters. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023.

[21] V. La Barbera, F. Pardo, Y. Tassa, M. Daley, C. Richards, P. Kormushev, and J. Hutchinson. Ostrichrl: A musculoskeletal ostrich simulation to study bio-mechanical locomotion. *arXiv preprint arXiv:2112.06061*, 2021.

[22] P. Schumacher, D. Häufle, D. Büchler, S. Schmitt, and G. Martius. Dep-rl: Embodied exploration for reinforcement learning in overactuated and musculoskeletal systems, 2023.

[23] A. S. Chiappa, A. M. Vargas, A. Z. Huang, and A. Mathis. Latent exploration for reinforcement learning. *arXiv preprint arXiv:2305.20065*, 2023.

[24] V. Caggiano, S. Dasari, and V. Kumar. Myodex: a generalizable prior for dexterous manipulation. In *International Conference on Machine Learning*, pages 3327–3346. PMLR, 2023.

[25] R. Cheng, Y. Sui, D. Sayenko, and J. W. Burdick. Motor control after human sci through activation of muscle synergies under spinal cord stimulation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(6):1331–1340, 2019.

[26] C. Berg, V. Caggiano, and V. Kumar. Sar: Generalization of physiological agility and dexterity via synergistic action representation. *arXiv preprint arXiv:2307.03716*, 2023.

[27] K. Zhao, H. Wen, Z. Zhang, M. Atzori, H. Müller, Z. Xie, and A. Scano. Evaluation of methods for the extraction of spatial muscle synergies. *Frontiers in neuroscience*, 16:732156, 2022.

[28] S. Grillner. Neurobiological bases of rhythmic motor acts in vertebrates. *Science*, 228(4696): 143–149, 1985.

[29] N. Dominici, Y. P. Ivanenko, G. Cappellini, A. d'Avella, V. Mondì, M. Cicchese, A. Fabiano, T. Silei, A. Di Paolo, C. Giannini, et al. Locomotor primitives in newborn babies and their development. *Science*, 334(6058):997–999, 2011.

[30] M. C. Tresch and O. Kiehn. Motor coordination without action potentials in the mammalian spinal cord. *Nature neuroscience*, 3(6):593–599, 2000.

[31] Y. P. Ivanenko, R. E. Poppele, and F. Lacquaniti. Five basic muscle activation patterns account for muscle activity during human locomotion. *The Journal of physiology*, 556(1):267–282, 2004.

[32] L. H. Ting and J. M. Macpherson. A limited set of muscle synergies for force control during a postural task. *Journal of neurophysiology*, 93(1):609–613, 2005.

[33] W. Zhou, S. Bajracharya, and D. Held. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, pages 1719–1735. PMLR, 2021.

[34] A. Allshire, R. Martín-Martín, C. Lin, S. Manuel, S. Savarese, and A. Garg. Laser: Learning a latent action space for efficient reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6650–6656. IEEE, 2021.

[35] E. Aljalbout, M. Karl, and P. van der Smagt. Clas: Coordinating multi-robot manipulation with central latent action spaces. In *Learning for Dynamics and Control Conference*, pages 1152–1166. PMLR, 2023.

[36] C. Zuo, K. He, J. Shao, and Y. Sui. Self model for embodied intelligence: Modeling full-body human musculoskeletal system and locomotion control with hierarchical low-dimensional representation. *arXiv preprint arXiv:2312.05473*, 2023.

[37] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[38] H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

[39] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[40] V. Caggiano, H. Wang, G. Durandau, M. Sartori, and V. Kumar. Myosuite – a contact-rich simulation suite for musculoskeletal motor control, 2022.

[41] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. Stable baselines3, 2019.

[42] A. E. Patla. Understanding the roles of vision in the control of human locomotion. *Gait & posture*, 5(1):54–69, 1997.

[43] A. E. Patla. How is human gait controlled by vision. *Ecological psychology*, 10(3-4):287–302, 1998.

[44] J. Jeka, K. S. Oie, and T. Kiemel. Multisensory information for human postural control: integrating touch and vision. *Experimental brain research*, 134:107–125, 2000.

[45] M. Millard, T. Uchida, A. Seth, and S. L. Delp. Flexing computational muscle: modeling and simulation of musculotendon dynamics. *Journal of biomechanical engineering*, 135(2):021005, 2013.

A Neuro-muscle dynamics

The input control signal of muscle-tendon units is the neural excitation $ctrl$, and the muscle activation act is calculated by a first-order nonlinear filter as follow:

$$\frac{\partial act}{\partial t} = \frac{ctrl - act}{\tau(ctrl, act)}, \tau(ctrl, act) = \begin{cases} \tau_{act}(0.5 + 1.5act) & ctrl > act \\ \tau_{deact}/(0.5 + 1.5act) & ctrl \leq act \end{cases}, \tag{9}$$

$(\tau_{act}, \tau_{deact})$ is a time constant to activate or deactivate latency of defaults $(10ms, 40ms)$. The force produced by a single muscle-tendon unit can be formulated as

$$f_m(act) = f_{max} \cdot [F_l(l_m) \cdot F_v(v_m) \cdot act + F_p(l_m)], \tag{10}$$

where f_{max} stands for the maximum isometric muscle force and act, l_m, v_m respectively stand for the activation, normalized length and normalized velocity of the muscle. F_l and F_v represent force-length and force-velocity functions fitted using data from biomechanical experiments [45].

B Environment Details

For all environments, the simulation time step is 0.01s. The action space consists of muscle excitations $ctrl$ (i.e. motor neuron signals). The dimensions of action and state spaces, number of joints and episode length of all the environments are summarized in Table A.1. Task and reward parameters are summarized in Table A.2.

FullBody-Gait We expect the full body MS-HUMAN-700 model with 206 joints actuated by 700 muscles (Section 3.1) to mimic a motion-capture walking trajectory. During training, the model may be initialized at any time step throughout a trajectory cycle. The state space consists of simulation time, joint positions, joint velocities, muscle forces, muscle lengths, muscle velocities, muscle activation and reference joint positions. The reward function is:

$$R = w_v R_v + w_q R_q + w_h \mathbb{I}_{\text{alive}} \tag{11}$$

$$R_v = \exp(-(v_x^{com} - v_x^t)^2) + \exp(-(v_y^{com} - v_y^t)^2) \tag{12}$$

$$R_q = -||q - q^r||_2 \tag{13}$$

where q is actual joint positions, q^r is the reference joint positions, $\left\{v_x^{com}, v_y^{com}\right\}$ is the velocity of the center of mass and $\left\{v_x^t, v_y^t\right\}$ is the desired velocity. w_v, w_q and w_h are the weights.

MyoLegs-Walk The *MyoLegs* model in MyoSuite with 20 DoF and 80 muscles is used. The model is expected to walk forward robustly, driven by biomechanically inspired reward functions:

$$R = w_v R_v - w_c R_c + w_r R_r + w_j R_j - w_a R_a - w_d R_d \tag{14}$$

$R_d = \{falled\}$, imposes a penalty when the model falls. The weights w_v, w_c, w_r, w_j, w_a, and w_d determine the importance of each reward term. The other terms are as follow:

$$R_v = \exp(-\sqrt{v_x^r - v_x}) + \exp(-\sqrt{v_y^r - v_y}) \tag{15}$$

v^r and v represent the desired and actual velocity of the center of mass. R_v represents the velocity reward.

$$R_c = |||[0.8\cos(\phi \times 2\pi + \pi), 0.8\cos(\phi \times 2\pi)] - [q_{rhip}, q_{lhip}]|| \tag{16}$$

ϕ is the phase percentage of the pre-define gait period. q_{rhip} and q_{lhip} are the hip flexion angle of both legs. R_c encourages rhythmic hip movements.

$$R_r = \exp(-5||(q_{pelvis} - q_{pelvis}^{init})||) \tag{17}$$

q_{pelvis} and q_{pelvis}^{init} are the quaternions of pelvis and its initial value when reset. R_r encourages the model to follow a predetermined rotation pattern.

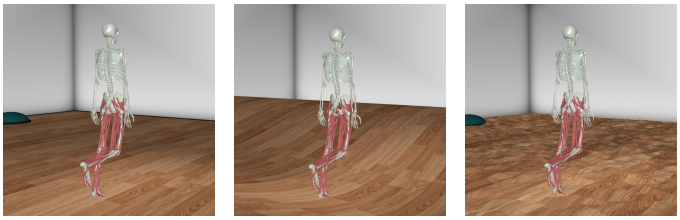
$$R_j = \exp(-5 \sum_{i=1}^{N} |q_i|/N) \tag{18}$$

In the environment, $N = 4$ and q_i are the hip abduction and rotation angles of both legs. R_j penalizes deviations from desired joint angles.

$$R_a = ||act||/N_a \tag{19}$$

act is the muscle activation vector, N_a is the number of muscles, and R_a promotes efficient actuator usage by computing the norm of the action divided by the number of actuators.

The state space consists of simulation time, joint positions (except x and y positions for the base segment), joint velocities, muscle forces, muscle lengths, muscle velocities and muscle activations. The task-specific observations include time phase percentage of the gait, velocity and height of the center of mass, torso angle, the height of the feet and their positions relative to the pelvis. The diverse terrain conditions including slopes and rough ground can be added to the task (see Figure 9).



(a) (b) (c)

Figure 9: **MyoLegs-Walk terrains.** (a) Flat terrain. (b) Hilly terrain. (c) Rough terrain.

Legs-Walk A 20-DoF *Legs* model actuated by 100 muscles is used. The observations and reward function are the same as **MyoLegs-Walk** environment except that there is no R_a term. In addition, we apply diverse walking speed targets in the simulation.

Arm-Locate An *Arm* model of 28 DoF and 81 muscles with wrist and fingers is used. The agent is trained to learn to grasp a bottle, relocate it to the target position and reorient it to the target orientation. The position and orientation of target are randomized when reset. The reward function is:

$$R = w_p R_p \times w_o R_o + w_r R_r - w_a R_a \tag{20}$$

$R_a = ||act||/N_a$, promotes efficient actuator usage. We use multiplication to enforce the relocation and reorientation simultaneously. The weights w_p, w_o, w_r and w_a determine the importance of each reward term. The other terms are as follow:

$$R_p = \exp(-10\sqrt{||p_{target} - p_{object}||}) \tag{21}$$

p_{target} and p_{object} represent the positions of the target and the object, respectively. R_p represents the reward for relocation.

$$R_o = \exp(-2||o_{target} - o_{object}||) \tag{22}$$

o_{target} and o_{object} represent the orientation of the target and the object (in Euler angle, except for the rotation of the bottle around the vertical axis). R_o represents the reward for reorientation.

$$R_r = \exp(-10||p_{palm} - p_{object}||) \tag{23}$$

p_{palm} and p_{object} represent the positions of the palm of the arm model and the position of object. R_r encourages the model to grab the object.

The state space consists of simulation time, joint positions, joint velocities, muscle forces, muscle lengths, muscle velocities and muscle activations. The task-specific observations include the positions of the object and the target, the orientations of the object and the target, the error of position and orientation, the position of the model's palm and the distance between the palm and the object.

MyoHand-Reorient100 *MyoHand* model in MyoSuite with 23 DoF and 39 muscles needs to rotate a set of 4 objects, each with 25 different geometries, to a given orientation without dropping

them. This set of 100 objects is randomly presented, and initialized onto the hand. The reward function is:

$$R = -w_p R_p + w_o R_o - w_d R_d - w_a R_a + w_b R_b \tag{24}$$

$R_a = ||act||/N_a$, promotes efficient actuator usage. The weights w_p, w_o, w_d, w_a and w_b determine the importance of each reward term. The other terms are as follow:

$$R_p = ||p_{target} - p_{object}|| \tag{25}$$

p_{target} and p_{object} represent the positions of the target and the object. R_p keeps the object at its initial position (i.e. onto the palm).

$$R_o = \cos(o_{target} - o_{object}) \tag{26}$$

o_{target} and o_{object} represent the orientations of the target and the object (in vector). R_o represents the reward for reorientation.

$$R_d = (||p_{target} - p_{object}|| > 0.075) \tag{27}$$

R_d represents the penalty for dropping objects.

$$\begin{aligned} R_b = &(\cos(o_{target} - o_{object}) > 0.9) \times (||p_{target} - p_{object}|| < 0.075) \\ &+ 5 \times (\cos(o_{target} - o_{object}) > 0.95) \times (||p_{target} - p_{object}|| < 0.075) \end{aligned} \tag{28}$$

R_b represents the bonus reward for simultaneous rotational and positional alignment above a threshold.

The state space consists of simulation time, joint positions, joint velocities, muscle forces, muscle lengths, muscle velocities and muscle activations. The task-specific observations include the positions of the object and the target, the orientations of the object and the target, the velocities of objects, and the error of position and orientation.

Ostrich-Run An *Ostrich* model [21] with 50 joints actuated by 120 muscles is used. It needs to run horizontally as fast as possible, rewarded only by the velocity of the center of mass projected to the x-axis.

$$R = w_v v_x^{COM} \tag{29}$$

The state space consists of joint positions (except x position for the base segment), joint velocities, muscle forces, muscle lengths, muscle velocities and muscle activations. The task-specific observations include the height of ostrich torso, the height of the feet and the horizontal velocity.

Table A.1: The action, state dimensions, number of joints and episode length of all the environments.

Environment	Action dimensions	State dimensions	Number of joints	Episode length
FullBody-Gait	700	2971	206 (6 for the base)	3000
Legs-Walk	100	488	36 (6 for the base)	1000
MyoLegs-Walk	80	403	34 (6 for the base)	1000
Arm-Locate	81	442	48 (6 for the object)	200
MyoHand-Reorient100	39	200	29 (6 for the object)	50
Ostrich-Run	120	596	56 (6 for the base)	1000

C Implementation Details

C.1 Action normalization

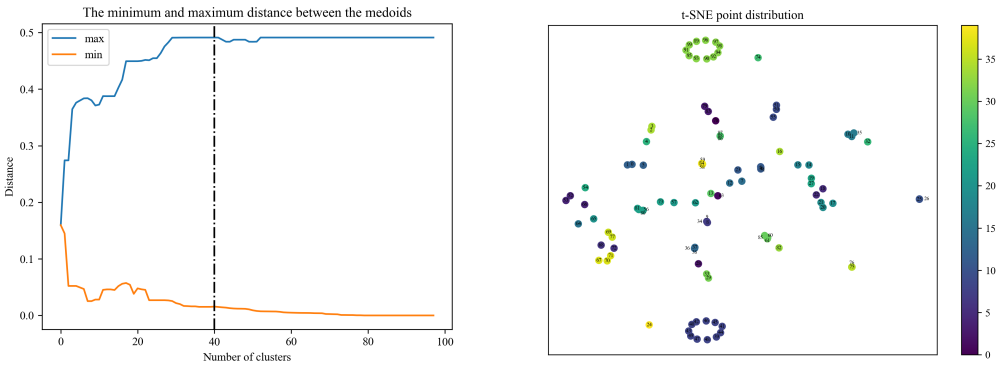
Our preliminary experiments reveal that in MyoSuite, the action space, originally [0, 1], is normalized to [-1, 1] using Equation 30. This normalization method enhances training effectiveness. Consequently, we apply this normalization approach in all our environments and algorithm comparison experiments.

$$a = \frac{1}{1 + e^{-5(a-0.5)}} \tag{30}$$

C.2 Dynamical synergistic representation generation

In the process of representation generation, determining the number of groups is crucial. In Figure 10(a), we illustrate the maximum and minimum values of the distance among cluster centers for different group configurations. The algorithm exhibits robust and explainable performance when we choose an appropriate number of clusters where the difference between maximum and minimum distances are large enough. In Figure 10(b), it is evident that when the selected number of groups is 40, the t-SNE visualization maintains symmetry and interpretability.

As illustrated in Figure 11 and Figure 12, the grouping results are shown to converge to their final grouping with a data point quantity as low as 25,600. It's also displayed that even if we have only 100 data points, the grouping result is similar to the final result.



(a) (b)

Figure 10: **Generation criteria and visualization.** (a) In *Legs* model, the K-medoids algorithm is employed for clustering with varying cluster numbers. The curve depicting the maximum and minimum distance between cluster centers changes with the number of clusters. (b) t-SNE algorithm is used to reduce the cosine similarity distance to two dimensions for visualization.

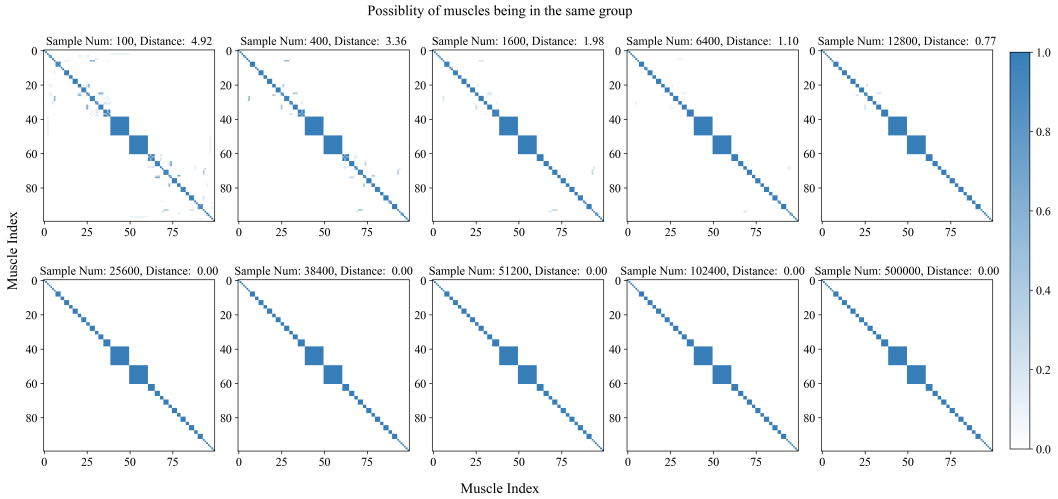


Figure 11: Grouping matrix P_n derived from varying sample sizes, and the distance (Frobenius norm) between them and a 500K-sample grouping matrix. p_{ij} is the probability that muscle i and muscle j are in the same group (averaged over 10 seeds). The grouping results are shown to converge to their final grouping with a data point quantity as low as 25,600. Even if we have only 100 data points, the grouping result is similar to the final result.

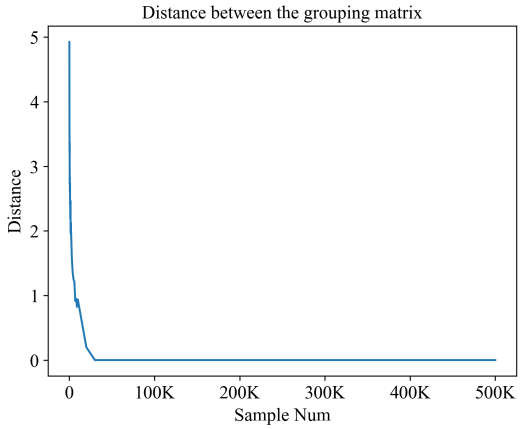


Figure 12: The distance (Frobenius norm) between grouping matrices derived from varying sample sizes and a 500K-sample grouping matrix.

C.3 Baselines

We compare our algorithm with the best performing baselines, SAC, SAR and DEP-RL. SAR and DEP-RL algorithms are implemented using the official released code. SAC, SAR and DynSyn all adopt the DRL framework Stable baselines3, and DEP-RL adopts the Tonic framework. All algorithms adopt SAC as the basic algorithm. The specific algorithm parameters of SAR and DEP-RL are those reported in the original papers, and for models with similar complexity we use the same parameters. See Table A.3 and Table A.4 for details.

C.4 Hyperparameters of *DynSyn* and baselines

Algorithm hyperparameters are summarized in Table A.3 and A.4.

Table A.2: The task and reward parameters of all the environments.

Task	FullBody-Gait		Ostrich-Run	
	Parameter	Value	Parameter	Value
	Min pelvis height	0.6	Min head height	0.9
	Min sternum height	1	Min pelvis height	0.6
	Max rot.	0.8	Min torso angle	-0.8
			Max torso angle	0.8
Reward	Pos.	1	Vel.	1
	Vel.	0.005		
	Concerned	1		
	Done	10		
Task	Legs-Walk		Arm-Locate	
	Parameter	Value	Parameter	Value
	Min height	0.8	Pos. x target bound	(-0.1, 0.1)
	Max rot.	0.8	Pos. y target bound	(-0.2, 0.2)
	Hip period	100	Pos. z target bound	(0.1, 0.3)
Reward	Target forward vel.	1.2 (3 in Fast env)	Ori. x target bound	(-0.4, 0.4)
	Target lateral vel.	0 (1.2 in Diagnal env)	Ori. y target bound	(-0.4, 0.4)
	Vel.	5	Pos.	50
	Cyclic hip	10	Ori.	5
	Ref rot.	10	Reach	10
	Joint angle	5	Action reg.	1
	Done	100		
Task	MyoLegs-Walk		MyoHand-Reorient100	
	Parameter	Value	Parameter	Value
	Min height	0.8	-	-
	Max rot.	0.8		
	Hip period	100		
	Target forward vel	1.2		
	Target lateral vel	0		
	Terrain	Flat (Rough, Hilly, in separate envs)		
Reward	Vel.	5	Pos.	1
	Cyclic hip	10	Ori.	1
	Ref rot.	10	Drop	5
	Joint angle	5	Action reg.	5
	Action reg.	1	Bonus	10
	Done	100		

Table A.3: Parameters of SAC, *DynSyn*, SAR and DEP-RL in the standard tasks

Algorithm	Parameter	Task					
		FullBody-Gait	Legs-Walk	MyoLegs-Walk	Arm-Locate	MyoHand-Reorient100	Ostrich-Run
SAC	Learning rate	linear_schedule(0.001)					
	Batch size	256					
	Buffer size	3e6					
	Warmup steps	100					
	Discount factor	0.98					
	Soft update coeff.	2					
	Train frequency (steps)	1					
	Gradient steps	4					
	Traget update interval	1					
	Environment number	80					
	Entropy coeff.	auto					
	Target entropy	auto					
	Policy hiddens	[512, 300]	[256, 256]				
	Q hiddens	[512, 300]	[256, 256]				
	Activation	ReLU					
	Training steps	5e7	3e6				5e6
DynSyn	Control Amplitude	5	10	10	5	100	10
	Trajectory steps	5e5					
	Control frequency	10					
	Number of groups	100	40		25		40
	a_D	3e7	1e6				
	k_D	5e-9	5e-8				
SAR	Dimensionality	200	20				
	Blend weight	0.66					
	Training steps (play phase + training phase)	2e7+3e7	1.5e6+1.5e6				2e6+3e6
DEP-RL	Bias rate	0.002					0.03
	Buffer size of DEP	200					90
	Intervention length	5					4
	Intervention proba	0.0004					
	Kappa	1169.7					20
	Normalization	Independent					
	Q norm selector	l2					
	regularization	32					
	s4avg	2					1
	Sensor delay	1					
	tau	40					8
	Test episode every	3					
	Time dist	5					
	With learning	True					
	Return steps	2					
	Entropy coeff.	0.2					
	Learning rate	3e-4					
	Environment number (parallel * sequential)	80*1					

Table A.4: Parameters of SAC, *DynSyn*, SAR and DEP-RL in the generalization tasks

Algorithm	Parameter	Task			
		Legs-Walk-Diagnal	Legs-Walk-Fast	MyoLegs-Walk-Hilly	MyoLegs-Walk-Rough
SAC	Learning rate	linear_schedule(0.001)			
	Batch size	256			
	Buffer size	3e6			
	Warmup steps	100			
	Discount factor	0.98			
	Soft update coeff.	2			
	Train frequency (steps)	1			
	Gradient steps	4			
	Traget update interval	1			
	Environment number	80			
	Entropy coeff.	auto			
	Target entropy	auto			
	Policy hiddens	[256, 256]			
	Q hiddens	[256, 256]			
	Activation	ReLU			
	Training steps	3e6			
DynSyn	a_D	1e6			
	k_D	5e-6			
SAR	Dimensionality	20			
	Blend weight	0.66			
	Training steps (play phase + training phase)	0+3e6			
DEP-RL	Bias rate	0.002			
	Buffer size of DEP	200			
	Intervention length	5			
	Intervention proba	0.0004			
	Kappa	1169.7			
	Normalization	Independent			
	Q norm selector	l2			
	regularization	32			
	s4avg	2			
	Sensor delay	1			
	tau	40			
	Test episode every	3			
	Time dist	5			
	With learning	True			
	Return steps	2			
	Entropy coeff.	0.2			
	Learning rate	3e-4			
	Environment number (parallel $\times$ sequential)	80$\times$1			