2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Subequivariant Reinforcement Learning Framework for Coordinated Motion Control

Haoyu Wang[1†], Xiaoyu Tan[2†], Xihe Qiu[1†*] and Chao Qu[2*]

***Abstract*— Effective coordination is crucial for motion control with reinforcement learning, especially as the complexity of agents and their motions increases. However, many existing methods struggle to account for the intricate dependencies between joints. We introduce CoordiGraph, a novel architecture that leverages subequivariant principles from physics to enhance coordination of motion control with reinforcement learning. This method embeds the principles of equivariance as inherent patterns in the learning process under gravity influence, which aids in modeling the nuanced relationships between joints vital for motion control. Through extensive experimentation with sophisticated agents in diverse environments, we highlight the merits of our approach. Compared to current leading methods, CoordiGraph notably enhances generalization and sample efficiency.**

I. INTRODUCTION

Reinforcement learning (RL) is a prominent approach for enabling intelligent agents to acquire skills for intricate tasks via iterative trial and error[1]. However, managing the coordinated movements of multiple joints through RL, especially in agents navigating complex physical environments like multi-joint robotic systems[2], is challenging[3], [4]. Traditional RL techniques[5] often address this problem through the lens of the curse of dimensionality and training instabilities [6]. Nevertheless, these techniques often overlook the interactions between joints and the physical principles prevalent in most agent operating conditions.

Graph neural networks (GNNs) have demonstrated potential in RL for coordinated motion control by representing node internal interactions[7], [8]. However, they do have some challenges prevent the GNN in practical utilization[9]. Specifically, GNNs can occasionally find it difficult to recognize dynamic symmetries and maintain equivariance in joint interactions, which can result suboptimal coordination performance[10]. They also often face challenges in exploration, limiting their efficiency in discovering rewards in novel scenarios[11], [12]. Additionally, GNNs typically require significant training data and time to reach the desired performance[13], which can constrain their practical adaptability[14].

To intergrete prior knowledge of symmetry-related prior in the RL-based motion control, many equivariant techniques have been proposed[15], [16]. However, most of equivariant neural networks are designed under the assumption of specific symmetries in the input graph data. This design choice can restrict their utility, especially for graph structures that exhibit global symmetries[17]. In complex graph datasets, these methods might not accurately capture intricate symmetry patterns, potentially hindering performance and generalization[18], [19].

In this study, we propose **CoordiGraph**, a novel approach that leverages coordinated subequivariant networks for joint motion control in reinforcement learning. Our method addresses the limitations of GNNs and equivariant technique in coordinating motion control with RL. By incorporating the concept of subequivariance into the GNNs framework, CoordiGraph effectively models symmetries and subequivariance within agent joints. This enhances agent performance and efficiency in cooperative motion learning tasks, as illustrated in Figure 1. Our model ensures the preservation of input data symmetry and accurately captures subequivariance properties between joints, enabling agent actions to maintain symmetry and equivariance.

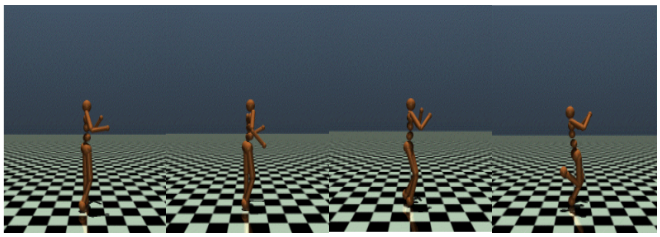


Fig. 1. Training a humanoid agent in the MuJoCo environment with the objective of enabling it to transition from an inability to stand to coordinated joint movements.

Through extensive experiments conducted on diverse reinforcement learning control benchmark tasks, we empirically demonstrate the effectiveness in coordinating agent motion, showcasing the significant benefits of incorporating subequivariant principles with reinforcement learning in motion control tasks by achieving improved coordination and learning efficiency.

II. RELATED WORK

A. Reinforcement Learning on Motion Control

Reinforcement learning is a method of learning optimal strategies by interacting with the environment, using reward signals to guide the actions of an agent and maximize cumulative rewards[20]. It has applications in domains like robot control and game AI[21]. The goal is to find the best policy that maximizes the agent's cumulative rewards. Various algorithms have been proposed in the field of reinforcement

† Both authors contributed equally to this work

* Corresponding author: Xihe Qiu, Chao Qu, qiuxihe@sues.edu.cn, quchao_tequila@inftech.ai

[1]Haoyu Wang and Xihe Qiu are with Shanghai University of Engineering Science, Shanghai, China

[2]Xiaoyu Tan and Chao Qu are with INF Technology (Shanghai) Co., Ltd

learning for motion control[22], including deep reinforcement learning methods like Deep Q-Network for training agents in Atari games[23]. Evolution Strategies have been used to train agents for complex behaviors like walking and running[24], [25]. Transfer learning from simulated to real environments has also been achieved. Genetic algorithms and particle swarm optimization[26], [27] have been used to optimize coordinated movements among different components of agents[28], [29]. Hierarchical approaches, where higher-level policies guide lower-level policies[30], have been successful in tasks requiring hierarchical organization, such as multi-agent navigation and cooperative transportation[31], [32].

B. Coordinating Joint Movements for Graph

Graph neural networks (GNN) are widely used in reinforcement learning to model interactions between entities in a graph structure[33]. GNN leverage information propagation through edges and nodes to capture precise dependencies between agents in coordinated tasks[34], [35]. Variants such as graph convolutional network aggregate information from neighboring agents, enabling localization and recognition of temporal actions[36]. Graph attention networks assign weights to features based on relevance, accurately predicting future trajectories by representing positions and relationships as a graph[37]. GraphSAGE uses sampling strategies to process information from fixed-sized agent neighborhoods[38], achieving better generalization capabilities regardless of graph size and structure[39], [40].

C. Advancements in Subequivariant Techniques

Subequivariant neural networks is an extension of neural Networks that handles symmetry and subequivariance[41]. Subequivariant network enhances the learning capability of traditional neural Networks by introducing the concept of subequivariance. It builds upon the idea of equivariant graph neural networks (EGNN) that handle graph data with symmetry[42]. EGNN achieves this by designing graph convolution operations with symmetry and preserving the symmetry properties of the input graph during learning. Researchers have also explored other improvement measures, such as graph matching networks (GMN) that handle local symmetry by applying equivariance operations like rotation and translation[43]. Leveraging subgraph information and incorporating equivariance into cross-domain graph neural networks has shown effective generalization to unseen target domains[44]. Other studies have also investigated reinforced learning[45] and low-dimensional feature extraction[46] in the context of equivariant graph neural networks[47].

III. Methodology

We define the state space S as an n-dimensional vector representing the state of each joint: $S = [s_1, s_2, \ldots, s_n]$.

Designing between Agents and Environment The agent, modeled as a discrete graph structure, can perform actions within a certain range, such as applying force or torque [48]. The action space A is an n-dimensional vector representing the actions of each joint: $A = [a_1, a_2, \ldots, a_n]$. By designating the body node as the coordinate system, the joint nodes represent the degrees of freedom between them. For example, Walker2d-v2, the root node determines the agent's position in MuJoCo.

The policy function π maps the state space S to a probability distribution over the action space A. $\pi(a|s)$ represents the probability of selecting action a given state s. The value function V estimates the expected return under state s, denoted as $V(s)$ mapping the state space S to the state value.

Subequivariant Learning Networks We design our model to explore equivariant properties by incorporating external fields, such as gravity, in a reasonable manner. It decomposes the graph structure into subgraphs based on equivariant properties and propagates equivariant information between these subgraphs to handle graph feature data across different joints of an intelligent agent, as shown in Figure 2.

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in N(i)} f(h_i^{(l)}, h_j^{(l)}, E_{ij})\right), \tag{1}$$

here $h_i^{(l)}$ represents the feature representation of the central node i in the l-th layer, $N(i)$ denotes the neighboring nodes of the central node i, f is the aggregation function, and E_{ij} represents the external field information between the central node i and the neighboring node j.

CoordiGraph introduces an Object-aware Message Passing mechanism for learning physical interactions between objects of different shapes. This mechanism handles object properties, sizes, and shapes, enabling hierarchical modeling and improving the model's ability to handle complex interactions.

$$m_{ij}^{(l+1)} = g(h_i^{(l)}, h_j^{(l)}, s_i^{(l)}, s_j^{(l)}), \tag{2}$$

$m_{ij}^{(l+1)}$ represents the object-aware message between the central node i and the neighboring node j in the layer of $(l+1)$. $s_i^{(l)}$ and $s_j^{(l)}$ represent the shape representations of the central node i and the neighboring node j in the l-th layer. The function g integrates node features and shape information, evolving with subsequent policy updates. Its purpose is to capture the physical interactions between objects of different shapes by combining their features and shape information.

$$\begin{aligned} m_v^{(l)} &= \sum_{u \in N(v)} \frac{1}{\sqrt{|N(v)||N(u)|}} \cdot W^{(l)} \cdot h_u^{(l-1)} \\ h_v^{(l)} &= \sigma(m_v^{(l)} + W_0^{(l)} \cdot h_v^{(l-1)}) \end{aligned} \tag{3}$$

It calculates the aggregated message $m_v^{(l)}$ for node v in the l-th layer by summing the weighted hidden states $h_u^{(l-1)}$ of its neighboring nodes u. The weights are normalized by the square root of the product of the degrees of the nodes, ensuring a balanced influence from different degrees. The weights are applied through the weight matrix $W^{(l)}$. The update rule for the hidden state $h_v^{(l)}$ of node v in the l-th layer combines the aggregated message $m_v^{(l)}$ with the previous hidden state $h_v^{(l-1)}$ using the self-loop weight matrix $W_0^{(l)}$.

In the subequivariant neural network model, we introduce two time-related concepts. Firstly, there is the classical notion

of time, represented by the time step t in the environment. Additionally, we define the internal propagation step as τ, which represents the series of steps performed by the model to determine node actions based on environmental observations within each time step.

$$m_{uv}^{(t)} = f(h_{tu}, h_{tv}), \tag{4}$$

here $m_{uv}^{(t)}$ represents the message vector from node u to node v, h_{tu} and h_{tv} denote the state vectors of node u and node v respectively in propagation step t.

During time step t, each node u performs information propagation by computing message vectors with its neighboring node v. By utilizing the state vectors h_{tu} and h_{tv} of nodes u and v respectively, node u calculates the message vector $m_{uv}^{(t)}$ using the function f. These message vectors update the node's state and facilitate further information propagation in subsequent propagation steps.

$$h_i^{(l+1)} = g(h_i^{(l)}, h_i^{(l-1)}), \tag{5}$$

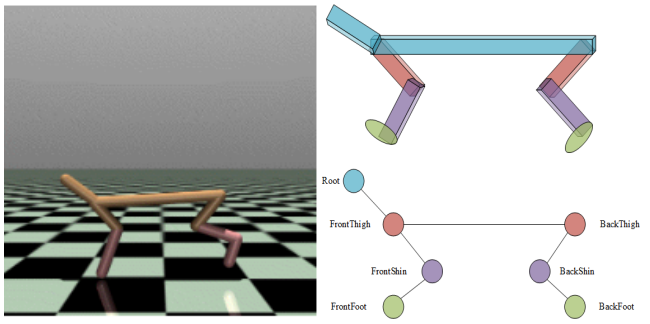
in this equation $h_i^{(l+1)}$ represents the updated feature representation of node i in the next layer.

The function g combines the current feature representation $h_i^{(l)}$ with the previous feature representation $h_i^{(l-1)}$ to generate the updated representation.

To avoid the loss of interaction features and enhance coordination capabilities between agent joints, we propose a novel feature representation approach. This approach preserves the properties of vector feature vectors while stacking them together to retain the interaction information.

$$u_i = v_i - \frac{1}{N}\sum_{j=1}^{N} v_j, \tag{6}$$

v_i represents the vector feature vector with index i. We convert each vector feature vector into a translation-invariant vector u_i by subtracting the average of all feature vectors $\frac{1}{N}\sum_{j=1}^{N} v_j$ from v_i. Here, N denotes the number of feature vectors.



(a)Mujoco (b)Format

Fig. 2. Modeling the environment in MuJoCo, where agents possess multiple hierarchical joints. Simple graph neural networks are insufficient to fully capture the interaction features among joints. Introducing subequivariance requires hierarchical classification for different joints, as depicted in this figure.

Reinforcement Algorithms and Optimization We employ the proximal policy optimization to update the policy function parameters for better adaptation to the environment. The advantage function A, which measures the relative advantage of each action, is defined as the difference between the action-value function $Q(s,a)$ and the state-value function $V(s)$.

$$A(s,a) = Q(s,a) - V(s) \tag{7}$$

The objective is to maximize the expected cumulative reward by adjusting the policy parameters θ.

$$R(\pi_\theta) = \sum_{t=0}^{T} \alpha^t r(s_t, a_t), \tag{8}$$

α represents the discount factor, t denotes the number of time steps.

At each time step t, the agent selects an action a_t based on the current state s_t and receives a reward $r(s_t, a_t)$ from the environment. The environment transitions to the next state according to the transition probability $P(s_{t+1}|s_t)$.

$$P(a^\tau|s^\tau) = \prod_{u\in\mathscr{O}} P_u(a_u^\tau|s^\tau) \tag{9}$$

$$P_u(a_u^\tau|s^\tau) = \frac{1}{\sqrt{2\pi\sigma_u^2}} \exp\left(\frac{(a_u^\tau - \mu_u)^2}{2\sigma_u^2}\right) \tag{10}$$

The agent's objective is to learn policy parameters θ that maximize the cumulative reward during interaction with the environment. This involves continuously updating the policy parameters through interactions to improve decision-making and maximize long-term cumulative reward [49].

The proximal policy optimization algorithm aims to maximize the advantage function A while constraining policy changes to ensure stability. The objective function of PPO is defined as:

$$\theta_{\text{new}} = \arg\max_\theta E\pi\theta_{\text{old}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}} \cdot A^{\pi_{\theta_{\text{old}}}}(s,a)\right], \tag{11}$$

θ_{new} represents the updated policy parameters, and θ_{old} represents the old policy parameters. $\pi_\theta(a|s)$ is the policy that selects action a given state s, while $\pi_{\theta_{\text{old}}}$ represents the policy based on the old parameters.

$A^{\pi_{\theta_{\text{old}}}}(s,a)$ is the advantage function based on the old policy, which estimates the advantage of selecting action a in state s relative to the old policy. The objective function maximizes the expected value of the ratio between the new and old policies, weighted by the advantage function.

$$\begin{aligned}\tilde{J}(\theta) = \mathbb{E}_{\pi_\theta}&\left[\sum_{\tau=0}^{\infty} \min\left(\hat{A}^\tau r^\tau(\theta), \hat{A}^\tau \operatorname{clip}\left(r^\tau(\theta), 1-\varepsilon, 1+\varepsilon\right)\right)\right] \\ &- \beta\mathbb{E}_{\pi_\theta}\left[\sum_{\tau=0}^{\infty} \mathrm{KL}\left[\pi_\theta(\cdot|s^\tau) \parallel \pi_{\theta_{\text{old}}}(\cdot|s^\tau)\right]\right] \\ &- \alpha\mathbb{E}_{\pi_\theta}\left[\sum_{\tau=0}^{\infty} \left(V_\theta(s^\tau) - V(s^\tau)^{\text{target}}\right)^2\right]\end{aligned} \tag{12}$$

The objective of policy updates in PPO is to maximize the expected ratio of action probabilities, weighted by the advantage under the old policy. This approach ensures that

the new policy gradually improves its performance while staying close to the old policy, represented by $\pi_{\theta_{\text{old}}}$ [50].

Algorithm 1 Total Algorithm

Inputs:Input the state of each joint S, the actions of each joint A and the policy function π.

Require:Update S and A, extract the feature representation of the node $h_i^{(l)}$, object-aware message $m_{ij}^{(l+1)}$ between the central node and the neighboring node, updated policy parameters θ_{new}.

Function Designing Agents and Environment(V, A, S)

$\pi \leftarrow S,A$, maping the policy function π a probability distribution and representing the probability of selecting action.

$V(s) \leftarrow S$, estimating the expected return under state.

$s,r \leftarrow \pi,S,A$, agent selects an action from the action space based on the current state and interacts with the environment by executing action until finding a policy that maximizes the expected reward.

Function Subequivariant Learning($h_i^{(l)}$, $h_j^{(l)}$, t)

$E_{ij} \leftarrow h_i,h_j$, representing the external field information between the central node i and the neighboring node j.

$m_{ij}^{(l+1)} \leftarrow h_i^{(l)},h_j^{(l)}$, using the equation 2, representing the object-aware message between the central node i and the neighboring node j in the layer indexed by $(l+1)$.

$m_{uv}^{(t)} \leftarrow m_{ij}^{(l+1)},t,f$, updating the node's state and facilitate further information propagation in subsequent propagation steps through equation 4.

Function Training and Optimization (s_t, a_t, θ, $\pi_\theta(a|s)$)

$\theta_{\text{new}} \leftarrow \theta_{\text{old}},\pi_\theta(a|s))$, the advantage function is used to estimate the advantage of selecting action a_t in state s_t relative to the old policy through equation 8.

IV. Experiments and Simulations results

In this section, we present the results of our experiments evaluating the effectiveness of the subequivariant-based neural network in improving agent motion in cooperative reinforcement learning tasks. We compare its performance with traditional graph networks and emphasize the advantages of integrating second-order invariance into the network architecture. These experiments cover various benchmark task environments to evaluate the generalizability and robustness of our model.

A. Experimental Setup

We selected multiple agents from MuJoCo, including Hopper-v2, Humanoid-v2[51], HalfCheetah-v2, Centipede-v1[10], and Walker2d-v2. The goal was to apply reinforcement learning to coordinate the joint movements of these agents. Tasks such as standing, coordinated joint manipulation, and normal locomotion were designed to test different aspects of coordination.

Traditional Graph in our experiments uses a network architecture of [512, 512], a learning rate of 3e-4, a hidden state size of 256, separate output networks tailored to each agent's coordination requirements, and 6 propagation steps.

B. Parameter Architecture and Evaluation Metrics

Experiments used an NVIDIA GeForce RTX 3080 GPU and an 8-core CPU. Each validation training session lasted around 16 hours. To ensure fair comparison, we maintained consistent batch size, training iterations, and performance metrics. The table I displays the hyperparameter settings for our modules.

TABLE I

Parameter architecture and evaluation metrics

CoordiGraph	Value Tried
Gradient clipping	0.05, 0.1, 0.2
Network Shape	$[128,128],[256,256],[512,512]$
Learning rate	1e-4, 3e-4
Hidden state size	128, 256
Size of Nodes' Hidden size	$32,64,128$
Output Network	Shared, Separate
Add Skip-connection from / to root	Yes, No
Number of Propogation Steps	$4,5,6$
Matrix embedding size	32×32, 64×64
Learning rate scheduler	adaptive, constant

C. Main Results

We adopt a reinforcement learning framework to train agents in a coordination task. By interacting with the environment, the agents receive rewards based on their actions and update their policies using policy gradient methods. We employ a variant of proximal policy optimization as the training algorithm, which has been proven effective in multi-agent reinforcement learning [52].

Our experimental results demonstrate that our method outperforms existing techniques in multi-joint robot control as shown in Table II. We observe significant improvements in coordination accuracy, exploration capability, and reward acquisition. These findings highlight the effectiveness of our approach in addressing the challenges of the Mujoco multi-joint robot control problem in Figure 3.

We conducted experiments to compare the performance of CoordiGraph and traditional graph networks in motion control tasks using reinforcement learning. The results, as depicted in the figure, consistently demonstrate the superior performance of CoordiGraph over traditional graph networks across all coordination tasks, with a notable advantage in terms of reward acquisition.

The superior performance of CoordiGraph in coordination tasks can be attributed to its incorporation of second-order variance characteristics. By effectively capturing dynamic variations in symmetry and isotropy within the joints of intelligent agents, CoordiGraph enhances coordination accuracy. In contrast, traditional graph networks, such as GNN, exhibit weaker performance in this aspect.

Furthermore, CoordiGraph showcases superior exploration and coordination capabilities, enabling it to adapt and learn more effectively in unfamiliar scenarios. In contrast, traditional graph network demonstrates weaker abilities in exploration and coordination. The advantages of CoordiGraph in motion control tasks extend beyond reward acquisition. It excels in accurately predicting actions and joint angle positions

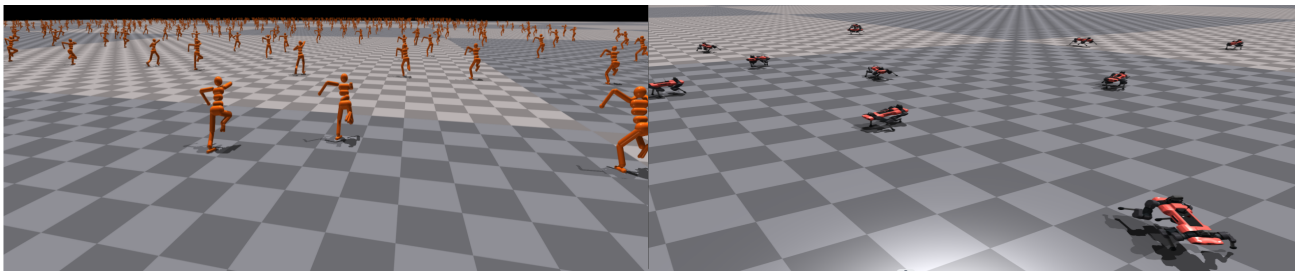


Fig. 3. We conducted large-scale training in a simulated environment, incorporating various environments and agents to ensure the generalizability and practicality of the model's performance.

of intelligent agents during task execution, facilitating precise coordination. This heightened accuracy empowers CoordiGraph to adapt more efficiently to complex environments and task requirements, ultimately enhancing task efficiency.

In the subsequent phase, we use ablation experiments to demonstrate the superiority of CoordiGraph. By surpassing existing neural networks in coordinating motion, our model validates its effectiveness in motion control.

Directionality of Intelligent Agent Motion In our experiments, we trained two CoordiGraph models under different environmental conditions: one with directionality constraints and one without. Directionality constraints limited the model's learning and exploration to specific directions, while the unconstrained model could learn and explore in all directions.

We tested both models in the same environment and compared their performance under different directionality constraint conditions. By observing the model's behavior, we evaluated the adaptability and generalization abilities of the CentipedeSix and Humanoid environments in Figure 4, providing insights into the impact of directionality constraints on learning and performance.

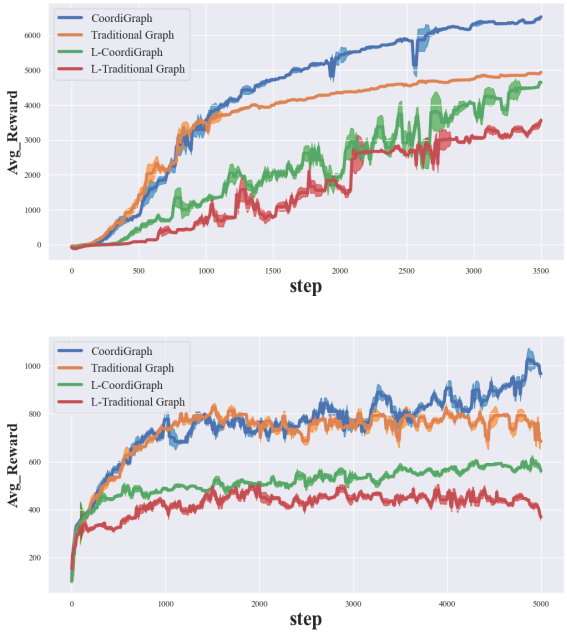


Fig. 4. Results of the dynamics of intelligent agent motion

The results show that the model trained with directionality constraints outperforms in testing. By focusing on specific behaviors and optimizing them through trial and error, the model gains a better understanding of the environment and takes actions that lead to higher rewards.

In contrast, the model trained without directionality constraints faces challenges in optimizing its behavior due to increased uncertainty and randomness during training. It may struggle to achieve the same high rewards as when directionality is controlled.

Imposing directionality constraints allows the model to concentrate on learning and optimizing specific behaviors, leading to improved learning outcomes during testing. This constraint enhances the model's understanding of the environment and enables it to take appropriate actions to maximize rewards.

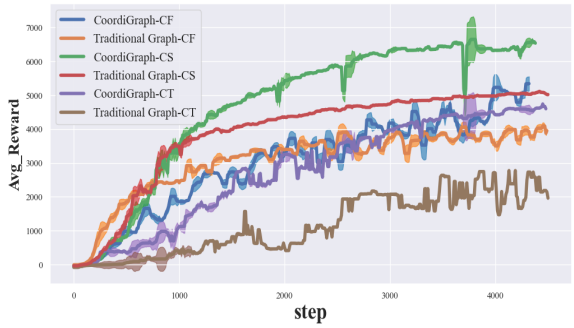


Fig. 5. Results of comprehensive analysis on the complexity of agents

Complexity of Agents: A Comprehensive Analysis In this experiment, we increased the structural complexity of agents by adding more connections in different parts. This aimed to assess CoordiGraph's generalization ability under varying levels of complexity. More connections provided additional parameters and richer feature representations, enabling the model to better understand the environment and learn complex strategies. We then tested these models with different complexities in the same environment.

By evaluating the performance and generalization ability in the same environment, we can understand how CoordiGraph performs under different levels of structural complexity in Figure 5. Analyzing agent behavior and contrasting learning outcomes facilitates evaluating their generalization capabilities.

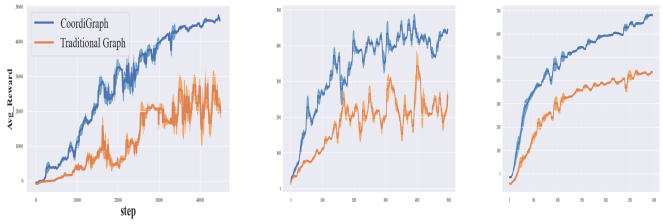
TABLE II

RESULTS OF OUR GRAPH NETWORK'S EFFICIENCY

Model	Avg_Reward						
Environment	Centipede-Four	Humanoid	HalfCheetah	Centipede-Six	Walker2D	Hopper	Centipede-Ten
CoordiGraph	5792.07	1088.36	7803.52	6325.47	4376.90	3693.25	4395.71
	5960.48	998.05	7612.81	**6732.12**	4685.21	3323.49	4632.08
	5868.34	**1110.37**	**7937.41**	6538.09	4615.75	3594.31	**4801.35**
	6042.98	1025.48	7452.94	6694.68	**4953.14**	**3774.04**	4438.92
	5586.42	1101.91	7567.58	6321.57	4831.48	3608.21	4735.86
Traditional Graph	3927.33	938.62	5364.29	4621.44	3683.17	2367.56	3808.19
	3643.29	967.53	5646.37	5012.19	3701.21	2965.28	3329.85
	4256.83	878.15	5491.42	4832.53	3358.91	2838.71	3894.24
	3847.51	908.89	4992.81	5177.47	3401.39	2547.19	3674.52
	4168.17	899.76	5513.24	4937.29	3786.58	2636.42	3752.39

The results show that as agents become more complex, the model's ability to adapt remains stronger than the baseline. This suggests that CoordiGraph greatly improve complex models' ability to adapt to different environments and tasks. The model becomes better at understanding the environment and learning complex strategies. Increased complexity improves generalization, highlighting the advantage of CoordiGraph in helping complex models adapt to diverse environments and tasks.

Generalization of Graph Neural Networks We conducted experiments comparing CoordiGraph and traditional graph network to assess their performance in generalization and reward acquisition. We tested the models' ability to adapt to unfamiliar environments by introducing them to new tasks and scenarios that were different from the training data. This allowed us to evaluate how well the models could adapt and perform in diverse and novel conditions in Figure 6.



(a)CentipedeTen (b)Walker2D (c)HalfCheetah

Fig. 6. Results on the generalization of graph neural networks

Based on our analysis, traditional graph network performs better in generalization, while CoordiGraph excels in both generalization and reward acquisition. CoordiGraph demonstrates superior adaptability and learning in unknown situations and new tasks, leading to better generalization. It also outperforms traditional graph network in reward acquisition by effectively understanding and utilizing reward signals to maximize rewards, resulting in faster learning and optimized agent coordination. In contrast, traditional graph network's performance in this area is relatively weaker, potentially leading to inaccurate action selection or suboptimal reward maximization.

Effects of Subequivariant We conducted experiments to investigate the impact of subequivariance on graph neural network models in reinforcement learning. We systematically removed subequivariant components to assess their effect on model performance. In motion coordination tasks, joint coordination behavior often exhibits symmetry and equivariance. Our goal was to understand how introducing subequivariance affects the model's ability to capture these characteristics and improve algorithmic model accuracy and performance in Figure 7.

(a)Hopper (b)CentipedeFour

Fig. 7. Results on effects of subequivariant

The experimental results show that models with subequivariance outperform models without subequivariant components in terms of coordination accuracy. This emphasizes the importance of subequivariance in enhancing coordination task accuracy. Additionally, models with subequivariance demonstrate better generalization abilities. They can adapt and learn effectively even in unfamiliar situations and new coordination tasks, leading to improved generalization. Subequivariance helps the model capture joint relationships and coordination behaviors more effectively, resulting in more stable training and control processes.

V. CONCLUSIONS

In this paper, we introduce a novel framework, CoordiGraph, which leverages the subequivariant property to address the challenges of weak inter-joint coupling in high-dimensional motion control tasks using reinforcement learning. Experimental results indicate that CoordiGraph outperforms several baseline methods in complex motion control scenarios. These findings hint at the potential of subequivariance as a method to enhance coordination in intricate motion control tasks.

VI. ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (62102241) and Shanghai Municipal Natural Science Foundation (23ZR1425400).

REFERENCES

[1] A. Heuillet, F. Couthouis, and N. Díaz-Rodríguez, "Explainability in deep reinforcement learning," *KNOWL-BASED. SYST*, vol. 214, p. 106685, 2021.

[2] M. Körber *et al.*, "Comparing popular simulation environments in the scope of robotics and reinforcement learning," *arXiv preprint arXiv:2103.04616*, 2021.

[3] G. Chen *et al.*, "Reinforcement learning control for the swimming motions of a beaver-like, single-legged robot based on biological inspiration," *ROBOT. AUTON. SYST*, vol. 154, p. 104116, 2022.

[4] W. Liu *et al.*, "Distance-directed target searching for a deep visual servo sma driven soft robot using reinforcement learning," *J. BIONIC. ENG*, vol. 17, pp. 1126–1138, 2020.

[5] P. Ladosz *et al.*, "Exploration in deep reinforcement learning: A survey," *INFORM. FUSION*, vol. 85, pp. 1–22, 2022.

[6] X. Qiu, X. Tan, Q. Li, S. Chen, Y. Ru, and Y. Jin, "A latent batch-constrained deep reinforcement learning approach for precision dosing clinical decision support," *Knowledge-Based Systems*, vol. 237, p. 107689, 2022.

[7] J. Park *et al.*, "Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning," *INT. J. PROD. RES*, vol. 59, no. 11, pp. 3360–3377, 2021.

[8] S. Chen, X. Qiu, X. Tan, Z. Fang, and Y. Jin, "A model-based hybrid soft actor-critic deep reinforcement learning algorithm for optimal ventilator settings," *Information Sciences*, vol. 611, pp. 47–64, 2022.

[9] J. Chai and M. Hayashibe, "Motor synergy development in high-performing deep reinforcement learning algorithms," *IEEE ROBOT. AUTOM. LET*, vol. 5, no. 2, pp. 1271–1278, 2020.

[10] T. Wang *et al.*, "Nervenet: Learning structured policy with graph neural networks," in *ICLR*, 2018.

[11] N. Wu, Y. Xie, and C. Hao, "Ironman: Gnn-assisted design space exploration in high-level synthesis via reinforcement learning," in *GLSVLSI*, 2021.

[12] C. S. de Witt *et al.*, "Deep multi-agent reinforcement learning for decentralized continuous cooperative control," *arXiv preprint arXiv:2003.06709*, vol. 19, 2020.

[13] P. Hart and A. Knoll, "Graph neural networks and reinforcement learning for behavior generation in semantic environments," in *IV*. IEEE, 2020.

[14] C. Shan *et al.*, "Reinforcement learning enhanced explainer for graph neural networks," *Advances in NIPS*, vol. 34, pp. 22 523–22 533, 2021.

[15] C. Blake *et al.*, "Snowflake: Scaling gnns to high-dimensional continuous control via parameter freezing," in *NIPS*, 2021, pp. 23 983–23 992.

[16] P. Zhao and Y. Liu, "Physics informed deep reinforcement learning for aircraft conflict resolution," *IEEE. T. INTELL. TRANSP*, vol. 23, no. 7, pp. 8288–8301, 2021.

[17] G. A. Castillo *et al.*, "Hybrid zero dynamics inspired feedback control policy design for 3d bipedal locomotion using reinforcement learning," in *ICRA*. IEEE, 2020.

[18] H. Wang *et al.*, "Scientific discovery in the age of artificial intelligence," *Nature*, vol. 620, no. 7972, pp. 47–60, 2023.

[19] ——, "Neural-seir: A flexible data-driven framework for precise prediction of epidemic disease," *Mathematical Biosciences and Engineering*, vol. 20, no. 9, pp. 16 807–16 823, 2023.

[20] S. Levine and et al., "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv*, 2020.

[21] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *IEEE SSCI*. IEEE, 2020.

[22] T. M. Moerland and et al., "Model-based reinforcement learning: A survey," *FOUND. TRENDS. MACH. LE.*, vol. 16, no. 1, pp. 1–118, 2023.

[23] F. Moreno-Vera, "Performing deep recurrent double q-learning for atari games," in *LA-CCI*. IEEE, 2019.

[24] M. Grillitsch and M. Sotarauta, "Trinity of change agency, regional development paths and opportunity spaces," *PROG. HUM. GEOG.*, vol. 44, no. 4, pp. 704–723, 2020.

[25] Z. Li and et al., "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *ICRA*. IEEE, 2021.

[26] A. K. Lakshmanan *et al.*, "Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot," *AUTOMAT. CONSTR.*, vol. 112, p. 103078, 2020.

[27] Y. Li *et al.*, "Constrained motion planning of free-float dual-arm space manipulator via deep reinforcement learning," *AEROSP. SCI. TECHNOL.*, vol. 109, p. 106446, 2021.

[28] W. Wang *et al.*, "A pso-optimized fuzzy reinforcement learning method for making the minimally invasive surgical arm cleverer," *IEEE ACCESS*, vol. 7, pp. 48 655–48 670, 2019.

[29] Y. Zhou, B. Li, and T. R. Lin, "Maintenance optimisation of multicomponent systems using hierarchical coordinated reinforcement learning," *RELIAB. ENG. SYST. SAFE.*, vol. 217, p. 108078, 2022.

[30] M. Schilling and A. Melnik, "An approach to hierarchical deep reinforcement learning for a decentralized walking control architecture," in *BICA*. Springer International Publishing, 2019.

[31] H. Fu *et al.*, "Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces," *arXiv*, 2019.

[32] S. Li *et al.*, "Deep implicit coordination graphs for multi-agent reinforcement learning," *arXiv*, 2020.

[33] J. Cai *et al.*, "Jolo-gcn: mining joint-centered light-weight information for skeleton-based action recognition," in *WACV*, 2021.

[34] Z. Tu *et al.*, "Joint-bone fusion graph convolutional network for semi-supervised skeleton action recognition," *IEEE. T. MULTIMEDIA.*, 2022.

[35] P. Ding and J. Yin, "Towards more realistic human motion prediction with attention to motion coordination," *IEEE. T. CIRC. SYST. VID.*, vol. 32, no. 9, pp. 5846–5858, 2022.

[36] J. Zhang *et al.*, "Graph-aware transformer for skeleton-based action recognition," *VISUAL. COMPUT.*, pp. 1–12, 2022.

[37] Y. Shao *et al.*, "Graph attention network-based multi-agent reinforcement learning for slicing resource management in dense cellular network," *IEEE T. VEH. TECHNOL*, vol. 70, no. 10, pp. 10 792–10 803, 2021.

[38] H. Dai *et al.*, "Cooperative path planning of multi-agent based on graph neural network," in *CCDC*. IEEE, 2022.

[39] J. Jiang *et al.*, "Graph convolutional reinforcement learning," *arXiv preprint arXiv:1810.09202*, 2018.

[40] D. Gammelli *et al.*, "Graph neural network reinforcement learning for autonomous mobility-on-demand systems," in *CDC*. IEEE, 2021.

[41] J. Han *et al.*, "Learning physical dynamics with subequivariant graph neural networks," in *NIPS*, 2022, pp. 26 256–26 268.

[42] V. G. Satorras, E. Hoogeboom, and M. Welling, "E (n) equivariant graph neural networks," in *ICML*. PMLR, 2021.

[43] W. Huang *et al.*, "Equivariant graph mechanics networks with constraints," *arXiv*, 2022.

[44] J. Yu, J. Liang, and R. He, "Finding diverse and predictable subgraphs for graph domain generalization," *arXiv*, 2022.

[45] H. Wang *et al.*, "Equivariant and stable positional encoding for more powerful graph neural networks," *arXiv*, 2022.

[46] T. Fu *et al.*, "Reinforced genetic algorithm for structure-based drug design," in *NIPS*, vol. 35, 2022, pp. 12 325–12 338.

[47] N. Dym and S. J. Gortler, "Low dimensional invariant embeddings for universal geometric learning," *arXiv*, 2022.

[48] K. Arulkumaran *et al.*, "Deep reinforcement learning: A brief survey," *IEEE SIGNAL. PROC. MAG.*, vol. 34, no. 6, pp. 26–38, 2017.

[49] N. Le *et al.*, "Deep reinforcement learning in computer vision: a comprehensive survey," *ARTIF. INTELL. REV.*, pp. 1–87, 2022.

[50] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *COMPUT. CHEM. ENG.*, vol. 139, p. 106886, 2020.

[51] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IROS*. IEEE, 2012.

[52] M. Carroll *et al.*, "On the utility of learning about humans for human-ai coordination," in *NIPS*, 2019.