

Learning Continuous Control with Geometric Regularity from Robot Intrinsic Symmetry

Shengchao Yan¹, Baohe Zhang¹, Yuan Zhang¹, Joschka Boedecker¹, Wolfram Burgard²

Abstract—Geometric regularity, which leverages data symmetry, has been successfully incorporated into deep learning architectures such as CNNs, RNNs, GNNs, and Transformers. While this concept has been widely applied in robotics to address the curse of dimensionality when learning from high-dimensional data, the inherent reflectional and rotational symmetry of robot structures has not been adequately explored. Drawing inspiration from cooperative multi-agent reinforcement learning, we introduce novel network structures for single-agent control learning that explicitly capture these symmetries. Moreover, we investigate the relationship between the geometric prior and the concept of Parameter Sharing in multi-agent reinforcement learning. Last but not the least, we implement the proposed framework in online and offline learning methods to demonstrate its ease of use. Through experiments conducted on various challenging continuous control tasks on simulators and real robots, we highlight the significant potential of the proposed geometric regularity in enhancing robot learning capabilities.

I. INTRODUCTION

Robots have the ability to undertake tasks that are dangerous or difficult for humans. With more degrees of freedom, they can perform increasingly complex tasks. For example, humanoid robots and quadrupedal robots can walk over challenging terrain, while robot arms and hands can achieve dexterous manipulation. However, controlling robots with a large number of degrees of freedom becomes increasingly difficult as the observation and action space grows exponentially. Although deep reinforcement learning has been employed to solve various robot control problems [1], [2], [3], [4], learning effective control strategies for these robots remains a challenging task.

Training neural networks on high-dimensional data is known to be challenging due to the curse of dimensionality [5]. To overcome this challenge, researchers have developed network architectures and incorporated various inductive biases that respect the structure and symmetries of the corresponding domains. For example, convolutional neural networks (CNNs) leverage the strong geometric prior of images by incorporating translation and rotation equivariance into the design of convolutional layers. This ensures that the extracted features move along with the original image, regardless of the direction it is shifted in. Similarly, graph neural networks (GNNs) take advantage of the geometric prior of permutation invariance in other domains to capture the relationships among objects. Overall, incorporating domain-specific inductive biases and symmetries can greatly

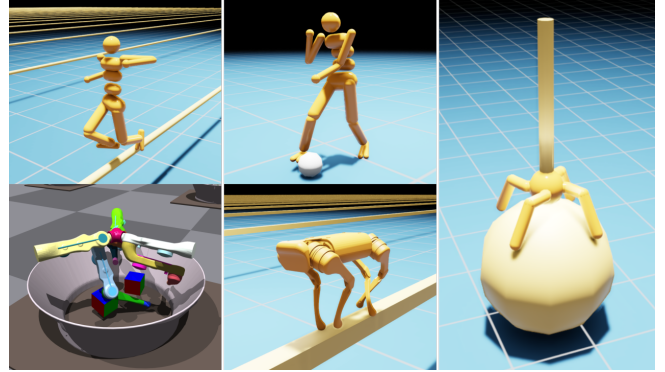


Fig. 1: Tasks challenging for current deep reinforcement learning baseline algorithms.

improve the ability of neural networks to learn from high-dimensional data.

However, in the realm of robot learning research, the potential benefits of exploiting symmetry structures present in environments, such as reflectional and rotational symmetry, remain largely unexplored. Therefore, how to combine this prior knowledge to effectively improve the existing approaches still is worth to be investigated. To bridge the research gap, we propose to reformulate the control problems using a Multi-Agent Reinforcement Learning (MARL) framework to better leverage the symmetry structures.

Specifically, we introduce the Multi-Agent with Symmetry Augmentation network structure (MASA), an architecture for designing control policies or value functions which leverage the transformation equivariance or invariance of the corresponding symmetric robot structure. Instead of learning policy and critic functions in the joint action space composed of all actuators in a robot, we divide the robot into several symmetric components and learn a policy for each of them. The critic function maps observations with or without actions from all agents to a centralized value. Additionally, we establish a connection between our proposed geometric prior and the important concept of “Parameter Sharing” in MARL, which drastically reduces the optimization space and speeds up the learning process. We demonstrate the surprising effectiveness of our approach by combining the new architecture with both online and offline model-free deep learning methods. We evaluate the proposed method on a set of challenging robot control tasks (see Fig. 1). The experimental results demonstrate that our method significantly improves the performance and data efficiency of robot control learning tasks.

The authors are with the ¹Department of Computer Science, University of Freiburg, Germany, and the ²Department of Engineering, University of Technology Nuremberg, Germany.

II. BACKGROUND AND RELATED WORK

A. Multi-Agent Reinforcement Learning (MARL)

MARL is an extended reinforcement learning method for decision-making problems, where multiple agents can interact and learn in one environment simultaneously. The common mathematical framework for MARL problems is Markov games. A Markov game is a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{O}, \mathcal{A}, P, R_i, \gamma \rangle$. \mathcal{N} is the set of all agents and \mathcal{S} is the set of states. \mathcal{O}_i and \mathcal{A}_i are observation space and action space for agent i , while $\mathcal{O} = \times_{i \in \mathcal{N}} \mathcal{O}_i$ and $\mathcal{A} = \times_{i \in \mathcal{N}} \mathcal{A}_i$ represent joint observation space and joint action space. Define $\Delta_{\mathcal{S}}$ and $\Delta_{\mathcal{A}}$ be the probability measure on \mathcal{S} and \mathcal{A} respectively. Then P is the transition probability $P(s'|s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \Delta_{\mathcal{S}}$. Each agent i maintains an individual reward function $R_i(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, and the future rewards are discounted by the discount factor $\gamma \in [0, 1]$. Let $\Pi_i = \{\pi_i(a_i|o_i) : \mathcal{O}_i \rightarrow \Delta_{\mathcal{A}_i}\}$ be the policy space for agent i , then the objective for agent i is represented as $\max_{\pi_i} \mathbb{E}_{\pi, P} \left[\sum_{t=0}^{+\infty} \gamma^t R_i(s_t, a_t) \right]$. In practice, the state space and the observation space can be identical if the observation has already fully described the system. Our paper also follows this assumption and uses observation alone.

Multi-Agent Mujoco [6] is a popular benchmark for MARL algorithms which divides a single robot into several distinct parts with separate action space. However, the state-of-the-art MARL algorithms still couldn't match the performance of the single-agent algorithms on this benchmark. Different from their work, in which they arbitrarily divide robots into parts and ignore the geometric structures of the robots, we leverage ideas from geometric regularity during the MARL training and our results show that MARL can outperform single-agent algorithms by a substantial margin.

B. Symmetry in Robot Learning

In the robot learning domain, two groups of symmetric structures have been used to improve performance and learning efficiency. 1) **Extrinsic Symmetry**: By extrinsic symmetry we refer to the symmetries existing in the exteroceptive sensors of the robot such as camera input. Some works [7], [8], [9], [10] have been proposed to integrate these symmetries into system identification approaches with neural networks, especially CNN-structured ones. These methods can largely improve the performance for manipulation tasks, but they are mostly used for manipulation tasks with image input and grippers without roll-pitch movement. Learning symmetry in the latent space directly from data [11] is still limited to representation learning from images. 2) **Intrinsic Symmetry**: Different from extrinsic symmetries, intrinsic symmetries mostly naturally come from the physical constraints in the control system. For example, a humanoid robot control task exhibits reflectional symmetry. A symmetric control policy on such robot is usually more natural and effective. A data-augmentation method [12] is proposed to improve reinforcement learning methods for rotation invariant locomotion. To directly incorporate symmetry in the policy, it is also proposed to numerically construct

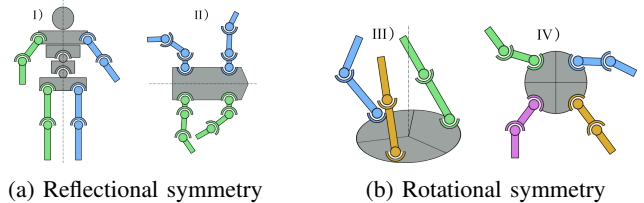


Fig. 2: **Agent partitioning considering symmetry structures**: Humanoid and Cheetah robots split into left and right parts by reflectional symmetry; TriFinger and Ant robots split into three and four parts by rotational symmetry, where each part is controlled individually by a dedicated agent. The central part (grey) is controlled by all agents.

equivariant network layers [13], [14]. However, additional calculation is required to design the network even if the domain specific transformation is given, leading to a relatively complex procedure. Moreover, the policy network of [13] only considers a pole balancing task with discrete action and [14] has no experiments on robot control policy learning. Researchers investigate four different methods to encourage symmetric motion of bipedal simulated robots [15]. They are implemented via specific policy network, data augmentation or auxiliary loss function. Even though the robots' motions become natural-looking, they do not show a major improvement on different tasks. The policy network method in [15] is similar to ours. But instead of a specific network merely for locomotion tasks with reflectional symmetry, we propose a generic equivariant policy network for both reflectional and rotational symmetries, the predominant symmetry features in robotic systems and animal biology. Moreover, we approach the control task from the viewpoint of multi-agent systems. Finally, we achieve substantial performance improvements in our experiments by reducing the policy search space.

III. SINGLE ROBOT CONTROL AS MARL

Instead of learning a single-agent policy to control the whole robot, which will lead to a large observation-action space that is difficult to optimize, we introduce multiple agents that are responsible for each individual component of the robot inspired by MARL. We further propose a framework driven by the presence of symmetry structures in many robots and exploit such inductive biases to facilitate the training by applying parameter sharing techniques.

Our method consists of (1) identifying the geometric structures of different robots and dividing single robots into multiple parts accordingly; (2) reformulating the control problem under a MARL framework; (3) optimizing policies with Parameter Sharing.

A. Dividing Single Robots into Multiple Parts

Previous research [6] divides a single robot into multiple parts to evaluate the performance of MARL methods. However, its irregular partitioning makes it hard for multi-agent methods to compete with the single-agent methods. In this paper, we instead take advantage of the symmetry structures of robots.

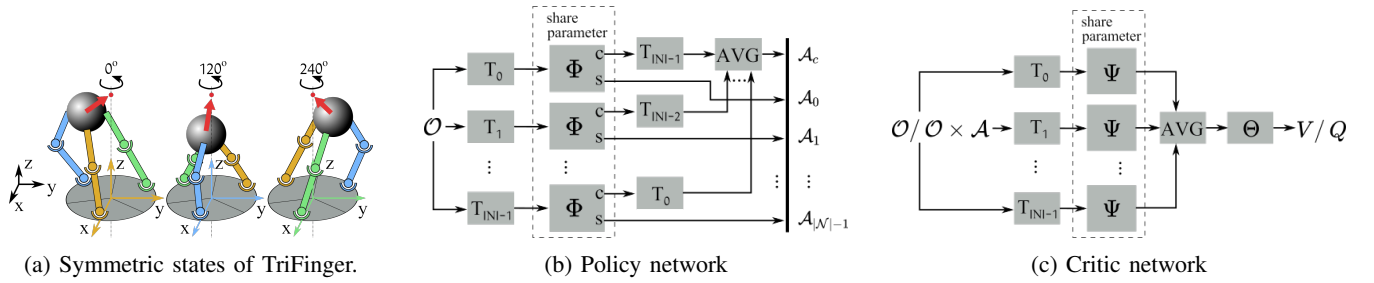


Fig. 3: a) TriFinger robot moves an object towards a target position. The black coordinate system is the global system, while the colored ones are local systems. The red arrow represents the desired moving direction of the manipulated object. Note that the actions of different body parts should be equivariant with regard to the rotations. b) Equivariant policy network with parameter Φ . **c** and **s** stand for central and symmetric actions. c) Invariant critic network with parameter Ψ, Θ .

As shown in Fig. 2a, robots with reflectional symmetry can be partitioned into left (blue), right (green) and a central part (grey). The robots with rotational symmetry in Fig. 2b are partitioned into parts with the same number of symmetric limbs (colour) and a central part (grey). For a robot with any of these symmetric structures, we split the whole robot’s original observation-action space $\mathcal{O} \times \mathcal{A}$ by $\mathcal{O} = \mathcal{O}_c \times \prod_{i \in \mathcal{N}} \mathcal{O}_{s,i}$ and $\mathcal{A} = \mathcal{A}_c \times \prod_{i \in \mathcal{N}} \mathcal{A}_{s,i}$. \mathcal{O}_c represents the central observation space, which consists of measurements that do not have symmetric counterparts, such as the position, orientation, velocity and joints of the torso, target direction, or states of the manipulated objects. Raw sensor data such as images and point clouds may also belong to central observation. $\mathcal{O}_{s,i}$ corresponds to symmetric observation spaces, whose measurements may include joint positions and velocities from the limbs, contact sensor measurements of the feet or fingers, and so on. The symmetric observation spaces are the same for any $i \in \mathcal{N}$ due to the robots’ symmetric property. \mathcal{A}_c and $\mathcal{A}_{s,i}$ are the action spaces for central (e.g., humanoid robot’s pelvis and twist) and symmetric (e.g., limbs) robot parts.

B. Multi-Agent Reinforcement Learning Formulation

Assume the original observation and action of the whole robot be $o \in \mathcal{O}$ and $a \in \mathcal{A}$ respectively and the number of agents $|\mathcal{N}|$, equal to the number of symmetry parts of the robots. For each agent $i \in \mathcal{N}$, there is a unique transformation function $T_i \in \mathcal{T}$ to obtain its own observation $o_i = T_i(o)$, where \mathcal{T} is a set of symmetry transformation functions for observations or actions defined by the corresponding symmetric structure. We describe the transformation functions later in this section. Each agent generates the local action a_i , consisting of $a_{c,i} \in \mathcal{A}_c$ and $a_{s,i} \in \mathcal{A}_{s,i}$ for central and symmetric actions, by its own policy network. Finally, the whole robot’s action a is recovered by gathering all symmetric actions $a_{s,i}$ and merging all central actions $a_{c,i}$ into a_c . Regarding the reward function, our formulation follows the cooperative MARL setup, where R_i for all $i \in \mathcal{N}$ are identical at every time step. This shared reward is calculated by a task-related reward function $R(o, a)$ which depends on the whole robot’s observation and action.

We take the TriFinger robot in Fig. 3a as an example to

explain the transformation set \mathcal{T} for rotational symmetry. First, we define a local coordinate system for each of the three agents, with the origin at the center of the robot base, z axis along the robot symmetry axis and x axis pointing to the base joint of the corresponding limb. Then, we arbitrarily select an agent as the base agent. Here we take the yellow one. The coordinate system of the base agent is used as the global system and the robot observation o should be converted into it, resulting in T_0 as identity transformation for the base agent. We further describe the transformation function of other agents. Different observation components are categorized into different groups by $o = [o_{inv}, o_v]$, where o_{inv} stands for quantities invariant under the symmetry transformation, such as robot id or the delay of control systems, and o_v for the variant ones. The basic idea of the transformation of agent i is to rotate the whole environment so that its local coordinate system overlaps with the global system. As a result of the rotation, coordinate-system-irrelevant quantities of the three fingers shift circularly, and other variant values are changed by the rotation transformation. The same rules apply for the action space transformation due to the symmetric robot structure. Note that some observation quantities have to be both shifted and transformed, such as the fingertip position. Given the observation $o = [t_{delay}, \alpha_0, \alpha_1, \alpha_2, p_{object}]$, the symmetric observation of agent i can be calculated by $T_i(o) = [t_{delay}, \alpha_i, \alpha_{(i+1) \bmod 3}, \alpha_{(i+2) \bmod 3}, R_i(p_{object})]$, where t_{delay} is the control delay, α_i is the joints angle position of finger i , p_{object} is the object position, and R_i is the corresponding rotation. The transformation function for reflectional symmetry is defined in a similar way. The only difference is that the local coordinate systems are reflected to overlap with the global system instead of being rotated.

We apply our method to both online and offline learning algorithms. To optimize the policies with interaction with the environment, we adopt the multi-agent version of Proximal Policy Optimization (PPO) [16] methods. PPO is a popular model-free actor-critic reinforcement learning algorithm in different domains [2], [4], [17] for its stability, good performance and ease of implementation. Its multi-agent version also achieves competitive performance on different MARL benchmarks [18], [19]. For offline settings where the

agent learns from fixed dataset without interacting with the environment, we implement the proposed framework with Behavior Cloning (BC) and Implicit Q-Learning (IQL) [20], a state-of-the-art offline reinforcement learning algorithm.

C. Geometric Regularization

Parameter Sharing has been considered as a crucial element in MARL for efficient training [21]. By enabling agents to share parameters in their policy networks, parameter sharing not only facilitates scalability to a large number of agents but also enables agents to leverage shared learned representations, leading to reduced training time and improved overall performance. However, it is revealed that indiscriminately applying parameter sharing could hurt the learning process [22]. Successful parameter sharing relies on the presence of homogeneous agents as a vital requirement. In other words, agents should execute the same action if they are given the same observation. For our method, this is realized by the symmetry transformation.

Take the task in Fig. 3a as an example, where the manipulator with three symmetrically aligned fingers has to move the sphere towards a target position. If the whole system is rotated by 120° or 240° around the z axis of the robot base, under the optimal policy, the actions should also shift circularly among the three fingers. Given the whole robot's observation o , this relationship can be denoted by:

$$A_{s,j}(T_i(o)) = A_{s,i}(T_j(o)), \quad A_c(T_i(o)) = T_i(A_c(o)) \quad (1)$$

where $A_{s,j}$ is the symmetric action of the j -th agent, A_c is the central action, which exists for Humanoid robot, T_i is the symmetry transformation of agent i . Note that the corresponding robot parts of agents can be defined arbitrarily. It does not influence the equivariance/invariance. Regarding the value function $V : \mathcal{O} \rightarrow \mathbb{R}$ in RL algorithms, it possesses the invariance: $V(T_i(o)) = V(T_j(o)), \forall i, j \in \mathcal{N}$, which also holds for the Q function $Q : \mathcal{O} \times \mathcal{A} \rightarrow \mathbb{R}$.

Based on Eq. 1, we design the multi-agent policy and critic network structures in Fig. 3b, 3c. Agent i gets a transformed observation $T_i(o)$ as the input of the policy network, the output action value consists of $a_{c,i}$ and $a_{s,i}$. The central joints are controlled by the mean action over all agents' output $a_{c,i}$, while $a_{s,i}$ will be used as the action to take for the robot part i . The policy network parameters are shared among agents. The critic network gets the observations with or without actions from all agents as input. The input first goes through the shared feature learning layers in the value network. Then the latent features are merged by a set operator *mean*. The critic value is finally calculated with the merged feature.

The proposed policy network is equivariant with respect to symmetric transformations we consider in this work, while the critic network is an invariant function. By sharing the parameters Φ and Ψ among all agents, we are able to incorporate the geometric regularization and reduce the dimension of the observation-action space.

Proof of the network equivariance/invariance. At the beginning we summarize the properties of the symmetry transformations in this work. They are:

- commutative: $T_j(T_i(o)) = T_{i+j}(o) = T_i(T_j(o))$
- distributive: $T_j(T_i(o) + T_k(o)) = T_j(T_i(o)) + T_j(T_k(o))$
- cyclic: $T_i(o) = T_{i+|\mathcal{N}|}(o)$

The equivariance of the policy for symmetric actions in Eq. 1 is proved as follows:

$$A_{s,j}(T_i(o)) = \Phi_s(T_{j+i}(o)) = A_{s,i}(T_j(o))$$

The equivariance for the central action is proved as follows:

$$\begin{aligned} A_c(T_i(o)) &= \frac{1}{|\mathcal{N}|} \sum_{j=0}^{|\mathcal{N}|-1} T_{|\mathcal{N}|-1-j}(\Phi_c(T_j(T_i(o)))) \\ &= \frac{1}{|\mathcal{N}|} \sum_{j=|\mathcal{N}|-i}^{2|\mathcal{N}|-i-1} T_{|\mathcal{N}|-1-j}(\Phi_c(T_{i+j}(o))) \\ &= \frac{1}{|\mathcal{N}|} \sum_{k=|\mathcal{N}|}^{2|\mathcal{N}|-1} T_{|\mathcal{N}|-1-k}(\Phi_c(T_k(o))) \\ &= \frac{1}{|\mathcal{N}|} \sum_{k=0}^{|\mathcal{N}|-1} T_i(T_{|\mathcal{N}|-1-k}(\Phi_c(T_k(o)))) \\ &= T_i\left(\frac{1}{|\mathcal{N}|} \sum_{k=0}^{|\mathcal{N}|-1} T_{|\mathcal{N}|-1-k}(\Phi_c(T_k(o)))\right) \\ &= T_i(A_c(o)) \end{aligned}$$

The invariance of the value network is proved as follows:

$$\begin{aligned} V(T_i(o)) &= \Theta\left(\frac{1}{|\mathcal{N}|} \sum_{j=0}^{|\mathcal{N}|-1} \Psi(T_j(T_i(o)))\right) \\ &= \Theta\left(\frac{1}{|\mathcal{N}|} \sum_{j=|\mathcal{N}|-i}^{2|\mathcal{N}|-i-1} \Psi(T_{i+j}(o))\right) \\ &= \Theta\left(\frac{1}{|\mathcal{N}|} \sum_{k=0}^{|\mathcal{N}|-1} \Psi(T_k(o))\right) = V(o) = V(T_j(o)) \end{aligned}$$

The invariance of the Q network can be proved in the same way with action a concatenated to the observation o . ■

IV. EXPERIMENTS

We evaluate our method in different tasks to clarify the following concerns: 1) Does the multi-agent framework incorporated with robots' intrinsic symmetry improve the performance and data-efficiency on robot learning tasks? 2) Can different learning paradigms benefit from this framework? 3) Is the proposed method applicable also to real-world problems?

A. Experiments with Online Reinforcement Learning

Previous robotic control benchmarks [23] evaluate algorithms on fundamental tasks, such as controlling agents to walk. The movements in these tasks are limited and it's relatively easy to learn a good policy. In this work, we adopt several more challenging robotic control tasks, where it is difficult for current state-of-the-art online algorithms to achieve good performance. The tasks are shown in Fig. 1:

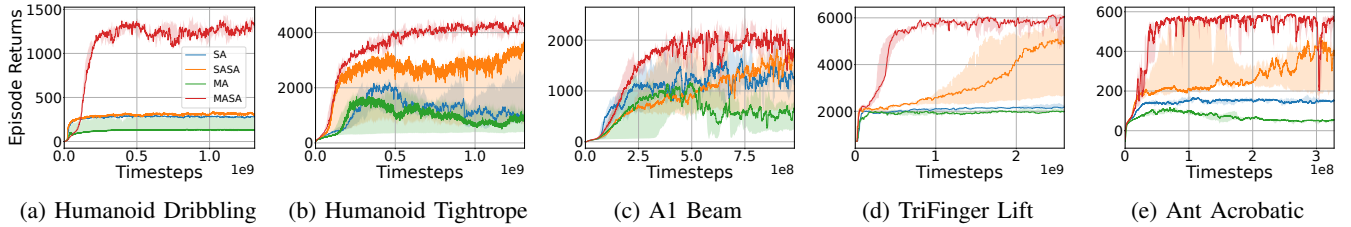


Fig. 4: Learning curves on robot control tasks. The x-axis is environment time steps and the y-axis is episodic returns during training. All graphs are plotted with median and 25%-75% percentile shading across five random seeds.

Humanoid Tightrope: The agent learns to control a humanoid robot to walk on a tightrope. The robot has 21 controllable motors. The tightrope is extremely narrow with a diameter of only 10 cm.

Humanoid Dribbling: The humanoid robot learns to dribble along routes with changing direction. Compared with the tightrope task, the observation space is augmented with features of the ball.

A1 Beam: The agent controls the quadruped robot Unitree A1 [24] to walk on a balance beam with width of 10 cm following a predefined speed. Considering the width of A1 and the balance beam, it is much harder than walking on the ground.

TriFinger Lift: TriFinger [25] is a 3-finger manipulator for learning dexterity. The goal is to move a cube from a random initial pose to an arbitrary 6-DoF target pose. The environment is the same as that in IssacGymEnvs [4], except that we remove the auxiliary penalty for finger movement to increase the difficulty of the task.

Ant Acrobatic: The ant robot learns to do complex acrobatics (e.g., heading a pole) on a ball, which extremely challenges the ability of agents to maintain balance.

All experiments are carried out based on the NVIDIA Isaac Gym [26] robotics simulator.

1) *Baselines:* For each task, we compare our method, named as Multi-Agent with Symmetry Augmentation (MASA), with three baselines. The first baseline is Single-Agent (SA), which treats the robot as a single agent and optimizes policy for the joint action space. This baseline can provide an intuitive comparison of our proposed framework to previous classic reinforcement learning works. The state space is kept the same as MASA’s for a fair comparison. The second baseline is Single-Agent with Symmetry Augmentation (SASA). It follows the SA’s setup and is augmented with a symmetry loss [15]. Specifically, for any received observation o , we calculate its symmetric representation $T_i(o)$. We regulate the policy function π and the value function V in PPO with extra symmetry losses by minimizing $\|T_i(A(o)) - A(T_i(o))\|_2$ and $|V(o) - V(T_i(o))|$, where A and V are the gathered action and critic value of the agent. The third baseline is Multi-Agent without Symmetry Augmentation (MA). It uses the same architecture with parameter sharing as MASA. However, it does not involve the transformations in Fig. 3b 3c. Thus the geometric regularity of symmetry is ignored, which follows the previous research [6]. We con-

catenate a one-hot id encoding to each agent’s observation as a common operation for non-homogeneous agents.

2) *Results:* Figure 4 presents the average return of all methods on different tasks during training. The proposed method MASA significantly outperforms other baselines across all 5 tasks. Further, the advantages over other baselines rise with the increasing difficulties of the task, which can be indicated by the increased number of joints, the extended state dimension and the enlarged state space in the task. Humanoid Tightrope and Humanoid Football control the same robot. However, in the tightrope task, the robot only needs to walk forward, while the football task involves random turns and manipulating an external object, so that other baselines can hardly learn meaningful behaviours.

By comparing the results of MASA, MA and SASA, we could observe that both of the two factors in MASA, multi-agent framework and symmetry structure, play an important role. Utilizing symmetry data structure alone (SASA) can gradually learn to solve a few tasks but with apparently lower data efficiency. Because the optimization space is not reduced and thus larger than that of MASA method. The multi-agent structure itself (MA) cannot guarantee meaningful results at all, which follows the criticism of naively sharing parameters among non-homogeneous agents [22].

In the Humanoid Dribbling task, MASA initially underperforms compared to other methods. This is because the baselines prioritize self-preservation and struggle to find a policy that balances dribbling and staying alive. By focusing on avoiding falling down and kicking the ball too far away, they learn to stand still near the ball while disregarding the rewards associated with ball movement. Consequently, the baseline agents survive longer at the beginning, resulting in higher returns compared to MASA.

B. Experiments with Offline Reinforcement Learning

Although reinforcement learning enables agent to learn from interactions with the environment, collecting samples during training could be inefficient or unsafe [27]. Offline learning solves this problem by learning from fixed dataset. To demonstrate our method’s generalizability, we also applied it to behavior cloning and implicit Q-learning, and evaluate it on both a simulator and real robots.

1) *System Setup:* The work [28] publishes large diverse datasets collected on a Pybullet simulator and a real robot cluster for benchmarking offline reinforcement learning algorithms. The simulator is provided and the robot cluster

Datasets	data	BC			IQL		
		SA	SASA-data	MASA	SA	SASA-data	MASA
Sim-Expert	0.87	0.64 ± 0.00	0.29 ± 0.13	0.71 ± 0.05	0.47 ± 0.06	0.37 ± 0.13	0.84 ± 0.02
Sim-Half-Expert	0.88	0.64 ± 0.02	0.37 ± 0.08	0.75 ± 0.05	0.04 ± 0.01	0.10 ± 0.05	0.78 ± 0.06
Sim-Weak&Expert	0.5	0.16 ± 0.04	0.10 ± 0.05	0.34 ± 0.07	0.24 ± 0.05	0.20 ± 0.07	0.55 ± 0.04
Sim-Mixed	0.68	0.01 ± 0.01	0.01 ± 0.01	0.23 ± 0.10	0.00 ± 0.00	0.03 ± 0.02	0.48 ± 0.09
Real-Expert	0.66	0.27 ± 0.09	0.13 ± 0.15	0.52 ± 0.18	0.29 ± 0.09	0.16 ± 0.18	0.61 ± 0.18
Real-Half-Expert	0.68	0.15 ± 0.01	0.13 ± 0.14	0.41 ± 0.25	0.12 ± 0.06	0.11 ± 0.13	0.61 ± 0.19
Real-Weak&Expert	0.40	0.02 ± 0.04	0.05 ± 0.09	0.26 ± 0.20	0.11 ± 0.13	0.17 ± 0.14	0.30 ± 0.24
Real-Mixed	0.42	0.00 ± 0.01	0.04 ± 0.09	0.06 ± 0.08	0.03 ± 0.02	0.03 ± 0.08	0.22 ± 0.15

TABLE I: Success rate on the TriFinger-Lift datasets. Average and standard deviation over five training seeds. ‘data’ denotes the mean over the dataset.

can be accessed remotely for evaluating learned policies. We choose the more complex task **Lift** in the dataset to show the potential of our method. The task is basically the same as *TriFinger Lift* described in Sec. IV-A.

2) *Baselines*: We evaluate MASA against two baselines: SA and SASA-data. The first baseline is the original offline algorithm treating the robot as a single agent and learn policies for the joint action space. The SASA-data baseline is different from SASA in Sec. IV-A, which add auxiliary loss functions to the policy and critic learning. It first augments the offline dataset with symmetric transitions $(T_i(o), T_i(a))$. Then an agent is trained with SA method on the larger dataset. We do not benchmark against the naive multi-agent method MA due to its poor performance shown in Sec. IV-A.

3) *Evaluation*: The evaluation results for the TriFinger Lift task on both simulator and real-robots are summarized in Table I. The values for SA come from the benchmark paper [28]. To keep fairness, we use the default hyperparameters for training all policies. The evaluation process are also kept unchanged. The policies trained on simulation datasets are evaluated in the corresponding simulator. 100 episodes are carried out for each seed and algorithm. Those trained on real-robot datasets are evaluated with the real-robot cluster. We run 6 episodes for each seed and algorithm. The final success rate values are averaged across five training seeds, which are also the same as those in the benchmark paper.

Our method outperforms the baselines over all datasets with a significant margin without any hyperparameter fine-tuning. For more than half of the datasets, the MASA agents achieve success rates close to or even better than that of the data collection policy. We observe two interesting results: I) The relative performance gain of MASA on most real-robot datasets is higher than that on simulation datasets. This can be explained as a result of the enhanced domain randomization with MASA. The robot in simulator is perfect and requires no domain randomization. The real robots, however, have different characteristics for different robots and fingers. Assuming k different robots are in the dataset, a MASA agent learns the experience from $3 \times k$ fingers due to the shared policy. II) The SASA-data agents underperform the SA agents. We think the performance drop is caused by ambiguous regression targets introduced by data augmentation. Given two samples (o, a) and (o', a') in the original dataset,

where $o' \approx T_1(o), a' \neq T_1(a)$, the resulting augmented data elements would be $(o, a), (T_1(o), T_1(a)), (T_2(o), T_2(a))$ and $(o, T_2(a')), (T_1(o), a'), (T_2(o), T_1(a'))$, which means different actions for similar observations. Since the data collection policy is not trained with a symmetric policy, such multi-modality in control sequences can be introduced by symmetric transformation. As a result, BC agents tend to learn out-of-distribution actions with such ambiguous regression targets. This problem is less severe for offline reinforcement learning algorithms like IQL, because they often are equipped with regularizers to avoid out-of-distribution actions.

V. CONCLUSION AND LIMITATIONS

This paper introduced a novel approach that incorporates the robot’s intrinsic symmetry into an agent’s policy and critic networks. Through our unique network architecture, MASA, we demonstrated remarkable success in learning robot control tasks, integrating our method with PPO, BC, and IQL from different learning paradigms. Despite the imperfections and variances in real-world robots due to mounting and manufacturing tolerances, our approach still enhances learning algorithms, even leveraging these imperfections to its advantage. Although the implementation of our method necessitates specific domain knowledge, such as understanding the robot structure and the transformation operations, it provides a significant contribution to robot learning in complex tasks. Furthermore, MASA serves as a valuable blueprint for developing robots with increased degrees of freedom, all the while keeping the complexity of observation-action space manageable. Future work offers promising avenues, including the exploration of additional symmetric structures and the automation of identifying robots’ intrinsic symmetries. The question how varying degrees of symmetry imperfection affect our method’s performance is also an interesting aspect for future work.

ACKNOWLEDGEMENT

We extend our gratitude to the Max Planck Institute for Intelligent Systems in Tübingen, Germany, for providing the TriFinger robots, encompassing software, hardware, and datasets. Special thanks to Nico Gürtler, whose support and invaluable discussions significantly contributed to our experiments.

REFERENCES

- [1] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” in *Proc. of the International Conference on Learning Representations (ICLR)*, 2016.
- [2] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning robust perceptive locomotion for quadrupedal robots in the wild,” *Science Robotics*, vol. 7, no. 62, 2022.
- [3] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, “Daydreamer: World models for physical robot learning,” in *Proc. of Conference on Robot Learning (CoRL)*, 2023, pp. 2226–2240.
- [4] A. Allshire, M. Mittal, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg, “Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger,” in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 11 802–11 809.
- [5] M. M. Bronstein, J. Bruna, T. Cohen, and P. Velickovic, “Geometric deep learning: Grids, groups, graphs, geodesics, and gauges,” *CoRR*, vol. abs/2104.13478, 2021.
- [6] B. Peng, T. Rashid, C. Schroeder de Witt, P.-A. Kamienny, P. Torr, W. Böhmer, and S. Whiteson, “Facmac: Factored multi-agent centralised policy gradients,” in *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021, pp. 12 208–12 221.
- [7] D. Wang, M. Jia, X. Zhu, R. Walters, and R. Platt, “On-robot learning with equivariant models,” in *Proc. of Conference on Robot Learning (CoRL)*, 2022.
- [8] X. Zhu, D. Wang, O. Biza, G. Su, R. Walters, and R. Platt, “Sample efficient grasp learning using equivariant models,” in *Proc. of Robotics: Science and Systems (RSS)*, 2022.
- [9] D. Wang and R. Walters, “So (2) equivariant reinforcement learning,” in *Proc. of the International Conference on Learning Representations (ICLR)*, 2022.
- [10] D. Wang, R. Walters, X. Zhu, and R. Platt, “Equivariant q learning in spatial action spaces,” in *Proc. of Conference on Robot Learning (CoRL)*, 2022, pp. 1713–1723.
- [11] A. K. Mondal, V. Jain, K. Siddiqi, and S. Ravanbakhsh, “Eqr: Equivariant representations for data-efficient reinforcement learning,” in *Proc. of the International Conference on Machine Learning (ICML)*, 2022.
- [12] A. Mavalankar, “Goal-conditioned batch reinforcement learning for rotation invariant locomotion,” *CoRR*, vol. abs/2004.08356, 2020.
- [13] E. Van der Pol, D. Worrall, H. van Hoof, F. Oliehoek, and M. Welling, “Mdp homomorphic networks: Group symmetries in reinforcement learning,” in *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2020, pp. 4199–4210.
- [14] D. F. O. Apraez, M. Martín, A. Agudo, and F. Moreno, “On discrete symmetries of robotics systems: A group-theoretic and data-driven analysis,” in *Proc. of Robotics: Science and Systems (RSS)*, 2023.
- [15] F. Abdolhosseini, H. Y. Ling, Z. Xie, X. B. Peng, and M. Van De Panne, “On learning symmetric locomotion,” in *Proc. of Motion, Interaction and Games*, 2019.
- [16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017.
- [17] S. Yan, T. Welschehold, D. Büscher, and W. Burgard, “Courteous behavior of automated vehicles at unsignalized intersections via reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 191–198, 2021.
- [18] C. Yu, A. Velu, E. Vinitzky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, “The surprising effectiveness of ppo in cooperative multi-agent games,” in *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 24 611–24 624.
- [19] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, and S. Whiteson, “Is independent learning all you need in the starcraft multi-agent challenge?” *CoRR*, vol. abs/2011.09533, 2020.
- [20] I. Kostrikov, A. Nair, and S. Levine, “Offline reinforcement learning with implicit q-learning,” in *Proc. of the International Conference on Learning Representations (ICLR)*, 2022.
- [21] J. K. Gupta, M. Egorov, and M. Kochenderfer, “Cooperative multi-agent control using deep reinforcement learning,” in *Proc. of Autonomous Agents and Multiagent Systems: Workshops*, 2017, pp. 66–83.
- [22] F. Christianos, G. Papoudakis, M. A. Rahman, and S. V. Albrecht, “Scaling multi-agent reinforcement learning with selective parameter sharing,” in *Proc. of the International Conference on Machine Learning (ICML)*, 2021.
- [23] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, “dm_control: Software and tasks for continuous control,” *Software Impacts*, vol. 6, p. 100022, 2020.
- [24] U. AI, “Unitree. a1: More dexterity, more possibility, 2018,” <https://www.unitree.com/a1/>, Jan. 2018.
- [25] M. Wuthrich, F. Widmaier, F. Grimmering, S. Joshi, V. Agrawal, B. Hammoud, M. Khadiv, M. Bogdanovic, V. Berenz, J. Viereck, M. Naveau, L. Righetti, B. Schölkopf, and S. Bauer, “Trifinger: An open-source robot for learning dexterity,” in *Proc. of Conference on Robot Learning (CoRL)*, J. Kober, F. Ramos, and C. J. Tomlin, Eds., vol. 155, 2020, pp. 1871–1882.
- [26] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, “Isaac gym: High performance GPU based physics simulation for robot learning,” in *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2021.
- [27] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Goyal, and T. Hester, “An empirical investigation of the challenges of real-world reinforcement learning,” *CoRR*, vol. abs/2003.11881, 2020.
- [28] N. Gürtler, S. Blaes, P. Kolev, F. Widmaier, M. Wuthrich, S. Bauer, B. Schölkopf, and G. Martius, “Benchmarking offline reinforcement learning on real-robot hardware,” in *Proc. of the International Conference on Learning Representations (ICLR)*, 2023.

APPENDIX

A. Extra Experimental Setups

1) *Hyperparameters*: Each baseline is run with 5 random seeds. All experiments are carried out on GPU card NVIDIA rtxA6000 and rtx3080 GPU. The hyperparameters of all baselines are consistent for a fair comparison. The detailed values can be accessed in Table II.

2) *Tasks Details*:

a) *Humanoid Tightrope*: In this task, the agent learns to control a humanoid robot to walk on a tightrope. The humanoid robot has 21 controllable motors. The tightrope is extremely narrow with a diameter of only 10 cm, which challenges the efficiency of learning algorithms. The agent is rewarded with a forward speed on the tightrope and a proper posture. At each non-terminating step, the reward $r = w_v \times r_v + w_{\text{alive}} \times r_{\text{alive}} + w_{\text{up}} \times r_{\text{up}} + w_{\text{heading}} \times r_{\text{heading}} + w_{\text{action}} \times r_{\text{action}} + w_{\text{energy}} \times r_{\text{energy}} + w_{\text{lateral}} \times r_{\text{lateral}}$, where

- r_v is the robot’s forward velocity, $w_v = 1.0$;
- $r_{\text{alive}} = 1$, $w_{\text{alive}} = 2.0$;
- $r_{\text{up}} = 1$ if $e_{\text{up},z} > 0.93$, where e_{up} is the basis vector of torso’s z axis in the global coordinate system, otherwise the value is 0, $w_{\text{up}} = 0.1$;
- $r_{\text{heading}} = e_{\text{forward},x}$, where e_{forward} is the basis vector of torso’s x axis in global coordinate system, $w_{\text{forward}} = 0.1$;
- $r_{\text{action}} = \|a\|_2^2$, where a is joints action, $w_{\text{action}} = -0.01$
- r_{energy} is the joints power consumption, $w_{\text{energy}} = -0.05$
- $r_{\text{lateral}} = v_{\text{torso},y}$ is the penalty for lateral velocity, $w_{\text{lateral}} = -1.0$

The reward is -1 for termination step. The action is the force applied to all joints.

TABLE II: Hyperparameters of all experiments.

HYPERPARAMETERS	HUMANOID TIGHTROPE	HUMANOID FOOTBALL	TRIFINGER MOVE	A1 BEAM	ANT ACROBATIC
BATCH SIZE	4096×32	4096×32	16384×16	4096×24	4096×16
MIXED PRECISION	TRUE	TRUE	FALSE	TRUE	TRUE
NORMALIZE INPUT	TRUE	TRUE	TRUE	TRUE	TRUE
NORMALIZE VALUE	TRUE	TRUE	TRUE	TRUE	TRUE
VALUE BOOTSTRAP	TRUE	TRUE	TRUE	TRUE	TRUE
NUM ACTORS	4096	4096	16384	4096	4096
NORMALIZE ADVANTAGE	TRUE	TRUE	TRUE	TRUE	TRUE
GAMMA	0.99	0.99	0.99	0.99	0.99
TAU	0.95	0.95	0.95	0.95	0.95
E-CLIP	0.2	0.2	0.2	0.2	0.2
ENTROPY COEFFICIENT	0.0	0.0	0.0	0.0	0.0
LEARNING RATE	5.E-4	5.E-4	3.E-4	3.E-4	3.E-4
KL THRESHOLD	0.0008	0.0008	0.0008	0.0008	0.0008
TRUNCATED GRAD NORM	1.0	1.0	1.0	1.0	1.0
HORIZON LENGTH	32	32	16	24	16
MINIBATCH SIZE	32768	32768	16384	32768	32768
MINI EPOCHS	5	5	4	5	4
CRITIC COEFFICIENT	4.0	4.0	4.0	2.0	2.0
MAX EPOCH	10K	10K	10K	10K	5K
POLICY NETWORK	[400,200,100]	[400,200,100]	[256,256,128,128]	[256, 128, 64]	[256, 128, 64]
CRITIC NETWORK	[400,200,100]	[400,200,100]	[256,256,128,128]	[256, 128, 64]	[256, 128, 64]
ACTIVATION FUNCTION	ELU	ELU	ELU	ELU	ELU

b) Humanoid Dribbling: In this task, the robot learns to dribble along routes with random turns. The observation space is augmented with features of the ball compared with the tightrope task. For observation calculation, the global coordinate system changes with the new target route at the turning position. At each non-terminating step, the reward $r = w_v \times r_v + w_{\text{alive}} \times r_{\text{alive}} + w_{\text{dist}} \times r_{\text{dist}} + w_{\text{heading}} \times r_{\text{heading}} + w_{\text{action}} \times r_{\text{action}} + w_{\text{energy}} \times r_{\text{energy}} + w_{\text{lateral}} \times r_{\text{lateral}}$, where

- r_v is the ball’s forward velocity, $w_v = 2.0$;
- $r_{\text{alive}} = 1$, $w_{\text{alive}} = 0.2$;
- $r_{\text{dist}} = e^{-d}$ where d is the 2d distance from torso to the ball, $w_{\text{dist}} = 0.2$;
- $r_{\text{heading}} = e_{\text{forward},x}$, where e_{forward} is the basis vector of torso’s x axis in the global system, $w_{\text{forward}} = 1.0$;
- $r_{\text{action}}, r_{\text{energy}}$ are the same with Humanoid Tightrope
- $r_{\text{lateral}} = v_{\text{ball},y}$ is the penalty for the ball’s lateral velocity, $w_{\text{lateral}} = -0.5$

The reward is -1 for termination step. The action is the force applied to all joints.

c) A1 Beam: In this task, the agent controls the quadruped robot Unitree A1 [24] to walk on a balance beam with width of 10 cm following a predefined speed. Considering the width of A1 and the balance beam, it is much harder than walking on the ground. There are overall 12 motors for Unitree A1, 3 for each leg. At each non-terminating step, the reward $r = w_v \times r_v + w_{\text{alive}} \times r_{\text{alive}} + w_{\text{heading}} \times r_{\text{heading}} + w_{\text{action}} \times r_{\text{action}} + w_{\text{lateral}} \times r_{\text{lateral}}$, where

- $r_v = e^{-|v_{\text{torso},x} - v_{\text{target}}|}$ is speed tracking reward, $w_v = 1.0$;
- $r_{\text{alive}} = 1$, $w_{\text{alive}} = 1.0$;
- $r_{\text{heading}} = e_{\text{forward},x}$, where e_{forward} is the basis vector of torso’s x axis in global coordinate system, $w_{\text{forward}} = 1.0$;
- $r_{\text{action}} = \|a\|_2^2$, where a is the joints action, $w_{\text{action}} =$

-0.5

- $r_{\text{lateral}} = v_{\text{torso},y}$ is penalty for lateral velocity, $w_{\text{lateral}} = -1.0$

The reward is -1 for termination step. The robot has a low-level joint controller. The action is the target angular position of all joints.

d) Trifinger Lift: Trifinger [25] is a 3-finger manipulator for learning dexterity. The goal of the task is to move a cube from a random initial pose to an arbitrary 6-DoF target position and orientation. The environment is the same as that of [4], except that we remove the auxiliary penalty for finger movement, which increases the difficulty of the task. The robot has a low-level joint controller. The action is the target angular position of all joints.

e) Ant Acrobatic: In this task, an ant learns to do complex acrobatics (e.g. heading a pole) on a ball, which extremely challenges the ability of agents to maintain balance. The action space is 8 dimensions. At each non-terminating step, the reward $r = w_{\text{alive}} \times r_{\text{alive}} + w_{\text{action}} \times r_{\text{action}} + w_{\text{energy}} \times r_{\text{energy}}$, where

- $r_{\text{alive}} = 1$, $w_{\text{alive}} = 0.5$;
- $r_{\text{action}} = \|a\|_2^2$, where a is joints action, $w_{\text{action}} = -0.005$
- r_{energy} is joints power consumption, $w_{\text{energy}} = -0.05$

The reward is -1 for termination step. The action is the force applied to all joints.

We conclude the observation space for each task in Table III for easier reading.

TABLE III: Tasks Information

	HUMANOID TIGHTROPE	HUMANOID FOOTBALL	TRIFINGER MOVE	A1 BEAM	ANT AEROBATIC	
OBSERVATION DIMENSION	74	80	41	47	57	
O_C	TORSO	y_{TORSO} z_{TORSO} $v_{\text{TORSO},x}$ $v_{\text{TORSO},y}$ $v_{\text{TORSO},z}$ $\omega_{\text{TORSO},x}$ $\omega_{\text{TORSO},y}$ $\omega_{\text{TORSO},z}$ α_{TORSO} β_{TORSO} γ_{TORSO}	y_{TORSO} z_{TORSO} $v_{\text{TORSO},x}$ $v_{\text{TORSO},y}$ $v_{\text{TORSO},z}$ $\omega_{\text{TORSO},x}$ $\omega_{\text{TORSO},y}$ $\omega_{\text{TORSO},z}$ α_{TORSO} β_{TORSO} γ_{TORSO}	y_{TORSO} z_{TORSO} $v_{\text{TORSO},x}$ $v_{\text{TORSO},y}$ $v_{\text{TORSO},z}$ $\omega_{\text{TORSO},x}$ $\omega_{\text{TORSO},y}$ $\omega_{\text{TORSO},z}$ α_{TORSO} β_{TORSO} γ_{TORSO}	x_{TORSO} y_{TORSO} z_{TORSO} $v_{\text{TORSO},x}$ $v_{\text{TORSO},y}$ $v_{\text{TORSO},z}$ $\omega_{\text{TORSO},x}$ $\omega_{\text{TORSO},y}$ $\omega_{\text{TORSO},z}$ α_{TORSO} β_{TORSO} γ_{TORSO}	
	TORSO JOINTS	$\theta_{\text{LOWER WAIST},x}$ $\theta_{\text{LOWER WAIST},y}$ $\theta_{\text{PELVIS},x}$ $\omega_{\text{LOWER WAIST},x}$ $\omega_{\text{LOWER WAIST},y}$ $\omega_{\text{PELVIS},x}$ $a_{\text{LOWER WAIST},x}$ $a_{\text{LOWER WAIST},y}$ $a_{\text{PELVIS},x}$	$\theta_{\text{LOWER WAIST},x}$ $\theta_{\text{LOWER WAIST},y}$ $\theta_{\text{PELVIS},x}$ $\omega_{\text{LOWER WAIST},x}$ $\omega_{\text{LOWER WAIST},y}$ $\omega_{\text{PELVIS},x}$ $a_{\text{LOWER WAIST},x}$ $a_{\text{LOWER WAIST},y}$ $a_{\text{PELVIS},x}$			
	EXTERNAL OBJECTS		x_{BALL} y_{BALL} z_{BALL} $v_{\text{BALL},x}$ $v_{\text{BALL},y}$ $v_{\text{BALL},z}$	x_{CUBE} y_{CUBE} z_{CUBE} $H_{\text{CUBE},x}$ $H_{\text{CUBE},y}$ $H_{\text{CUBE},z}$ $H_{\text{CUBE},w}$ $x_{\text{CUBE TARGET}}$ $y_{\text{CUBE TARGET}}$ $z_{\text{CUBE TARGET}}$ $H_{\text{CUBE TARGET},x}$ $H_{\text{CUBE TARGET},y}$ $H_{\text{CUBE TARGET},z}$ $H_{\text{CUBE TARGET},w}$	x_{POLE} y_{POLE} z_{POLE} $v_{\text{POLE},x}$ $v_{\text{POLE},y}$ $v_{\text{POLE},z}$ $\omega_{\text{POLE},x}$ $\omega_{\text{POLE},y}$ $\omega_{\text{POLE},z}$ $UP_{\text{POLE},x}$ $UP_{\text{POLE},y}$ $UP_{\text{POLE},z}$ x_{BALL} y_{BALL} z_{BALL} $v_{\text{BALL},x}$ $v_{\text{BALL},y}$ $v_{\text{BALL},z}$ $\omega_{\text{BALL},x}$ $\omega_{\text{BALL},y}$ $\omega_{\text{BALL},z}$	
$O_{S,i}$	LIMB JOINTS	$\theta_{\text{UPPER ARM},x}$ $\theta_{\text{UPPER ARM},z}$ $\theta_{\text{LOWER ARM},x}$ $\theta_{\text{THIGH},x}$ $\theta_{\text{THIGH},y}$ $\theta_{\text{THIGH},z}$ $\theta_{\text{KNEE},x}$ $\theta_{\text{FOOT},x}$ $\theta_{\text{FOOT},y}$ $\omega_{\text{UPPER ARM},x}$ $\omega_{\text{UPPER ARM},z}$ $\omega_{\text{LOWER ARM},x}$ $\omega_{\text{THIGH},x}$ $\omega_{\text{THIGH},y}$ $\omega_{\text{THIGH},z}$ $\omega_{\text{KNEE},x}$ $\omega_{\text{FOOT},x}$ $\omega_{\text{FOOT},y}$ $a_{\text{UPPER ARM},x}$ $a_{\text{UPPER ARM},z}$ $a_{\text{LOWER ARM},x}$ $a_{\text{THIGH},x}$ $a_{\text{THIGH},y}$ $a_{\text{THIGH},z}$ $a_{\text{KNEE},x}$ $a_{\text{FOOT},x}$ $a_{\text{FOOT},y}$	$\theta_{\text{UPPER ARM},x}$ $\theta_{\text{UPPER ARM},z}$ $\theta_{\text{LOWER ARM},x}$ $\theta_{\text{THIGH},x}$ $\theta_{\text{THIGH},y}$ $\theta_{\text{THIGH},z}$ $\theta_{\text{KNEE},x}$ $\theta_{\text{FOOT},x}$ $\theta_{\text{FOOT},y}$ $\omega_{\text{UPPER ARM},x}$ $\omega_{\text{UPPER ARM},z}$ $\omega_{\text{LOWER ARM},x}$ $\omega_{\text{THIGH},x}$ $\omega_{\text{THIGH},y}$ $\omega_{\text{THIGH},z}$ $\omega_{\text{KNEE},x}$ $\omega_{\text{FOOT},x}$ $\omega_{\text{FOOT},y}$ $a_{\text{UPPER ARM},x}$ $a_{\text{UPPER ARM},z}$ $a_{\text{LOWER ARM},x}$ $a_{\text{THIGH},x}$ $a_{\text{THIGH},y}$ $a_{\text{THIGH},z}$ $a_{\text{KNEE},x}$ $a_{\text{FOOT},x}$ $a_{\text{FOOT},y}$	$\theta_{\text{FINGER UPPER}}$ $\theta_{\text{FINGER MIDDLE}}$ $\theta_{\text{FINGER LOWER}}$ $\omega_{\text{FINGER UPPER}}$ $\omega_{\text{FINGER MIDDLE}}$ $\omega_{\text{FINGER LOWER}}$ $a_{\text{FINGER UPPER}}$ $a_{\text{FINGER MIDDLE}}$ $a_{\text{FINGER LOWER}}$	$\theta_{\text{FRONT HIP}}$ $\theta_{\text{FRONT THIGH}}$ $\theta_{\text{FRONT CALF}}$ $\theta_{\text{REAR HIP}}$ $\theta_{\text{REAR THIGH}}$ $\theta_{\text{REAR CALF}}$ $\omega_{\text{FRONT HIP}}$ $\omega_{\text{FRONT THIGH}}$ $\omega_{\text{FRONT CALF}}$ $\omega_{\text{REAR HIP}}$ $\omega_{\text{REAR THIGH}}$ $\omega_{\text{REAR CALF}}$ $a_{\text{FRONT HIP}}$ $a_{\text{FRONT THIGH}}$ $a_{\text{FRONT CALF}}$ $a_{\text{REAR HIP}}$ $a_{\text{REAR THIGH}}$ $a_{\text{REAR CALF}}$	
	$ \mathcal{N} $	2	2	3	2	4
ACTION DIMENSION	21	21	9	12	8	