Morphologically Symmetric Reinforcement Learning for Ambidextrous Bimanual Manipulation

Zechu Li [1] **Yufeng Jin** [1,2] **Daniel Ordoñez Apraez** [3] **Claudio Semini** [3]
Puze Liu [4*] **Georgia Chalvatzaki** [1,5,6]
TU Darmstadt [1] Honda Research Institute Europe [2] Istituto Italiano di Tecnologia [3]
DFKI [4] Hessian.AI [5] Robotics Institute Germany [6]

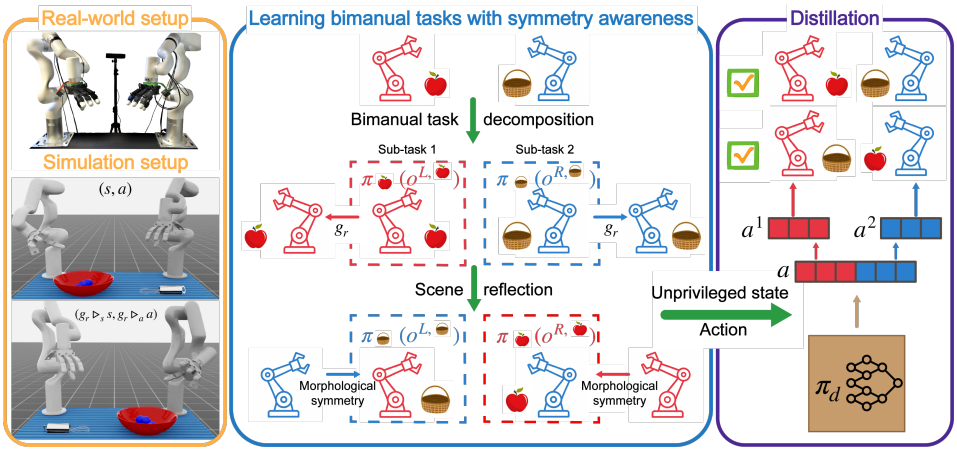


Figure 1: Overview of SYMDEX: (Left) Digital twin of our bimanual robot. (Middle) The task is decomposed into two sub-tasks, each trained with a dedicated equivariant policy that transfers across symmetric configurations. (Right) Task-specific policies are distilled into an equivariant policy.

Abstract: Humans naturally exhibit bilateral symmetry in their gross manipulation skills, effortlessly mirroring simple actions between left and right hands. Bimanual robots—which also feature bilateral symmetry—should similarly exploit this property to perform tasks with either hand. Unlike humans, who often favor a dominant hand for fine dexterous skills, robots should ideally execute ambidextrous manipulation with equal proficiency. To this end, we introduce SYMDEX (SYMmetric DEXterity), a reinforcement learning framework for ambidextrous bi-manipulation that leverages the robot's inherent bilateral symmetry as an inductive bias. SYMDEX decomposes complex bimanual manipulation tasks into per-hand subtasks and trains dedicated policies for each. By exploiting bilateral symmetry via equivariant neural networks, experience from one arm is inherently leveraged by the opposite arm. We then distill the subtask policies into a global ambidextrous policy that is independent of the hand-task assignment. We evaluate SYMDEX on six challenging simulated manipulation tasks and demonstrate successful real-world deployment on two of them. Our approach strongly outperforms baselines on complex task in which the left and right hands perform different roles. We further demonstrate SYMDEX's scalability by extending it to a four-arm manipulation setup, where our symmetry-aware policies enable effective multi-arm collaboration and coordination. Our website is made publicly available at: https://supersglzc.github.io/projects/symdex/.

Keywords: Bimanual Dexterous Manipulation, Reinforcement Learning

*Corresponding author

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

1 Introduction

Humans inherently exhibit bilateral symmetry in their gross motor skills, which allows them to effortlessly mirror simple actions between their left and right limbs. However, when it comes to fine dexterous tasks (e.g., writing, playing instruments), most people develop a dominant side, a phenomenon known as *handedness*. This functional control asymmetry often leads to suboptimal task strategies, such as switching hands to maintain control robustness. In contrast, bimanual robots—which frequently also feature bilateral symmetry—are not inherently bound by handedness. Hence, in the context of manipulation, there is a unique opportunity to design algorithms that perform tasks ambidextrously, which enables the interchangeable use of limbs across a wide range of task configurations and plans actions based on efficiency rather than a left/right preference.

Achieving ambidextrous bimanual manipulation requires control policies incorporating awareness of the robot's bilateral symmetry. Recent robotics research has explored such structural symmetries—referred to as *morphological symmetries* [1]—to develop symmetry-aware learning methods. Most previous work has focused on legged locomotion, where exploiting morphological symmetry improves control robustness and sample efficiency [2, 3, 4, 5, 6, 7, 8]. In contrast, its use in manipulation—especially for bimanual or multi-robot systems—remains uncharted [9]. Whether embedding symmetry in manipulation policies can offer similar gains in generalization and sample efficiency for high-dimensional, contact-rich tasks is still an open question.

Reinforcement Learning (RL) is a compelling paradigm for bimanual dexterous manipulation, especially in sim-to-real settings [10, 11, 12]. Unlike imitation learning, which needs large, high-quality demonstrations, RL trains in randomized environments, acquiring robust behaviors via massive parallel simulation. Yet the complexity of bimanual or multi-robot systems has confined prior work to narrowly scoped tasks enforced by system constraints (e.g., hand-only control [10] or arm joint locking [11]). Hence we ask: *Can RL scale to fully actuated bimanual—and multi-robot—systems by embedding morphological symmetry as a structural prior in policy learning?*

Learning bimanual (and multi-arm) manipulation via RL presents significant challenges. First, the **high-dimensional observation and action spaces** make policy learning difficult: a bimanual robot with dexterous hands must jointly process and control numerous joints, leading to an exponentially large exploration space [10, 11, 12, 13]. Second, the dual-arm setup and task complexity exacerbate the **credit assignment problem** [14, 15]. During exploration, one arm may succeed while others fail, and the reward signal merges both positive and negative feedback, making reward shaping intractable in the presence of numerous task-specific rewards. Additionally, when ambidexterity is involved, each arm may perform different tasks, turning the problem into **multi-task learning**. Although individual reward terms can be well-defined, the scale of each term needs careful tuning to balance learning across tasks. Finally, **zero-shot sim-to-real transfer** introduces challenges such as the sim-to-real gap and safety concerns, requiring the learned policy to at least prevent inter-arm collisions during deployment [16, 17, 18]. We provide a detailed review of related work in Appx. C.

To address these open challenges, we introduce **SYMDEX** (SYMmetric DEXterity), a RL framework for ambidextrous bimanual (and multi-arm) dexterous manipulation, that explicitly incorporates morphological symmetry as an inductive bias, both architecturally and algorithmically. SYMDEX decomposes complex bimanual tasks into per-hand subtasks and trains a separate policy for each using an equivariant neural network [19]. This structure inherently shares experience across symmetric limbs, exploiting morphological symmetry to accelerate learning. SYMDEX operates entirely in joint space, without relying on task-space solvers or handcrafted action symmetries. To enable flexibility and remove the need for fixed hand-task assignment, we distill these sub-policies into a unified global equivariant policy via teacher-student distillation. The resulting policy is ambidextrous by design and zero-shot deployable in the real world.

Our contributions are as follows: **1. Morphological symmetry-aware policy learning approach.** We present SYMDEX, a RL framework that explicitly leverages the inherent morphological symmetry in bimanual robotic systems to enable ambidextrous control and generalization across structurally symmetric task configurations. **2. Scalable framework generalizable to multi-arm tasks.**

We demonstrate that our approach can be deployed to more complex multi-arm settings with complexer symmetry groups without increasing the task complexity. **3. A full learning recipe enables zero-shot sim-to-real transfer**. Our framework addresses key challenges to ensure robust real-world deployment, we incorporate a curriculum learning strategy for zero-shot sim-to-real transfer. We evaluate SYMDEX on six diverse and challenging bimanual manipulation tasks in simulation and successfully deploy it on two of them in the real world.

2 Background

Here, we review the foundational concepts and notation necessary for formalizing how symmetries serve as an inductive bias in learning bimanual (and multi-robot) dexterous manipulation policies via RL. Extended definitions are provided in Appx. A.

A **symmetry group** (see Def. A.1) is a set of invertible transformations, denoted as $\mathbb{G} = \{e, g_a, g_b, \dots\}$, that can be defined to act on distinct objects, such as the state $\mathcal{S}$ and action $\mathcal{A}$ spaces of a Markov decision process (MDP). To do so we define the **group actions** (see Def. A.2). Specifically, let $(\triangleright_{\mathcal{S}}) \colon \mathbb{G} \times \mathcal{S} \to \mathcal{S}$ and $(\triangleright_{\mathcal{A}}) \colon \mathbb{G} \times \mathcal{A} \to \mathcal{A}$ denote the action of $\mathbb{G}$ on $\mathcal{S}$ and $\mathcal{A}$, respectively. Then, given a symmetry transformation $g \in \mathbb{G}$ and a state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, the g-transformed pair is denoted by $(g \triangleright_{\mathcal{S}} s, g \triangleright_{\mathcal{A}} a) \in \mathcal{S} \times \mathcal{A}$ (see Fig. 1-left).

The **symmetries of MDPs** are defined as state–action transformations that preserve the MDP's dynamics. This property is characterized by the $\mathbb{G}$-equivariance (see Def. A.5) of the dynamics:

$$g \triangleright_{\mathcal{S}} \mathbb{E}[f(s,a)] = \mathbb{E}[f(g \triangleright_{\mathcal{S}} s,\ g \triangleright_{\mathcal{A}} a)], \qquad \forall (s,a) \in \mathcal{S} \times \mathcal{A},\ \forall g \in \mathbb{G}, \tag{1}$$

where $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a function mapping state–action pairs to subsequent states. For example, consider the bimanual environment in Fig. 1, where the symmetry group is the reflection group $\mathbb{G} = \mathbb{C}_2 = \{e, g_r \mid g_r^2 = e\}$—with g_r denoting the robots' bilateral symmetry. Here, Eq. (1) shows that the in-hand manipulation dynamics for the left and right hands are equivalent, up to g_r.

The symmetry priors from Eq. (1) constrain the MDP's optimal policy and value function. To see this, let's formally denote a Partially Observable MDP (POMDP) by the tuple $\langle \mathcal{S}, \mathcal{A}, r, \tau, \rho_0, \gamma, \mathcal{O}, \boldsymbol{\sigma} \rangle$[2], where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{O}$ are the state, action, and observation spaces; $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function; $\tau : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is the transition kernel; $\rho_0 : \mathcal{S} \to \mathbb{R}_+$ is the initial state distribution; γ is the discount factor; and $\boldsymbol{\sigma} : \mathcal{S} \to \mathcal{O}$ is the observation function. A POMDP is said to be **symmetric** if the following conditions hold:

Definition 2.1 (Symmetric POMDP)**.** *A POMDP $\langle \mathcal{S}, \mathcal{A}, r, \tau, \rho_0, \gamma, \mathcal{O}, \boldsymbol{\sigma} \rangle$ possess the symmetry group $\mathbb{G}$ when the state and action spaces $\mathcal{S}$ and $\mathcal{A}$ admit group actions $(\triangleright_{\mathcal{S}})$ and $(\triangleright_{\mathcal{A}})$, and (r, τ, ρ_0) are all $\mathbb{G}$-invariant. That is, if for every $g \in \mathbb{G}$, $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{S}$, and $\boldsymbol{a} \in \mathcal{A}$, we have:*

$$\tau(g \triangleright_{\mathcal{S}} \boldsymbol{s}' \mid g \triangleright_{\mathcal{S}} \boldsymbol{s}, g \triangleright_{\mathcal{A}} \boldsymbol{a}) = \tau(\boldsymbol{s}' \mid \boldsymbol{s}, \boldsymbol{a}), \qquad \rho_0(g \triangleright_{\mathcal{S}} \boldsymbol{s}) = \rho_0(\boldsymbol{s}), \qquad r(g \triangleright_{\mathcal{S}} \boldsymbol{s}, g \triangleright_{\mathcal{A}} \boldsymbol{a}) = r(\boldsymbol{s}, \boldsymbol{a}). \tag{2}$$

*POMDP's satisfying Eq. (2) are constrained to have **optimal** policy and value functions satisfying:*

$$\underbrace{g \triangleright_{\mathcal{A}} \pi^*(\boldsymbol{\sigma}(s)) = \pi^*(\boldsymbol{\sigma}(g \triangleright_{\mathcal{S}} s))}_{\text{Policy } \mathbb{G}\text{-equivariance}}, \qquad \underbrace{V^*(\boldsymbol{\sigma}(s)) = V^*(\boldsymbol{\sigma}(g \triangleright_{\mathcal{S}} s))}_{\text{Value function } \mathbb{G}\text{-invariance}}, \quad \forall\ s \in \mathcal{S},\ g \in \mathbb{G}. \text{ (refer to [20])} \tag{3}$$

Note 2.1. *A policy π of a $\mathbb{G}$-symmetric POMDP can satisfy the $\mathbb{G}$-equivariance constraints of Eq. (3) only if the observation function $\boldsymbol{\sigma}$ is $\mathbb{G}$-equivariant (see Prop. B.1). Therefore, to leverage Eq. (3) in policy learning, $\mathcal{O}$ must carry a group action $(\triangleright_{\mathcal{O}})$ and $\boldsymbol{\sigma}$ must be $\mathbb{G}$-equivariant.*

Bimanual and multi-robot manipulation In bimanual (and multi-robot) dexterous manipulation, each task (e.g., stir eggs; see Fig. 1) can be decomposed into a sequence of concurrent and sequential subtasks, with each agent assigned subtasks (e.g., left arm grasps the egg beater while right arm holds the bowl). Hence, these environments are modeled as a Multi-Task Multi-Agent POMDP (MTMA-POMDP) defined by the tuple $\langle \mathcal{S}, \mathcal{A}, R, \tau, \rho_0, \gamma, \mathcal{O}, \boldsymbol{\sigma}, \mathbb{K}, \mathbb{N} \rangle$, where $\mathbb{N}$ denotes the agent set—with $n \in \mathbb{N}$ representing a unique robot arm (with a dexterous hand)—and $\mathbb{K}$ denotes the task set—with $k \in \mathbb{K}$ a manipulation subtask. This structure enables decomposition of the overall action

[2]In manipulation tasks, observations of number of contacts and contact points are typically unavailable, hence the control problem of manipulation is modeled as a POMDP.

space as $\mathcal{A} = \oplus_{n\in\mathbb{N}}\mathcal{A}_n$, and defines subtask policies $\boldsymbol{a}^n \sim \pi_k(\boldsymbol{o}^{n,k}) \in \mathcal{A}_n$ for all $k \in \mathbb{K}$, where $\boldsymbol{o}^{n,k} = \boldsymbol{\sigma}^n(\boldsymbol{s}, k)$ denotes the subtask-and-agent specific observation. Each task defines a reward r_k, which define the corresponding value function $V^k(\boldsymbol{o}_t^{n,k}) = \mathbb{E}_{\pi_k}\left[\sum_t^\infty \gamma^t r_k(\boldsymbol{o}_t^{n,k})\right]$. Consequently, the MTMA-POMDP reward and value functions are defined as: $r(\boldsymbol{s}_t) = \sum_{(n,k)\in\mathbb{I}} r_k(\boldsymbol{\sigma}^n(\boldsymbol{s}_t, k))$ and $V(\boldsymbol{s}_t) = \sum_{(n,k)\in\mathbb{I}} V_k(\boldsymbol{\sigma}^n(\boldsymbol{s}_t, k))$. Where $\mathbb{I}$ denotes the set of agent-subtask pairwise pairings.

3 Method

In this paper, we aim to achieve ambidextrous bimanual manipulation—enabling a robot to perform tasks equally well with either arm and choose actions based on efficiency. This requires each arm to flexibly handle different subtasks depending on scene configuration. For example, in a bimanual scenario (see Fig. 1-left-middle), the robot uses its right arm to operate an egg beater while its left holds a bowl; in a reflected workspace (see Fig. 1-left-bottom), the bowl is closer to the right arm and the egg beater to the left. Consequently, the optimal behavior is to switch roles—using the right arm to hold the bowl and the left to operate the egg beater.

Learning such ambidextrous policy is challenging due to the high-dimensional observation-action spaces and the difficulty of credit assignment between arms. To address this, we formulate bimanual manipulation as a MTMA-POMDP (Sec. 2), where each agent corresponds to a single robot arm executing one subtask. This reduces the dimensionality of each agent's observation-action spaces and assigns subtask-specific reward, simplifying credit assignment. However, each agent must still learn to perform all subtasks to achieve ambidexterity. Notably, there is symmetry between the subtasks assigned to each agent (Fig. 1), which motivates leveraging morphological symmetries as a strong inductive bias and learning an equivariant policy for each subtask.

An illustrative example To express this ambidexterity using the formalism of Sec. 2, note that changes in agents' subtask assignments are formalized through group action on set of agent-task pairs $\mathbb{I}$ (see Def. A.2), i.e., $(\triangleright_{\mathbb{I}})\colon \mathbb{G} \times (\mathbb{N} \times \mathbb{K}) \to (\mathbb{N} \times \mathbb{K})$. Thus, in the bimanual manipulation environment of Fig. 1, with $\mathbb{N} = \{\mathrm{R}, \mathrm{L}\}$ and $\mathbb{K} = \{\mathrm{B}, \mathrm{E}\}$—where R and L denote the left/right arms, and B and E denote the bowl-holding and egg-beater-operating subtasks—a bilateral reflection of the workspace, g_r, leads to the following permutation of agents and tasks:

$$g_r \triangleright_{\mathbb{I}} \begin{bmatrix} (\mathrm{L}, \mathrm{B}) \\ (\mathrm{R}, \mathrm{E}) \end{bmatrix} = \begin{bmatrix} (\mathrm{L}, g_r \triangleright_{\mathbb{K}} \mathrm{B}) \\ (\mathrm{R}, g_r \triangleright_{\mathbb{K}} \mathrm{E}) \end{bmatrix} = \begin{bmatrix} (\mathrm{L}, \mathrm{E}) \\ (\mathrm{R}, \mathrm{B}) \end{bmatrix}, \tag{4}$$

where $\triangleright_{\mathbb{K}}$ denotes the group action on task permutation. Note that since we learn a dedicated policy per subtask, these changes lead to the following group action on the action space of the POMDP:

$$g_r \triangleright_{\mathcal{A}} \boldsymbol{a} := g_r \triangleright_{\mathcal{A}} \begin{bmatrix} \boldsymbol{a}^{\mathrm{L}} \sim \pi_{\mathrm{B}}(\boldsymbol{o}^{\mathrm{L},\mathrm{B}}) \\ \boldsymbol{a}^{\mathrm{R}} \sim \pi_{\mathrm{E}}(\boldsymbol{o}^{\mathrm{R},\mathrm{E}}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}^{\mathrm{L}} \sim g_r \triangleright_{\mathcal{A}_{\mathbb{N}}} (\pi_{\mathrm{E}}(\boldsymbol{o}^{\mathrm{R},\mathrm{E}})) \\ \boldsymbol{a}^{\mathrm{R}} \sim g_r \triangleright_{\mathcal{A}_{\mathbb{N}}} (\pi_{\mathrm{B}}(\boldsymbol{o}^{\mathrm{L},\mathrm{B}})) \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}^{\mathrm{L}} \sim \pi_{\mathrm{E}}(\boldsymbol{\sigma}^{\mathrm{L}}(g_r \triangleright_{\mathcal{S}} \boldsymbol{s}, \mathrm{E})) \\ \boldsymbol{a}^{\mathrm{R}} \sim \pi_{\mathrm{B}}(\boldsymbol{\sigma}^{\mathrm{R}}(g_r \triangleright_{\mathcal{S}} \boldsymbol{s}, \mathrm{B})) \end{bmatrix} \tag{5}$$

Essentially, this shows that the action of a robot arm in the reflected environment equals the symmetry-transformed action of the opposite arm in the original environment (see Fig. 1-left). Here, $(\triangleright_{\mathcal{A}_{\mathbb{N}}})$ denotes the group action on an individual arm's action space. Crucially, the right-hand side of Eq. (5) relies on the $\mathbb{G}$-equivariance of each subtask policy and observation function, ensuring that the global policy is equivariant. Moreover, this analysis directly extends to multi-robot systems with more complex symmetry groups:

Morphological symmetries in MTMA-POMDP Let $(\mathcal{S}, \mathcal{A}, \boldsymbol{r}, \tau, \rho_0, \gamma, \mathcal{O}, \boldsymbol{\sigma}, \mathbb{K}, \mathbb{N})$ denote a N-robot manipulation MTMA-POMDP, with agents $\mathbb{N} = \{1, \dots, N\}$, tasks $\mathbb{K} = \{k_1, \dots, k_N\}$, and agent-task pairs $\mathbb{I} = \{(1, k_1), \dots\}$ associated with a $\mathbb{G}$-symmetric POMDP (as per Def. 2.1). Then, the group action on the POMDP action space $\mathcal{A}$ is defined via the tensor product (see note A.1) of the group actions $(\triangleright_{\mathcal{A}})$ expressed by:

$$g \triangleright_{\mathcal{A}} \boldsymbol{a} = g \triangleright_{\mathcal{A}} \begin{bmatrix} \boldsymbol{a}^1 \sim \pi_{k_1}(\boldsymbol{o}^{1,k_1}) \\ \vdots \\ \boldsymbol{a}^N \sim \pi_{k_N}(\boldsymbol{o}^{N,k_N}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}^1 \sim \pi_{g \triangleright_{\mathbb{K}} k_1}(\boldsymbol{\sigma}^1(g \triangleright_{\mathcal{S}} \boldsymbol{s}, g \triangleright_{\mathbb{K}} k_1)) \\ \vdots \\ \boldsymbol{a}^N \sim \pi_{g \triangleright_{\mathbb{K}} k_N}(\boldsymbol{\sigma}^N(g \triangleright_{\mathcal{S}} \boldsymbol{s}, g \triangleright_{\mathbb{K}} k_N)) \end{bmatrix}, \tag{6}$$

Eq. (6) generalizes the bimanual manipulation example in Eq. (5) to an N-robot task with $\mathbb{G}$-equivariant dynamics (see Eq. (1)). Crucially, this analysis identifies the symmetry constraints for

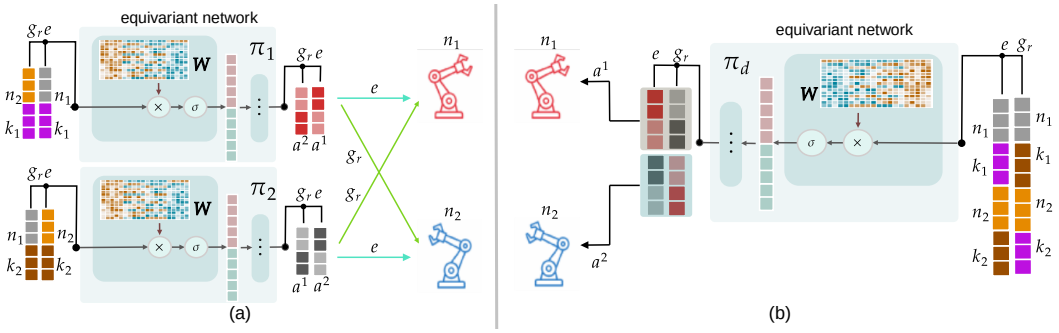


Figure 2: A comparison of action execution between (a) subtask policies and (b) global policy, where n_1, n_2 are agents, k_1, k_2 are two subtasks, and e, g_r are transformations in the reflection group.

each subtask policy and observation function of the MTMA-POMDP while characterizing the group actions on the global action space of the POMDP. This enables us to first learn $\mathbb{G}$-equivariant policies for each *subtask* and then distill them into a global $\mathbb{G}$-equivariant policy for the entire system.

Symmetry-aware learning of subtask policies After decomposing a multi-robot manipulation tasks into subtasks we learn a policy for each. Since each subtask has a unique observation space—comprising the assigned robot arm's state and the task-specific objects state—each subtask policy is parameterized as a $\mathbb{G}$-equivariant Neural Network (NN) [19] (see Fig. 2(a)), satisfying:

$$g \triangleright_{\mathcal{A}_{\mathbb{N}}} \pi_k^{\boldsymbol{\theta}_k}(\boldsymbol{o}^{n,k}) = \pi_k^{\boldsymbol{\theta}_k}(g \triangleright_{\mathcal{O}_k} \boldsymbol{\sigma}^n(\boldsymbol{s}, k)) = \pi_k^{\boldsymbol{\theta}_k}(\boldsymbol{\sigma}^{(g \triangleright_{\mathbb{I}} \mathbb{I})[k]}(g \triangleright_{\mathcal{S}} \boldsymbol{s}, k)), \quad \forall\, (n,k) \in \mathbb{I}, g \in \mathbb{G}. \tag{7}$$

Here $\boldsymbol{\theta}_k$ are the parameters of the k-th subtask network, the group action on the observation space $\triangleright_{\mathcal{O}_k}$ is analogous to $\triangleright_{\mathcal{A}_{\mathbb{N}}}$ which transforms the robot-arm state and task-specific measurements to its symmetric counterpart, and $\mathbb{I}[\cdot]$ operator denotes the agent query from the agent-task pair, whose task equals $(\cdot)$. See [1] for details on how to construct these actions.

Under the assumption that each subtask reward is $\mathbb{G}$-invariant, i.e., $r_k(\boldsymbol{\sigma}^n(\boldsymbol{s}, k)) = r_k(\boldsymbol{\sigma}^{(g \triangleright_{\mathbb{I}} \mathbb{I})[k]}(g \triangleright_{\mathcal{S}} \boldsymbol{s}, k))$ for all $(n,k) \in \mathbb{I},\ g \in \mathbb{G}$ —a premise that holds naturally in dexterous manipulation tasks with morphological symmetries because most reward terms depend on hand–object pose errors— the corresponding subtask value function can be parameterized by a $\mathbb{G}$-invariant NN satisfying:

$$V_k^{\boldsymbol{\theta}_k}(\boldsymbol{o}^{n,k}) = V_k^{\boldsymbol{\theta}_k}\big(g \triangleright_{\mathcal{O}_k} \boldsymbol{\sigma}^n(\boldsymbol{s}, k)\big) = V_k^{\boldsymbol{\theta}_k}\big(\boldsymbol{\sigma}^{(g \triangleright_{\mathbb{I}} \mathbb{I})[k]}(g \triangleright_{\mathcal{S}} \boldsymbol{s}, k)\big), \quad \forall\, (n,k) \in \mathbb{I},\ g \in \mathbb{G}. \tag{8}$$

This parameterization allows us to employ the Proximal Policy Optimization (PPO) algorithm [21] to learn the K subtask $\mathbb{G}$-equivariant policies and $\mathbb{G}$-invariant value functions in parallel [20].

Global $\mathbb{G}$-equivariant policy distillation We follow a teacher-student paradigm [16] that distills subtask policies into a global $\mathbb{G}$-equivariant policy—which yields an ambidextrous policy in the case of bimanual manipulation. Specifically, the learned N subtask policies serve as expert policies to generate a dataset of state–action pairs $\mathbb{D} = \{(\boldsymbol{s}_i, \boldsymbol{a}_i)\}_{i=1}^M$ (see Eq. (6)), which we use to learn a global policy $\pi_d^{\boldsymbol{\phi}}$ satisfying:

$$g \triangleright_{\mathcal{A}} \pi_d^{\boldsymbol{\phi}}(\boldsymbol{\sigma}(\boldsymbol{s})) = \pi_d^{\boldsymbol{\phi}}(g \triangleright_{\mathcal{O}} \boldsymbol{\sigma}(\boldsymbol{s})) = \pi_d^{\boldsymbol{\phi}}(\boldsymbol{\sigma}(g \triangleright_{\mathcal{S}} \boldsymbol{s})), \quad \forall\, g \in \mathbb{G}, (\boldsymbol{s}, \boldsymbol{a}) \in \mathcal{S} \times \mathcal{A}. \tag{9}$$

Here $\boldsymbol{\phi}$ are the network parameters; $\triangleright_{\mathcal{A}}$ and $\triangleright_{\mathcal{O}}$ are the group actions on the global action and observation spaces (see Eq. (6) and Appx. E). Notably, the distilled policy infers task–arm assignments directly from demonstrations (Fig. 2(b)), while $\mathbb{G}$-equivariance guarantees identical performance from any symmetric initial state—i.e. $\boldsymbol{s}_0 = \bar{\boldsymbol{s}}$ and $\boldsymbol{s}_0 = g \triangleright_{\mathcal{S}} \bar{\boldsymbol{s}}$ yield the same outcome for all $g \in \mathbb{G}$. This constraint boosts robustness and promotes generalization to unseen configurations [1, 22, 23]. In addition, we ensure that the global policy is trained exclusively on non-privileged observations, enabling zero-shot deployment in the real world (Fig. 1-right). The equivariant/invariant MLPs are implemented using the ESCNN library [24].

Curriculum Learning for Sim-to-Real Transfer To smooth sim-to-real transfer, we employ a simple curriculum that progresses whenever the agent surpasses a success-rate threshold. It comprises two components. **Randomization**: training begins with scene-level symmetry randomization;

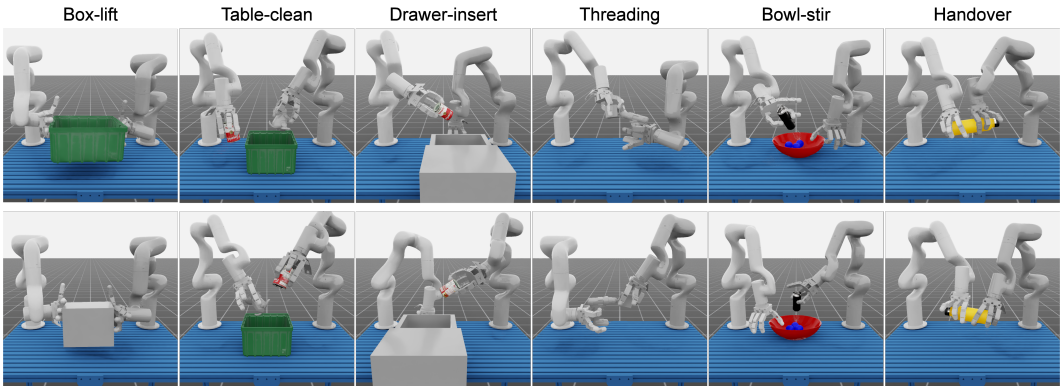


Figure 3: Our benchmark of six bimanual dexterous manipulation tasks with diverse levels of cooperation and dexterity (TOP); and their symmetric counterparts (Below).

once performance stabilizes, environment variations (object poses, physical parameters) are gradually introduced. **Safety penalty**: collision and energy penalties are phased in later, avoiding early over-constraint that would produce overly conservative, exploration-averse policies.

4 Experiments

We evaluate our method on six simulated bimanual manipulation tasks: `box-lift`, `table-clean`, `drawer-insert`, `threading`, `bowl-stir`, and `handover`. These tasks span a range of coordination and dexterity challenges (see Fig. 3). We validate the learned policy across all simulated tasks and further deploy it in the real world on two representative tasks: `box-lift` and `table-clean`, showcasing effective transfer from simulation to the real world, enabled by curriculum learning. To control the robot, the policy outputs relative joint position actions a_t, with joint commands computed as $q_{t+1}^{\text{cmd}} = q_t^{\text{cmd}} + \eta \cdot \text{EMA}(a_t)$, where η is a scaling factor and EMA denotes exponential moving average smoothing [25]. Task setups, success criteria, and reward definitions are detailed in Appx. E, while the full real-world setup is provided in Appx. F. All simulation experiments are conducted using NVIDIA Isaac Lab [26].

Baselines and Evaluation Metric We evaluate five PPO-based baselines, each targeting a specific design aspect: action space dimensionality, task decomposition, value function structure, and symmetry handling via equivariant networks or data augmentation. The baselines include: (1) Equivariant PPO (E-PPO), a single 44-DoF equivariant policy; (2) Independent IPPO (IPPO), two independent policies fixed on each arm and trained to handle both subtasks under scene randomization; (3) Equivariant IPPO (E-IPPO), a 22-DoF single-arm policy shared across arms with task encoding, effectively doubling training data compared to IPPO; (4) SYMDEX-c (SM-c), our architecture with a centralized value function; and (5) SYMDEX-aug (SM-aug), which replaces equivariant networks with on-policy symmetry-based data augmentation [5]. All baselines (and ours) use shared hyperparameters (Appx. H), and a detailed comparison is summarized in Appx. G. Task success rate is the primary metric, averaged over five random seeds with 4096 rollouts each in simulation. Real-world evaluation reports both overall task and per-hand subtask success over 30 independent trials.

4.1 Simulation Results

We evaluate SYMDEX on our simulation benchmark against all baselines, where symmetry transformations are randomly applied to the initial state. As shown in Fig. 4, SYMDEX consistently learns all six tasks with success rates exceeding 80%, significantly outperforming the baselines.

Advantage of Task Decomposition Task decomposition is highly beneficial when the subtasks assigned to each arm differ significantly. For example, the baseline E-PPO, which jointly controls the entire system (44 DoF, cf. Tab. 2), succeeds only on `box-lift`, partially on `table-clean`, and fails on the rest. This occurs because in `box-lift` and `table-clean`, both arms perform similar

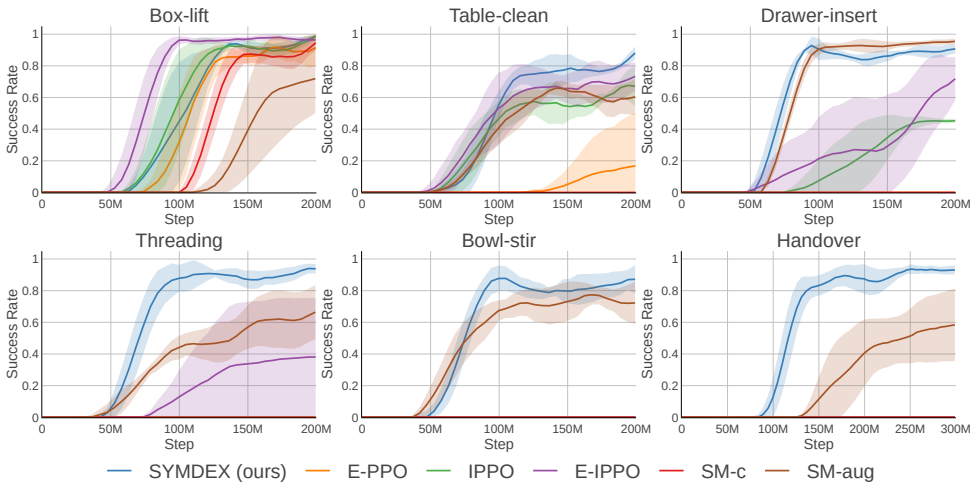


Figure 4: Performance of SYMDEX and baseline methods on six benchmark tasks. SYMDEX consistently learns all six tasks and achieves success rates exceeding 80%, outperforming all baselines.

actions, making joint learning tractable, whereas when arm subtasks diverge, E-PPO's monolithic policy struggles to specialize appropriately.

Moreover, decomposing the task at the subtask level—in addition to at the robot arm level—is critical. Baselines like IPPO and E-IPPO use a decomposed 22 DoF action space, yet each policy remains need to select and perform both subtasks, i.e., a multi-task policy. While IPPO and E-IPPO perform comparably to SYMDEX on `box-lift` and `table-clean`, they fail to generalize to other tasks. In contrast, SYMDEX assigns a policy per subtask, avoiding the issues of multi-task learning.

Impact of $\mathbb{G}$-equivariance/invariance Constraints When comparing SYMDEX to SM-aug—which employs on-policy data augmentation [5]—, our proposed method consistently outperforms across all tasks (see Fig. 4). A similar trend is observed when comparing IPPO and E-IPPO. This performance boost across all tasks highlights the advantage of embedding symmetry priors directly into the network architecture rather than relying solely on data augmentation.

Impact of Centralized Learning We examine the effect of global value functions on multi-arm cooperation. Both E-PPO and SM-c use global critics to estimate total rewards across subtasks; however, our results show that such designs suffer from poor credit assignment in complex, contact-rich settings. Notably, E-PPO outperforms SM-c on `box-lift` and `table-clean`, despite SM-c's reduced action space via task decomposition. This suggests that while decomposition lowers dimensionality, it can introduce uncertainty in joint optimization, hindering accurate reward assignment. Our findings indicate that deglobal value functions, as used in SYMDEX, are more effective for high-dimensional, coordinated manipulation tasks.

Distillation We use a teacher-student distillation approach to train a unified global policy (Sec. 3). We compare three student variants: a vanilla Gaussian policy, an equivariant Gaussian policy, and an equivariant diffusion policy [27]. As shown in Tab. 1, both $\mathbb{G}$-equivariant Gaussian and diffusion policies outperform the vanilla Gaussian policy across all six tasks. This suggest that incorporating equivariant constraints facilitates robust policy distillation. Interestingly, the equivariant Gaussian policy performs comparably to the diffusion variant—likely because the teacher policies used for data collection are Gaussian, allowing the Gaussian policy to fit the dataset effectively.

4.2 Real-World Results

We conduct sim-to-real experiments to evaluate the performance of our distilled policy and its two variants on two real-world tasks: `box-lift` and `table-clean`. The real-world setup is in Fig. 10 and videos are in supplementary material. As shown in Tab. 1-Bottom, the equivariant Gaussian

Method	Box	Table	Drawer	Threading	Bowl	Handover
Gaussian policy (GP)	0.83 ± 0.03	0.74 ± 0.05	0.69 ± 0.09	0.62 ± 0.13	0.75 ± 0.12	0.54 ± 0.23
Equi. GP	0.89 ± 0.01	0.83 ± 0.01	$\mathbf{0.87 \pm 0.07}$	$\mathbf{0.63 \pm 0.17}$	0.87 ± 0.08	$\mathbf{0.86 \pm 0.12}$
Equi. Diffusion policy	$\mathbf{0.91 \pm 0.04}$	$\mathbf{0.84 \pm 0.02}$	$\mathbf{0.87 \pm 0.13}$	0.60 ± 0.1	$\mathbf{0.88 \pm 0.15}$	0.68 ± 0.18

	Box			Table		
	Subtask 1	Subtask 2	Overall	Subtask 1	Subtask 2	Overall
Equi. GP w/o Curriculum	0.2 ± 0.12	0.17 ± 0.23	0.13 ± 0.08	0.13 ± 0.05	0.1 ± 0.08	0.07 ± 0.12
Equi. GP	$\mathbf{0.87 \pm 0.08}$	$\mathbf{0.83 \pm 0.11}$	$\mathbf{0.77 \pm 0.09}$	$\mathbf{0.83 \pm 0.13}$	$\mathbf{0.67 \pm 0.32}$	$\mathbf{0.63 \pm 0.25}$
Equi. Diffusion Policy	0.7 ± 0.20	0.73 ± 0.13	0.6 ± 0.23	0.73 ± 0.15	0.47 ± 0.34	0.4 ± 0.21

Table 1: (Top) Simulation distillation results for six different tasks using three architectural choices. (Below) Real-world performance comparison on `box-lift` and `table-clean`, assessed through qualitative evaluations by human operators.

policy consistently outperforms both its counterpart trained without curriculum and the variant that replaces the Gaussian model with a diffusion model.

First, we observe that removing the curriculum leads to a significant performance drop, highlighting the importance of domain randomization and safety constraints for successful sim-to-real transfer. Second, although the equivariant diffusion policy achieves better distillation results than the Gaussian policy in simulation, the Gaussian policy proves to be more robust in the real world. We attribute this to the homogeneous dataset collected from the teacher policies: the diffusion model struggles to generalize to out-of-distribution observations, particularly under imperfect state estimation from the perception system. In contrast, the Gaussian policy directly fits the teacher policy's distribution, making it more robust to the sim-to-real gap.

4.3 Scaling to Multi-Robot Symmetry

We demonstrate the scalability of SYMDEX on a multi-robot task involving a system of four arms, each equipped with a right dexterous hand. The objective is for two arms to hold the flaps of a cardboard box while the other two arms pick up objects from the table and place them into the box (Fig. 5). Once the objects are inside, the two arms holding the flaps then close the box. Since a constant force is applied to the box flaps to keep them open, the task requires coordinated collaboration among all arms to succeed.

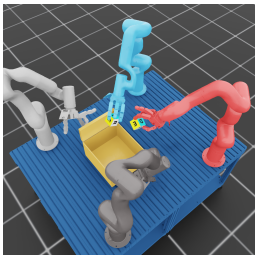


Figure 5: The four-arm system setup.

The four-arm system exhibits symmetry under the group $\mathbb{G} = \mathbb{C}_4 = \{e, g_r, g_r^2, g_r^3 \mid g_r^4 = e\}$, where g_r is a $90°$ rotation about the vertical axis. Following the method described in Sec. 3, we treat each arm as an agent and assign a specific subtask to each. After training, the system successfully completes the task from different orientations, with Fig. 8 visualizing all symmetric scenarios from a fixed camera viewpoint. Additional experiment results are provided in Appx. I.

5 Conclusion

In this work, we presented SYMDEX, a novel RL framework for learning morphological symmetry-aware policies that achieve ambidextrous bimanual manipulation. SYMDEX enables efficient policy learning across six complex dexterous manipulation tasks, enhances policy robustness through symmetry exploitation, and achieves zero-shot sim-to-real transfer on two real-world tasks. Furthermore, we demonstrated the scalability of SYMDEX on a four-arm setup, successfully handling more intricate symmetry groups and multi-agent coordination. We believe that incorporating symmetry as an inductive bias offers a powerful tool for advancing robotic learning, particularly as morphologically inspired humanoid and multi-armed robots become increasingly prominent.

6 Limitations

We acknowledge that the primary limitation of SYMDEX is its reliance on the presence of morphological symmetry within the robotic system. However, we emphasize that such symmetry is common in many modern robotic platforms, including bimanual systems [11, 10], tri-arm robots like Trifinger [28], and humanoid robots [12, 29].

We note that SYMDEX primarily leverages kinematic-level symmetry, where joint positions and end-effector poses are symmetric under group transformation. This design choice allows us to use joint position control, which is sufficient for many manipulation tasks and avoids the need for full dynamic symmetry, as required by torque control. Achieving symmetry at the dynamics level—particularly under reflection—would require the robot components to be true mirror models. While this condition holds for the left and right hands, it does not strictly apply to the arms, which are typically identical in construction rather than mirrored. As a result, their dynamic properties, such as mass distribution and collision avoidance, may not fully follow reflectional symmetry. However, since SYMDEX operates at the kinematic level, this does not significantly impact its effectiveness in practice.

Regarding the failure cases in the real world experiments, we observe that the major issue comes from the perception part. Since our policy is state-based, it depends on accurate multi-object pose tracking, which is difficult in practice. However, our equivariant architecture can also be applied to vision-based inputs, such as RGB-D images and point clouds [30], to improve robustness, which we will leave as future work.

Acknowledgments

This research work has received funding from the Emmy Noether Programme DFG Project Nr. CH 2676/1-1 and the Project Pose Confidences for Telerobotics from Honda Research Institute, Europe.

References

[1] D. Ordoñez Apraez, G. Turrisi, V. Kostic, M. Martin, A. Agudo, F. Moreno-Noguer, M. Pontil, C. Semini, and C. Mastalli. Morphological symmetries in robotics. *The International Journal of Robotics Research*, 2025. doi:10.1177/02783649241282422.

[2] Z. Su, X. Huang, D. Ordoñez Apraez, Y. Li, Z. Li, Q. Liao, G. Turrisi, M. Pontil, C. Semini, Y. Wu, et al. Leveraging symmetry in rl-based legged locomotion control. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6899–6906. IEEE, 2024.

[3] D. Ordoñez Apraez, M. Martin, A. Agudo, and F. Moreno-Noguer. On discrete symmetries of robotics systems: A group-theoretic and data-driven analysis. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi:10.15607/RSS.2023.XIX.053.

[4] M. Mittal, N. Rudin, V. Klemm, A. Allshire, and M. Hutter. Symmetry considerations for learning task symmetric robot policies. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7433–7439. IEEE, 2024.

[5] K. Bao, C. Li, Y. As, A. Krause, and M. Hutter. Toward task generalization via memory augmentation in meta-reinforcement learning. *arXiv preprint arXiv:2502.01521*, 2025. Meta-RL with data-augmentation of the replay bufffer, for locomotion.

[6] D. Butterfield, S. S. Garimella, N.-J. Cheng, and L. Gan. MI-HGNN: Morphology-Informed Heterogeneous Graph Neural Network for Legged Robot Contact Perception. *arXiv preprint arXiv:2409.11146*, 2024. URL `https://arxiv.org/abs/2409.11146`.

[7] F. Xie, S. Wei, Y. Song, Y. Yue, and L. Gan. Morphological-symmetry-equivariant heterogeneous graph neural network for robotic dynamics learning. *arXiv preprint arXiv:2412.01297*, 2024.

[8] H. Zhao, D. Wang, Y. Zhu, X. Zhu, O. L. Howell, L. Zhao, Y. Qian, R. Walters, and R. Platt. Hierarchical equivariant policy via frame transfer. In *Forty-second International Conference on Machine Learning*, 2025.

[9] F. Amadio, A. Colomé, and C. Torras. Exploiting symmetries in reinforcement learning of bimanual robotic tasks. *IEEE Robotics and Automation Letters*, 4(2):1838–1845, 2019.

[10] T. Lin, Z.-H. Yin, H. Qi, P. Abbeel, and J. Malik. Twisting lids off with two hands. In *8th Annual Conference on Robot Learning (CoRL)*, 2024. URL https://openreview.net/forum?id=3wBqoPfoeJ.

[11] B. Huang, Y. Chen, T. Wang, Y. Qin, Y. Yang, N. Atanasov, and X. Wang. Dynamic handover: Throw and catch with bimanual hands. In *Conference on Robot Learning*, pages 1887–1902. PMLR, 2023.

[12] T. Lin, K. Sachdev, L. Fan, J. Malik, and Y. Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids. *arXiv preprint arXiv:2502.20396*, 2025.

[13] F. Lan, S. Wang, Y. Zhang, H. Xu, O. O. Oseni, Z. Zhang, Y. Gao, and T. Zhang. Dexcatch: Learning to catch arbitrary objects with dexterous hands. In *8th Annual Conference on Robot Learning*.

[14] J.-J. Jiang, X.-M. Wu, Y.-X. He, L.-A. Zeng, Y.-L. Wei, D. Zhang, and W.-S. Zheng. Rethinking bimanual robotic manipulation: Learning with decoupled interaction framework. *arXiv preprint arXiv:2503.09186*, 2025.

[15] L. Wang, Y. Zhang, Y. Hu, W. Wang, C. Zhang, Y. Gao, J. Hao, T. Lv, and C. Fan. Individual reward assisted multi-agent reinforcement learning. In *International conference on machine learning*, pages 23417–23432. PMLR, 2022.

[16] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand dexterous manipulation from depth. In *Icml workshop on new frontiers in learning, control, and dynamical systems*, 2023.

[17] G. Tiboni, P. Klink, J. Peters, T. Tommasi, C. D'Eramo, and G. Chalvatzaki. Domain randomization via entropy maximization. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GXtmuiVrOM.

[18] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[19] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[20] M. Zinkevich and T. Balch. Symmetry in markov decision processes and its implications for single agent and multi agent learning. Citeseer, 2001.

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

[22] I. Higgins, S. Racanière, and D. Rezende. Symmetry-based representations for artificial and biological general intelligence. *Frontiers in Computational Neuroscience*, page 28, 2022.

[23] D. Wang, M. Jia, X. Zhu, R. Walters, and R. Platt. On-robot learning with equivariant models. In *6th Annual Conference on Robot Learning*, 2022.

[24] M. Weiler and G. Cesa. General e (2)-equivariant steerable cnns. *Advances in neural information processing systems*, 32, 2019.

[25] T. Chen, E. Cousineau, N. Kuppuswamy, and P. Agrawal. Vegetable peeling: A case study in constrained dexterous manipulation. *arXiv preprint arXiv:2407.07884*, 2024.

[26] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi:10.1109/LRA.2023.3270034.

[27] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[28] M. Wüthrich, F. Widmaier, F. Grimminger, J. Akpo, S. Joshi, V. Agrawal, B. Hammoud, M. Khadiv, M. Bogdanovic, V. Berenz, et al. Trifinger: An open-source robot for learning dexterity. *arXiv preprint arXiv:2008.03596*, 2020.

[29] Y. Ze, Z. Chen, W. Wang, T. Chen, X. He, Y. Yuan, X. B. Peng, and J. Wu. Generalizable humanoid manipulation with improved 3d diffusion policies. *arXiv preprint arXiv:2410.10803*, 2024.

[30] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. J. Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021.

[31] M. Weiler, P. Forré, E. Verlinde, and M. Welling. *Equivariant and Coordinate Independent Convolutional Networks*. 2023. URL `https://maurice-weiler.gitlab.io/cnn_book/EquivariantAndCoordinateIndependentCNNs.pdf`.

[32] E. Van der Pol, D. Worrall, H. van Hoof, F. Oliehoek, and M. Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020.

[33] D. Ordoñez Apraez, V. Kostic, G. Turrisi, P. Novelli, C. Mastalli, C. Semini, and M. Pontil. Dynamics harmonic analysis of robotic systems: Application in data-driven koopman modelling. In *6th Annual Learning for Dynamics & Control Conference*, pages 1318–1329. PMLR, 2024.

[34] H. Ryu, J. Kim, H. An, J. Chang, J. Seo, T. Kim, Y. Kim, C. Hwang, J. Choi, and R. Horowitz. Diffusion-edfs: Bi-equivariant denoising generative modeling on se (3) for visual robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18007–18018, 2024. Manipulation of objects in 3D space with SO(3) grasp pose generation.

[35] J. Brehmer, J. Bose, P. De Haan, and T. S. Cohen. Edgi: Equivariant diffusion for planning with embodied agents. *Advances in Neural Information Processing Systems*, 36:63818–63834, 2023. Diffusion policy encoding SO(3) and O(n) (permutation of multiple instances of the same object).

[36] H. Huang, D. Wang, R. Walters, and R. Platt. Equivariant transporter network. In *Proceedings of Robotics: Science and Systems*, 2022.

[37] X. Zhu, D. Wang, O. Biza, G. Su, R. Walters, and R. Platt. Sample efficient grasp learning using equivariant models. In *Robotics: Science and Systems*, 2022. Manipulation of objects in 3D space with SO(3) grasp pose generation.

[38] D. Wang, R. Walters, and R. Platt. $\mathrm{SO}(2)$-equivariant reinforcement learning. In *International Conference on Learning Representations*, 2022.

[39] R. Wang, R. Walters, and R. Yu. Incorporating symmetry into deep dynamics models for improved generalization. *arXiv preprint arXiv:2002.03061*, 2020.

[40] D. Wang, S. Hart, D. Surovik, T. Kelestemur, H. Huang, H. Zhao, M. Yeatman, J. Wang, R. Walters, and R. Platt. Equivariant diffusion policy. *arXiv preprint arXiv:2407.01812*, 2024.

[41] B. Hu, H. Tian, D. Wang, H. Huang, X. Zhu, R. Walters, and R. Platt. Push-grasp policy learning using equivariant models and grasp score optimization. *arXiv preprint arXiv:2504.03053*, 2025.

[42] D. Ordoñez Apraez, A. Agudo, F. Moreno-Noguer, and M. Martin. An adaptable approach to learn realistic legged locomotion without examples. In *2022 international conference on Robotics and automation (ICRA)*, pages 4671–4678. IEEE, 2022.

[43] F. Abdolhosseini, H. Y. Ling, Z. Xie, X. B. Peng, and M. Van de Panne. On learning symmetric locomotion. In *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*, pages 1–10, 2019.

[44] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023.

[45] B. Wen, W. Yang, J. Kautz, and S. Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17868–17879, 2024.

[46] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[47] V. E. Kremer. Quaternions and slerp. In *Embots. dfki. de/doc/seminar ca/Kremer Quaternions. pdf*, 2008.

Appendix

A Background on group and representation theory

Group actions and representations

This section provides a brief overview of the fundamental concepts in group and representation theory, which are used to define symmetry groups of robotic systems and MDPs. For a comprehensive and intuitive background on group and representation theory in machine learning, we refer the reader to Weiler et al. [31].

To begin, we define a group as an abstract mathematical object.

Definition A.1 (Group)**.** *A group is a set $\mathbb{G}$, endowed with a binary composition operator defined as:*

$$\begin{array}{rccc} (\circ): & \mathbb{G}\times\mathbb{G} & \longrightarrow & \mathbb{G} \\ & (g_1, g_2) & \longrightarrow & g_1 \circ g_2, \end{array} \tag{10a}$$

such that the following axioms hold:

$$\textit{Associativity:}\quad (g_1 \circ g_2)\circ g_3 = g_1 \circ (g_2 \circ g_3), \quad \forall\, g_1, g_2, g_3 \in \mathbb{G}, \tag{10b}$$

$$\textit{Identity:}\quad \exists\, e \in \mathbb{G} \textit{ such that } e \circ g = g = g \circ e, \quad \forall\, g \in \mathbb{G}, \tag{10c}$$

$$\textit{Inverses:}\quad \forall\, g \in \mathbb{G},\ \exists\, g^{-1} \in \mathbb{G} \textit{ such that } g \circ g^{-1} = e = g^{-1} \circ g. \tag{10d}$$

We focus on symmetry groups—that is, groups of transformations acting on a set $\mathcal{X}$ where each transformation is a bijection preserving an intrinsic property. For example, if $\mathcal{X}$ represents the states of a dynamical system, the invariant property might be the state's energy (see Fig. 1).

Definition A.2 (Group action on a set [31])**.** *Let $\mathcal{X}$ be a set endowed with the symmetry group $\mathbb{G}$. The (left) group action of the group $\mathbb{G}$ on the set $\mathcal{X}$ is a map:*

$$\begin{array}{rccc} (\triangleright): & \mathbb{G}\times\mathcal{X} & \longrightarrow & \mathcal{X} \\ & (g, \boldsymbol{x}) & \longrightarrow & g \triangleright \boldsymbol{x} \end{array} \tag{11a}$$

that is compatible with the group composition and identity element $e \in \mathbb{G}$, such that the following properties hold:

$$\textit{Identity:}\quad e \triangleright \boldsymbol{x} = \boldsymbol{x}, \qquad \forall\, \boldsymbol{x} \in \mathcal{X} \tag{11b}$$

$$\textit{Associativity:}\quad (g_1 \circ g_2) \triangleright \boldsymbol{x} = g_1 \triangleright (g_2 \triangleright \boldsymbol{x}), \qquad \forall\, g_1, g_2 \in \mathbb{G}, \forall\, \boldsymbol{x} \in \mathcal{X} \tag{11c}$$

We are primarily interested in studying symmetry transformations on sets with a vector space structure. In most practical cases, the group action on a vector space is linear, allowing symmetry transformations to be represented as linear invertible maps. These maps can be expressed in matrix form once a basis for the space is chosen.

Definition A.3 (Linear group representation)**.** *Let $\mathcal{X}$ be a vector space endowed with the symmetry group $\mathbb{G}$. A* linear representation *of $\mathbb{G}$ on $\mathcal{X}$ is a map, denoted by $\boldsymbol{\rho}_{\mathcal{X}}$, between symmetry transformation and invertible linear maps on $\mathcal{X}$ (i.e., elements of the general linear group $\mathbb{GL}(\mathcal{X})$):*

$$\begin{array}{rccc} \boldsymbol{\rho}_{\mathcal{X}}: & \mathbb{G} & \longrightarrow & \mathbb{GL}(\mathcal{X}) \\ & g & \longrightarrow & \boldsymbol{\rho}_{\mathcal{X}}(g), \end{array} \tag{12a}$$

such that the following properties hold:

$$\textit{composition}: \boldsymbol{\rho}_{\mathcal{X}}(g_1 \circ g_2) = \boldsymbol{\rho}_{\mathcal{X}}(g_1)\boldsymbol{\rho}_{\mathcal{X}}(g_2), \qquad \forall\, g_1, g_2 \in \mathbb{G}, \tag{12b}$$

$$\textit{inversion}: \boldsymbol{\rho}_{\mathcal{X}}(g^{-1}) = \boldsymbol{\rho}_{\mathcal{X}}(g)^{-1}, \qquad \forall\, g \in \mathbb{G}. \tag{12c}$$

$$\textit{identity}: \boldsymbol{\rho}_{\mathcal{X}}(g \circ g^{-1}) = \boldsymbol{\rho}_{\mathcal{X}}(e) = \boldsymbol{I}, \tag{12d}$$

Whenever the vector space is of finite dimension $|\mathcal{X}| = n < \infty$, linear maps admit a matrix form $\boldsymbol{\rho}_{\mathcal{X}}(g) \in \mathbb{R}^{n\times n}$, once a basis set $\mathbb{I}_{\mathcal{X}}$ for the vector space $\mathcal{X}$ is chosen. In this case, Eqs. (12b)

to (12d) show how the composition and inversion of symmetry transformations translate to matrix multiplication and inversion, respectively. Moreover, $\boldsymbol{\rho}_{\mathcal{X}}$ allows to express a (linear) group action (Def. A.2) as a matrix-vector multiplication:

$$(\triangleright): \begin{array}{ccc} \mathbb{G} \times \mathcal{X} & \longrightarrow & \mathcal{X} \\ (g, \boldsymbol{x}) & \longrightarrow & g \triangleright \boldsymbol{x} := \boldsymbol{\rho}_{\mathcal{X}}(g)\boldsymbol{x} \end{array} \tag{12e}$$

Definition A.4 (Tensor product representation)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be (finite-dimensional) vector spaces endowed with a common symmetry group $\mathbb{G}$. Denote by $\boldsymbol{\rho}_{\mathcal{X}} : \mathbb{G} \to \mathbb{GL}(\mathcal{X})$, and $\boldsymbol{\rho}_{\mathcal{Y}} : \mathbb{G} \to \mathbb{GL}(\mathcal{Y})$ the corresponding linear representations. The tensor product representation is defined through the Kronecker product of the representations of group actions on the vector spaces:*

$$(\boldsymbol{\rho}_{\mathcal{X}} \otimes \boldsymbol{\rho}_{\mathcal{Y}}): \begin{array}{ccc} \mathbb{G} & \longrightarrow & \mathbb{GL}(\mathcal{X} \otimes \mathcal{Y}) \\ g & \longrightarrow & \boldsymbol{\rho}_{\mathcal{X}}(g) \otimes \boldsymbol{\rho}_{\mathcal{Y}}(g), \end{array} \tag{13}$$

Note A.1. *Whenever denoting group actions by $(\triangleright_{\mathcal{X}})$ and $(\triangleright_{\mathcal{Y}})$, we will use the notation $\triangleright_{\mathcal{X}\otimes\mathcal{Y}}$ to denote the group action on the tensor product space $\mathcal{X} \otimes \mathcal{Y}$. Such that:*

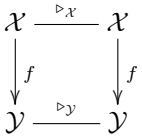
$$\triangleright_{\mathcal{X}\otimes\mathcal{Y}}: \begin{array}{ccc} \mathbb{G} \times (\mathcal{X} \otimes \mathcal{Y}) & \longrightarrow & (\mathcal{X} \otimes \mathcal{Y}) \\ (g, \boldsymbol{x} \otimes \boldsymbol{y}) & \longrightarrow & [\boldsymbol{\rho}_{\mathcal{X}}(g) \otimes \boldsymbol{\rho}_{\mathcal{Y}}(g)](\boldsymbol{x} \otimes \boldsymbol{y}) \end{array} \tag{14}$$

Maps between symmetric vector spaces

We will frequently study and use linear and non-linear maps between symmetric vector spaces. Our focus is on maps that preserve entirely or partially the group structure of the vector spaces. These types of maps can be classified as $\mathbb{G}$-equivariant, $\mathbb{G}$-invariant maps:

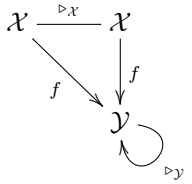
Definition A.5 ($\mathbb{G}$-equivariant and $\mathbb{G}$-invariant maps)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be two vector spaces endowed with the same symmetry group $\mathbb{G}$, with the respective group actions $\triangleright_{\mathcal{X}}$ and $\triangleright_{\mathcal{Y}}$. A map $f : \mathcal{X} \mapsto \mathcal{Y}$ is said to be $\mathbb{G}$-equivariant if it commutes with the group action, such that:*

$$\begin{aligned} g \triangleright_{\mathcal{Y}} \boldsymbol{y} = g \triangleright_{\mathcal{Y}} f(\boldsymbol{x}) = f(g \triangleright_{\mathcal{X}} \boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathcal{X}, g \in \mathbb{G}. \\ \boldsymbol{\rho}_{\mathcal{Y}}(g) f(\boldsymbol{x}) = f(\boldsymbol{\rho}_{\mathcal{X}}(g)\boldsymbol{x}) \end{aligned} \iff \tag{15a}$$



A specific case of $\mathbb{G}$-equivariant maps are the $\mathbb{G}$-invariant ones, which are maps that commute with the group action and have trivial output group actions $\triangleright_{\mathcal{Y}}$ such that $\boldsymbol{\rho}_{\mathcal{Y}}(g) = \boldsymbol{I}$ for all $g \in \mathbb{G}$. That is:

$$\begin{aligned} \boldsymbol{y} = g \triangleright_{\mathcal{Y}} f(\boldsymbol{x}) = f(g \triangleright_{\mathcal{X}} \boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathcal{X}, g \in \mathbb{G}. \\ \boldsymbol{y} = \boldsymbol{\rho}_{\mathcal{Y}}(g) f(\boldsymbol{x}) = f(\boldsymbol{\rho}_{\mathcal{X}}(g)\boldsymbol{x}) \end{aligned} \iff \tag{15b}$$



B Symmetries in MDPs

This section introduces a formal definition and notation of symmetries in POMDPs, based on the previous works of [1, 20, 32].

Definition B.1 (Symmetric POMDP)**.** *A POMDP $(\mathcal{S}, \mathcal{A}, r, \tau, \rho_0, \gamma, \mathcal{O}, \boldsymbol{\sigma})$ possess the symmetry group $\mathbb{G}$ when the state and action spaces $\mathcal{S}$ and $\mathcal{A}$ admit group actions $(\triangleright_{\mathcal{S}})$ and $(\triangleright_{\mathcal{A}})$, and (r, τ, ρ_0) are all $\mathbb{G}$-invariant. That is, if for every $g \in \mathbb{G}$, $s, s' \in \mathcal{S}$, and $a \in \mathcal{A}$, we have:*

$$\tau(g \triangleright_{\mathcal{S}} s' \mid g \triangleright_{\mathcal{S}} s, g \triangleright_{\mathcal{A}} a) = \tau(s' \mid s, a), \qquad \rho_0(g \triangleright_{\mathcal{S}} s) = \rho_0(s), \qquad r(g \triangleright_{\mathcal{S}} s, g \triangleright_{\mathcal{A}} a) = r(s, a). \tag{16}$$

*POMDP's satisfying Eq. (2) are constrained to have **optimal** policy and value functions satisfying:*

$$\underbrace{g \triangleright_{\mathcal{A}} \pi^*(\boldsymbol{\sigma}(s)) = \pi^*(\boldsymbol{\sigma}(g \triangleright_{\mathcal{S}} s))}_{\text{Policy } \mathbb{G}\text{-equivariance}}, \qquad \underbrace{V^*(\boldsymbol{\sigma}(s)) = V^*(\boldsymbol{\sigma}(g \triangleright_{\mathcal{S}} s))}_{\text{Value function } \mathbb{G}\text{-invariance}}, \quad \forall\, s \in \mathcal{S},\ g \in \mathbb{G}. \text{ (refer to [20])} \tag{17}$$

Proposition B.1 (Conditions for optimality [33])**.** *Given the $\mathbb{G}$-equivariance constraint on the **optimal** policy π^* and the $\mathbb{G}$-invariance of the optimal value function V^* in Eq. (17) of a symmetric POMDP, any parametric policy $\pi_{\boldsymbol{\theta}} \colon \mathcal{O} \to \mathcal{A}$ and value function $V_{\boldsymbol{\theta}} \colon \mathcal{O} \to \mathbb{R}$ can be made $\mathbb{G}$-equivariant and $\mathbb{G}$-invariant, respectively, if the observation function $\boldsymbol{\sigma}$ is $\mathbb{G}$-equivariant, thus endowing the observation space with the same symmetry group $\mathbb{G}$ and group action $\triangleright_{\mathcal{O}}$.*

This holds because for the composition of two functions to be $\mathbb{G}$-equivariant ($\pi_{\boldsymbol{\theta}} \circ \boldsymbol{\sigma} \colon \mathcal{S} \to \mathcal{A}$) or $\mathbb{G}$-invariant ($V_{\boldsymbol{\theta}} \circ \boldsymbol{\sigma} \colon \mathcal{S} \to \mathbb{R}$), both functions must be $\mathbb{G}$-equivariant, such that:

$$g \triangleright_{\mathcal{A}} \pi_{\boldsymbol{\theta}}(\boldsymbol{\sigma}(\boldsymbol{s})) = \pi_{\boldsymbol{\theta}}(g \triangleright_{\mathcal{O}} \boldsymbol{\sigma}(\boldsymbol{s})) \quad = \pi_{\boldsymbol{\theta}}(\boldsymbol{\sigma}(g \triangleright_{\mathcal{S}} \boldsymbol{s})), \qquad \forall\, \boldsymbol{s} \in \mathcal{S},\ g \in \mathbb{G}. \tag{18}$$

$$V_{\boldsymbol{\theta}}(\boldsymbol{\sigma}(\boldsymbol{s})) = V_{\boldsymbol{\theta}}(g \triangleright_{\mathcal{O}} \boldsymbol{\sigma}(\boldsymbol{s})) \quad = V_{\boldsymbol{\theta}}(\boldsymbol{\sigma}(g \triangleright_{\mathcal{S}} \boldsymbol{s})), \tag{19}$$

C Related Work

Symmetry in robotic manipulation Recent works leverage inherent rotational symmetries in 3D environments to design $\mathbb{SE}_3$-, $\mathbb{SO}_3$-, or $\mathbb{SO}_2$-equivariant grasping and pose estimation pipelines [34, 35, 36, 37, 38, 39, 40, 41]. These approaches typically define the MDP's action as the target task-space configuration and use off-the-shelf inverse kinematics (IK) solvers with built-in collision avoidance. In contrast, our method focuses on multi-robot manipulation environments with the action space defined in generalized coordinates, forcing the policy to implicitly learn collision avoidance, in-hand manipulation, and IK. Furthermore, our work focuses on leveraging the morphological symmetries [1] of the manipulation MDP, rather than the environmental symmetries of Euclidean space. Consequently, learned policies are equivariant only to *finite* subgroups of $\mathbb{E}_3$, because practical manipulation environments rarely exhibit full $\mathbb{E}_3$-symmetry—joint limits and workspace obstacles break the symmetries of the continuous group (see Def. 2.1).

Morphological symmetry in reinforcement learning Considering morphological symmetry priors as an inductive bias is a trend in state-of-the-art robotics research to enhance sample efficiency and policy generalization. There are two main approaches to leverage the symmetry priors of Eq. (3), namely employing equivariant network and data augmentation [1, 20, 32]. We studied both methods in our experiments and demonstrated the superior performance of equivariant network when the symmetry is properly defined. However, existing works focus on quadrupedal locomotion [2, 4, 42, 43], and in our work we investigate bimanual (and multi-arm) dexterous manipulation.

Reinforcement learning for (bimanual) dexterous manipulation Bimanual dexterous manipulation is a well-known challenging problem in robotics. Recent works focus on specific tasks, underscoring the problem's complexity. For example, [10] addresses unscrewing a lid, while [11] and [13] focus on handover/catch scenarios between arms. Notably, [13] presents simulation-only results, and both [10] and [11] simplify the system by reducing degree of freedom (DoF)—[10] fixes the dual arms and controls only the hands, and [11] locks several arm joints—thus shrinking the exploration space and avoiding the task's complexity. In contrast, our work maintains full control over all DoF in both arms and hands, preserving the inherent richness—and challenge—of the original problem.

Sim-to-real Transfer A key challenge is transferring trained policies to the real world. Two primary strategies have emerged for sim-to-real transfer. Teacher-student distillation has been successfully applied in dexterous manipulation [16, 25, 44]. This approach leverages privileged simulation information to teach a student policy that operates under realistic sensory constraints; our method builds on this by incorporating permutation invariance during distillation. The second strategy, curriculum learning, automatically increases task difficulty to improve both generalization and policy robustness [17, 18]. For example, in [17], it directly maximizes the entropy of the environment distribution as long as the the success rate is sufficiently high. We use a similar idea, but simplifiy the maximum entropy objective to a fixed step curriculum.

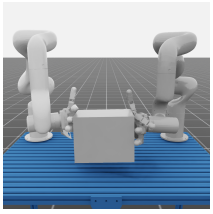
D Pseudoalgorithm

Algorithm 1 Symmetric Dexterity (SYMDEX)

Input: number of agents and tasks N, initial policies $\{\pi_k\}_{k=1}^N$, initial value functions $\{V_k\}_{k=1}^N$, horizon length T, update-to-data (UTD) ratio G.
for each iteration **do**
  for $t = 1, \cdots, T$ **do**
    Observe state $\boldsymbol{s}_t$ and construct observation $\boldsymbol{o}_t = \boldsymbol{\sigma}(\boldsymbol{s}_t, \mathbb{I}_k)$.
    Sample action $\{\boldsymbol{a}_t^n \sim \pi_{\mathbb{I}_{k_n}}(\boldsymbol{o}^{n_t, \mathbb{I}_{k_n}})\}_{n=1}^N$ for each agent-task pair.
    Concatenate for global action $\boldsymbol{a}_t = \oplus_{n \in \mathbb{N}} \boldsymbol{a}_t^n$.
    Execute action $\boldsymbol{a}_t$ in the environment and collect data $\{(\boldsymbol{o}_t^{n,\mathbb{I}_{k_n}}, \boldsymbol{a}_t^n, \boldsymbol{r}_t^{n,\mathbb{I}_{k_n}}, \boldsymbol{o}_{t+1}^{n,\mathbb{I}_{k_n}})\}_{n=1}^N$.
  end for
  Compute advantage estimates $\{\boldsymbol{\Lambda}^n\}_{n=1}^N$ using $V_{\mathbb{I}_{k_n}}$.
  for $g = 1, \cdots, G$ **do**
    for $n = 1, \cdots, N$ **do**
      Sample a batch B_g from $\{(\boldsymbol{o}_t^{n,\mathbb{I}_{k_n}}, \boldsymbol{a}_t^n, \boldsymbol{r}_t^{n,\mathbb{I}_{k_n}}, \boldsymbol{o}_{t+1}^{n,\mathbb{I}_{k_n}})\}$.
      Update policy $\pi_{\mathbb{I}_{k_n}}$ on PPO loss.
      Update value function $V_{\mathbb{I}_{k_n}}$ on MSE loss.
    end for
  end for
end for

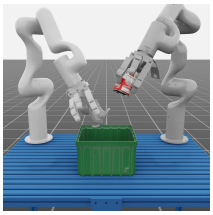
E Environment Details

In this section, we provide a detailed description for all six tasks, including task descriptions, success criteria, and reward functions. For all tasks, the episode length is 100.



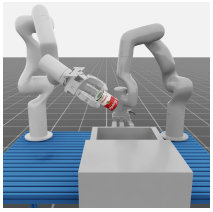
`Box-lift`: The goal is to use both hands to lift a box and hold it at a target pose. Each subtask involves one hand approaching the box from one side and lifting it in coordination with the other hand. The two subtasks are identical but mirrored, requiring tight cooperation between both agents.

Success criteria: The box must be held at the target pose for 20 consecutive steps.
Reward functions for both subtasks: (1) A hand alignment reward that encourages the palm to align with the side of the box; (2) A box pose reward that encourages the box's position and orientation to match the target.



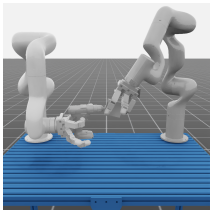
`Table-clean`: The goal is to clean objects from the workbench by placing them into a basket. Subtask 1 involves directly picking and placing the object into the basket. Subtask 2 involves picking up the object, waiting until the other agent completes its task, and then placing the object. To avoid collisions, the hand closer to the basket is expected to place its object first, while the other waits until the first has finished. Thus, the hands must coordinate their timing.

Success criteria: Both objects must be successfully placed inside the basket without any collisions.
Reward functions for both subtasks: (1) A reaching reward between finger and the object; (2) An object distance reward to encourage moving the object toward the basket; (3) A success bonus for placing the object inside the basket.
Additional reward for subtask 2: (4) A waiting reward to encourage proper timing and coordination with the other agent.



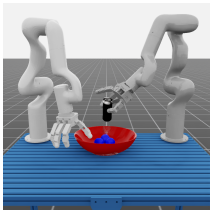
`Drawer-insert`: The goal is to place an object into a drawer. Subtask 1 involves directly picking up the object and placing it into the open drawer. Subtask 2 involves pulling the drawer open, waiting until the object is placed inside, and then pushing the drawer closed. The subtasks are loosely coupled, therefore requiring minimal coordination.

Success criteria: The object is inside the drawer, and the drawer is fully closed for 20 consecutive steps.
Reward functions for subtask 1: (1) A reaching reward for between finger and the object; (2) An object distance reward to encourage moving the object toward the drawer; (3) A success bonus for placing the object inside the drawer.
Reward functions for subtask 2: (4) A pulling reward for opening the drawer; (5) A pushing reward for closing it.



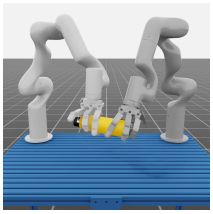
`Threading`: The goal is to thread a drill into a holed cube in mid-air. Subtask 1 involves grasping the drill naturally and inserting its pin into the hole of the cube. Subtask 2 involves picking up the cube, reorienting it so that the hole faces the drill, and maintaining alignment. This task requires precise bimanual coordination and synchronization for successful insertion.

Success criteria: The drill pin must remain inside the cube's hole for 20 consecutive steps.
Reward functions for subtask 1: (1) A hand alignment reward to align the palm with the drill; (2) A drill pose reward to encourage lifting it to the correct mid-air position; (3) A drill-cube distance reward to bring the drill closer to the cube.
Reward functions for subtask 2: (4) A reaching reward to guide the fingers to the cube; (5) A cube distance reward to move and align the cube with the drill.



`Bowl-stir`: The goal is to use an egg-beater to stir balls inside a bowl. Subtask 1 involves pushing the bowl to the center and stabilizing it for stirring. Subtask 2 involves picking up the egg-beater, reorienting it to face downward, and stirring the balls inside the bowl. This task emphasizes everyday dexterity, particularly the challenge of in-hand reorientation.

Success criteria: The egg-beater must be aligned above the bowl and positioned correctly for stirring.
Reward functions for subtask 1: (1) A hand alignment reward to align the palm with the bowl; (2) A bowl pose reward to encourage centering and stabilization.
Reward functions for subtask 2: (3) A reaching reward to guide the hand to the egg-beater; (4) An egg-beater distance reward to position it correctly above the bowl; (5) A stirring reward based on the motion (velocity) of the balls inside the bowl.



`Handover`: The goal is to use the closer hand to grasp a bottle from the table and hand it over to the other hand. Subtask 1 involves grasping the bottle in a way that facilitates the handover and passing it to the other hand. Subtask 2 involves receiving the bottle from the other hand and holding it steadily in mid-air. The main challenges lie in grasp planning and ensuring a smooth, coordinated transition between the hands.

Success criteria: The hand farther from the bottle must hold it steadily for 20 consecutive steps.
Reward functions for subtask 1: (1) A reaching reward to guide the hand to the bottle; (2) A

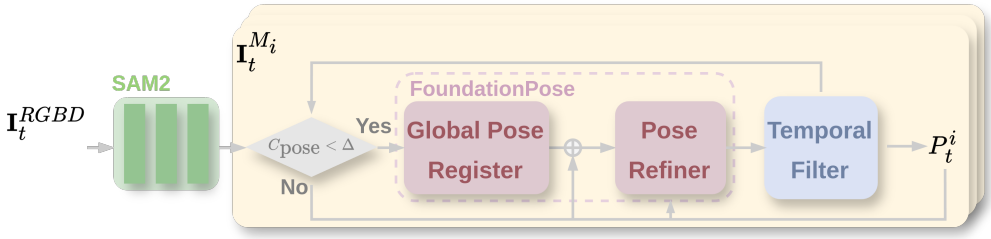


Figure 6: Overview of Perception

bottle pose reward to encourage lifting it to the correct mid-air position; (3) A releasing reward that penalizes excessive holding force, encouraging proper release during handover.
Reward functions for subtask 2: (4) A hand alignment reward to align the palm with the bottle; (5) A bottle pose reward to encourage stable holding after receiving the bottle.

F Real world System

Robot Platform, Sensors and Control Our robotic platform consists of two 6-DoF xArm UF850 arms, each equipped with a 16-DoF Allegro Hand V4, yielding a total of 44 degrees of freedom. The system operates over a 1.2×0.8m tabletop workspace under standard safety constraints. A low-level joint-level PD controller runs at 120Hz, while policy inference is executed at 20Hz. Perception is provided by a single egocentric ZED2i RGB-D camera mounted between the arms. We integrate FoundationPose [45] and SAM2 [46] for robust multi-object tracking.

Perception Pipeline Our perception pipeline (Fig. 6) combines FoundationPose [45] and SAM2 [46] to achieve robust, real-time 6D object pose tracking in cluttered and dynamic scenes. The input consists of RGB-D frames captured at 1080p and 30 FPS from a single ZED2i camera mounted between the robot arms. We operate the camera in ultra mode to maximize depth range and preserve Z-accuracy along the sensing axis, which is crucial for high-precision pose estimation.

For each incoming frame, FoundationPose is executed in parallel for all known objects to predict their 6D poses. While FoundationPose is robust under typical conditions and performs well on standard benchmarks, it fails to recover object pose when faced with rapid motion or complete occlusion.

To handle such cases, we integrate SAM2 for multi-object segmentation and tracking. For each object, we render expected RGB and depth images using a lightweight offscreen renderer based on the object's CAD model. These rendered views are compared against the observed images from the ZED2i camera. Pose confidence is computed by measuring photometric and geometric discrepancies between the rendered and observed RGB-D images. Specifically, we define the confidence score as:

$$c = \exp\left(-\frac{1}{\sum_i M^{(i)}} \sum_i M^{(i)} \left(\left\| I_{\text{obs}}^{(i)} - I_{\text{rend}}^{(i)} \right\|_1 + \lambda \cdot \left\| D_{\text{obs}}^{(i)} - D_{\text{rend}}^{(i)} \right\|_1 \right)\right) \tag{20}$$

where $M^{(i)}$ is a binary foreground mask obtained from SAM2. I_{obs}, I_{rend} denote observed and rendered RGB images, D_{obs}, D_{rend} denote depth images, N is the number of valid pixels, and λ balances color and depth contributions. If the confidence $c < 0.5$, the object is deemed lost, and its pose is re-initialized using FoundationPose without relying on temporal priors.

To mitigate jitter and ensure smooth input to the policy, we apply SLERP interpolation [47] for rotations and linear interpolation for translations in SE(3) across consecutive pose estimates, followed by exponential moving average (EMA) filtering. This ensures temporally coherent trajectories and aligns the pose update rate with the policy's inference frequency of 20 FPS.

G Baseline

Algorithms	E-PPO	IPPO	E-IPPO	SM-c	SM-aug	**SYMDEX (Ours)**
# of Policies	1	2	1	2	2	2
# of Tasks per Policy	2	2	2	1	1	1
Action Dim. per Policy	44	22	22	22	22	22
# of Value Functions	1	2	1	1	2	2
Uses Equi. Network	Yes	No	Yes	Yes	No	Yes

Table 2: Comparison design choices of the five baselines and SYMDEX.

H Hyperparameters

We list the hyperparameters used for all baselines. Since all methods, including SYMDEX, use PPO as the backbone algorithm, they share identical hyperparameters to ensure a fair comparison. Additionally, we use the same set of hyperparameters across all tasks—except for the entropy coefficient (Tab. 4)—highlighting the robustness of our method to hyperparameter tuning. As shown in the table, we generally recommend starting with an entropy coefficient of 0.01 for new tasks. If the task does not require extensive exploration, this can be reduced to 0.005.

Hyperparameters	Values
Num. Environments	$4,096$
Critic Learning Rate	5×10^{-4}
Actor Learning Rate	3×10^{-4}
Optimizer	Adam
Batch Size	$32,768$
Horizon Length	64
UTD Ratio	8
Ratio Clip	0.15
λ for GAE	0.95
Discount Factor (γ)	0.99
Gradient Clipping	0.5
Critic Network	$[256, 256, 256]$
Actor Network	$[256, 256, 256]$
Curriculum: Threshold	0.7
Curriculum: Update Freq.	100
Curriculum: Total Step	10

Table 3: Hyperparameter setup for **all methods** and **all tasks**.

	Entropy Coefficient
`Box-lift`	0.0
`Table-clean`	0.005
`Drawer-insert`	0.01
`Threading`	0.01
`Bowl-stir`	0.01
`Handover`	0.005

Table 4: The entropy coefficient used for six tasks.

We list the final domain randomization and safety penalty values in Tab. 5. We split the curriculum into 10 stages (as shown in Tab. 3), where each stage increases the level of randomization and penalty scale, allowing the agent to adapt progressively. For every 100 policy updates, we track

the agent's success rate during training; if the success rate exceeds a predefined threshold 0.7, the agent advances to the next stage. By simplifying the environment in the early stages, the agent can first focus on mastering the core task before dealing with harder, more variable situations—enabling more stable and effective training.

	`Box-lift`	`Table-clean`	`Drawer-insert`	`Threading`	`Bowl-stir`	`Handover`
Obj. Mass(kg)	$[0.1, 0.5]$	$[0.02, 0.2]$	$[0.02, 0.2]$	$[0.1, 0.3]$	$[0.02, 0.2]$	$[0.1, 0.4]$
Obj. Init. Pos.(cm)	$+\mathcal{U}(-0.1, 0.1)$	$+\mathcal{U}(-0.1, 0.1)$	$+\mathcal{U}(-0.15, 0.15)$	$+\mathcal{U}(-0.05, 0.05)$	$+\mathcal{U}(-0.1, 0.1)$	$+\mathcal{U}(-0.1, 0.1)$
Obj. Init. Orien.(rad.)	$+\mathcal{U}(-0.7, 0.7)$	$+\mathcal{U}(-1.57, 1.57)$	$+\mathcal{U}(-1.57, 1.57)$	$+\mathcal{U}(-0.75, 0.75)$	$+\mathcal{U}(-0.6, 0.6)$	$+\mathcal{U}(-1.57, 1.57)$
Obj. Random Force(N)			0.5			
Static Friction			$[0.24, 1.6]$			
Dynamic Friction			$[0.24, 1.6]$			
Restitution			$[0.0, 1.0]$			
Obj. Pose Obs. Noise			$+\mathcal{U}(-0.01, 0.01)$			
Joint Position Noise			$+\mathcal{N}(0, 0.005)$			
Energy Penalty Coeff.			-0.001			
Collision Penalty Coeff.			-1000.0			

Table 5: Domain randomization and safety penalty setup.

I Additional Experiments

I.1 Curriculum Learning

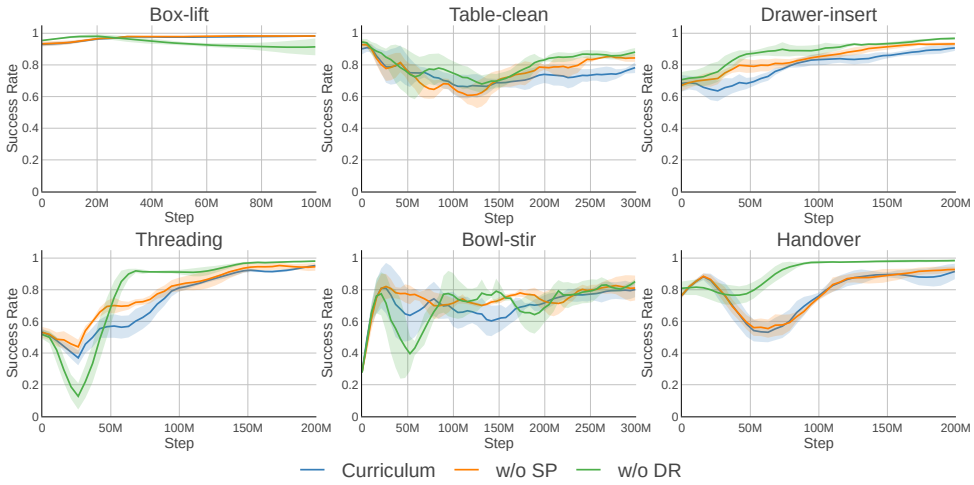


Figure 7: Performance comparison of curriculum learning, curriculum w/o safety penalty (SP), and curriculum w/o domain randomization (DR) on six benchmark tasks.

We report the learning performance during the curriculum learning stage, which we treat as a separate phase. In this stage, we load the best checkpoint from initial training and perform fine-tuning. Since we use PPO as the backbone algorithm, the fine-tuning process remains stable.

We evaluate the effectiveness of curriculum learning by comparing the full curriculum to ablations of its two key components: safety penalty and domain randomization. As shown in Fig. 7, the curriculum initially causes a performance drop across all six tasks, and then the performance is gradually improved as training progresses. We observe that the full curriculum converges more slowly, which is expected given it combines both components. Notably, the agent adapts more easily to the safety penalty than to domain randomization. In the `Box-lift` task, performance remains stable during the curriculum phase, since object randomization and collision penalties were already introduced in the initial training stage.

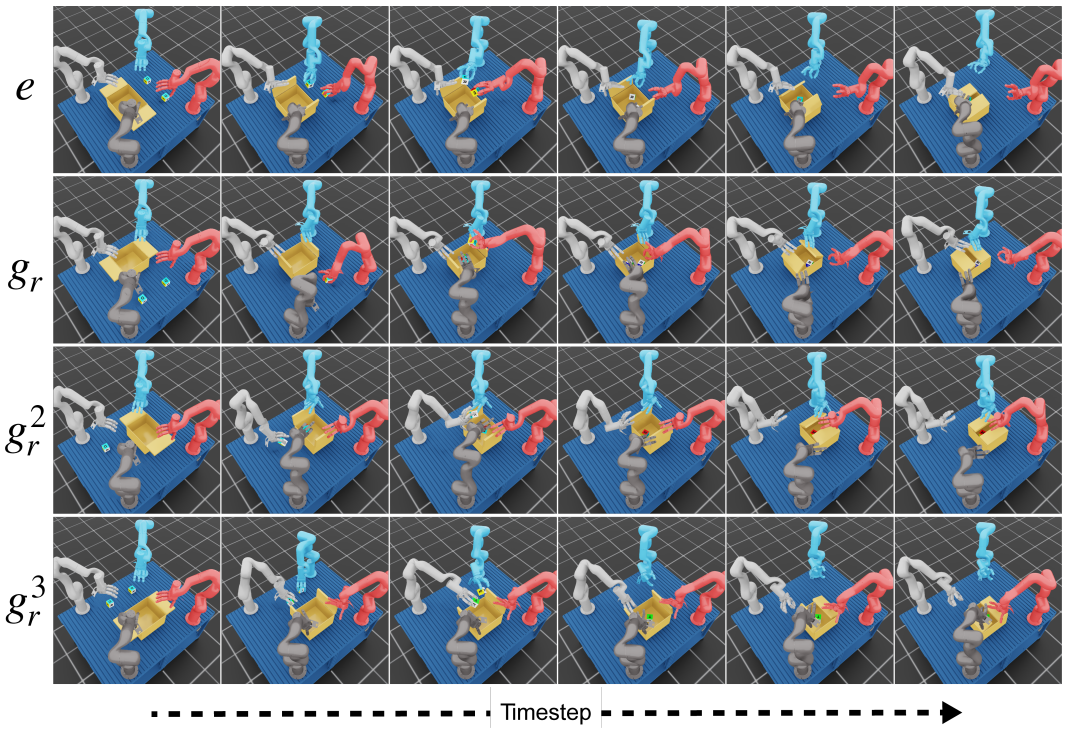


Figure 8: Environment-policy rollout for the multi-arm task starting from state $\boldsymbol{s}_0$ and all its symmetry states $g_r \triangleright_{\mathcal{S}} \boldsymbol{s}_0$, $g_r^2 \triangleright_{\mathcal{S}} \boldsymbol{s}_0$, and $g_r^3 \triangleright_{\mathcal{S}} \boldsymbol{s}_0$. The four-arm system exhibits symmetry under the group $\mathbb{G} = \mathbb{C}_4 = \{e, g_r, g_r^2, g_r^3 \mid g_r^4 = e\}$, where g_r is a 90° rotation about the vertical axis.

I.2 Multi-arm Experiment

We visualize the environment-policy rollouts across all symmetry groups defined by $\mathbb{G} = \mathbb{C}_4 = \{e, g_r, g_r^2, g_r^3 \mid g_r^4 = e\}$, as shown in Fig. 8. The first column shows the initial states, where the object configurations are rotated by 90° about the vertical axis across different symmetry groups. As the policy executes, we observe symmetric behaviors generated by the equivariant policy. Although the four colored arms remain fixed, SYMDEX successfully solves all configurations with consistent performance, as shown in Fig. 9.

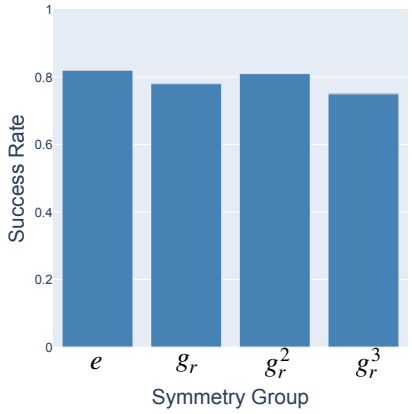


Figure 9: Performance of SYMDEX on the multi-arm task.

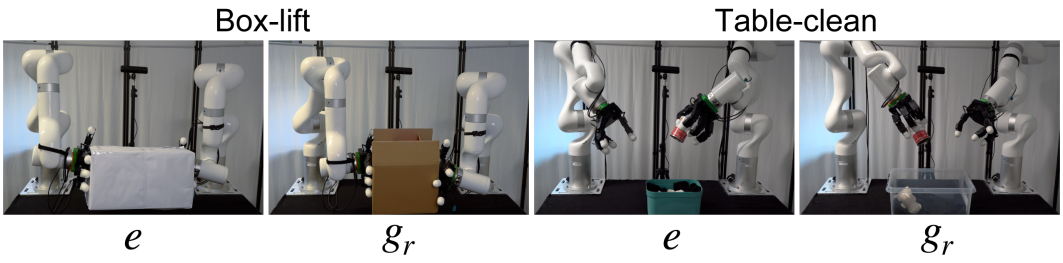


Figure 10: Snapshots from the real-world experiments.

I.3 Real World Experiment

We provide snapshots from real-world experiments on `Box-lift` and `Table-clean`, as shown in Fig. 10, covering both original and symmetric scenarios. In `Box-lift`, both agents manipulate the same object and perform identical subtasks, so there is no significant difference between the original and symmetric settings.

Additionally, we evaluate our policy on out-of-distribution (OOD) objects. For example, in `Box-lift`, we use boxes of varying sizes that were never seen during training; in `Table-clean`, we introduce an OOD toy dog. Thanks to the curriculum learning strategy, our policy generalizes well and successfully handles these OOD cases.

Acronyms

DoF degree of freedom.

E-IPPO Equivariant Independent Proximal Policy Optimization: Trains a single policy for one arm (22-DoF) and applies it to both arms. The symmetry effectively doubles the training data and includes task encoding in the observation to facilitate learning.

E-PPO Equivariant Proximal Policy Optimization: A single policy with an equivariant network controlling the entire system jointly (44-DoF). Represents standard PPO with symmetry handling.

IK inverse kinematics.

IPPO Independent Proximal Policy Optimization: Learns two fixed policies, each assigned to one arm. Due to scene randomization, each policy must learn both subtasks and select the appropriate one.

MDP Markov decision process.

MTMA-POMDP Multi-Task Multi-Agent POMDP.

NN Neural Network.

POMDP Partially Observable MDP.

PPO Proximal Policy Optimization: desc if needed.

RL Reinforcement Learning.

SM-aug SYMDEX-aug: Uses the same training scheme as our method but replaces the equivariant network with on-policy data augmentation using group transformations, a common alternative in symmetry learning.

SM-c SYMDEX-c: Matches our method's policy structure but uses a global value function instead of separate ones, ablating the effect of global value functions commonly used in multi-agent cooperative settings.