META-EVOLVE: CONTINUOUS ROBOT EVOLUTION FOR ONE-TO-MANY POLICY TRANSFER

Anonymous authors
Paper under double-blind review

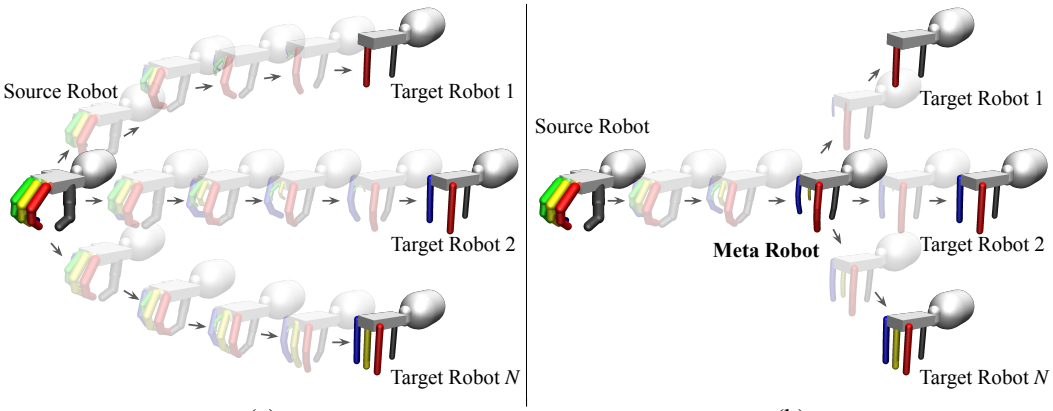


Figure 1: (a) **REvolveR and HERD** (Liu et al., 2022a;b) are methods for transferring policy between a pair of robots using continuous robot evolution. Therefore, to transfer a policy on the source robot to multiple target robots, they must launch multiple independent runs for each target robot. (b) **Our Meta-Evolve** first uses continuous robot evolution to transfer an expert policy from the source robot to a "meta robot" that is closer to all target robots and then to each target robot.

ABSTRACT

We investigate the problem of transferring an expert policy from a source robot to multiple different robots. To solve this problem, we propose a method name Meta-Evolve that uses continuous robot evolution to efficiently transfer the policy to a newly defined meta robot and then to each target robot. Since the meta robot is closer to the target robots, our approach can significantly naive one-to-one policy transfer. We also present three heuristic approaches with theoretical results to determine the meta robot. Experiments have shown that with three target robots, our method is able to improve over the baseline of launching multiple independent one-to-one robot-to-robot policy transfers by up to 2.4× in terms of training and exploration needed.

1 INTRODUCTION

The robotics industry has designed and developed a large number of commercial robots deployed in various applications. How to efficiently learn robotic skills on diverse robots in a scalable fashion? A popular solution is to train a policy for every new robot on every new task from scratch. This is not only inefficient in terms of sample efficiency but also impractical for complex robots due to a large exploration space. Statistic matching Imitation learning (IL) methods that optimize to match the distribution of actions (Ross et al., 2011), transitioned states (Liu et al., 2019; Radosavovic et al., 2020), or reward function (Ng et al., 2000; Ho & Ermon, 2016) could be possible solutions. However, they can only be applied to robots with similar dynamics to yield optimal performance.

Recent advances in evolution-based imitation learning (Liu et al., 2022a;b) inspires us to view this problem from the perspective of policy transferring from one robot to another. The core idea is to interpolate two different robots by producing a large number of intermediate robots between them which gradually evolve from the source robot toward the target robot. These continuously evolving

robots act as the bridge for transferring the policy from the source robot to the target robot. The source robot is usually selected as a robot such that it is easy to collect sufficient demonstrations to train a high-performance expert policy, e.g. a Shadow Hand robot (ShadowRobot, 2005) that can be trained from large-scale human hand demonstration data such as Grauman et al. (2022) and Damen et al. (2018). While the continuous robot evolution has shown success in learning challenging robot manipulation tasks (Liu et al., 2022a), the policy transfer is limited to being between a pair of robots. As illustrated in Figure 1(a), for N different target robots, it requires launching N independent runs of robot-to-robot policy transfer and is not scalable. How can one efficiently transfer a well-trained policy from one source robot to multiple different target robots?

Our intuition is that when robots are designed to complete certain tasks, they often share similar forms of morphology and dynamics to interact with other objects in similar ways. Examples include robot grippers that are all designed to close their fingers to grasp objects and multi-legged robots that are all designed to stretch their legs for agile motion. To transfer the policy to N different target robots to complete similar tasks, it may be possible to find a *meta robot* that is close to all N target robots during continuous robot evolution. Then the robot evolution and policy transfer can first reach the meta robot and then launch N independent robot-to-robot policy transfers toward each of the N target robots. In this way, the initial part of the robot evolution path between the source and the meta robot can be shared among N target robots, and can significantly reduce the amount of exploration and training cost needed if the meta robot is much closer to the target robots. The idea is illustrated in Figure 1(b).

We propose a method named *Meta-Evolve* to instantiate the above idea that uses robot micro-evolution for efficiently transferring a policy from one source robot to multiple target robots. Given the source and an arbitrary number of target robots, our method first matches their morphology. This allows the source and target robots to be represented in the same high-dimensional continuous space of physical parameters and also allows the sampling of new intermediate robots. We propose three different ways to determine the meta robot within the continuous space by heuristics for minimizing the total cost of policy transfer training and exploration.

We showcase our Meta-Evolve on three Hand Manipulation Suite (Rajeswaran et al., 2018) manipulation tasks where the source robot is a five-finger dexterous hand and the target robots are three robot grippers with two, three, and four fingers respectively. Our Meta-Evolve is able to reduce the total amount of simulation epochs and training iterations by up to 2.4$\times$ compared to pairwise robot policy transfer methods (Liu et al., 2022a;b) to reach the same performance on the three target robots. To summarize, our contributions are as follows:

- We introduce a new research problem of one-to-many policy transfer where the goal is to transfer an expert policy from a source robot to multiple target robots. Different from traditional multi-task learning, we propose to use continuous robot evolution as the formulation which allows studying this problem from a robot or task evolutionary perspective.
- We develop a method named Meta-Evolve as a viable solution to the new problem. We propose to employ a meta robot for efficient evolution path sharing among target robots and proposed three different ways of determining the meta robots heuristically.
- We conduct experiments on Hand Manipulation Suite tasks and our Meta-Evolve shows significant gains compared to pairwise policy transfer baselines. This shows that our Meta-Evolve can allow more scalable imitation learning and learning from demonstration for diverse robots.

2 PRELIMINARY

Notation We use bold letters to denote the vector variables. Specially, $\mathbf{0}$ and $\mathbf{1}$ are the all-zero and all-one vectors with proper dimensions respectively. $\odot$ is the element-wise product. For a vector $\boldsymbol{\theta}$, we use $\boldsymbol{\theta}(j)$ to denote its j-th element. We use $\text{MAX}(\cdot)$ and $\text{MIN}(\cdot)$ to denote element-wise maximum and minimum of a set of vectors respectively.

MDP Preliminary We consider a continuous control problem formulated as Markov Decision Process (MDP). It is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^S$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^A$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. A policy $\pi : \mathcal{S} \to \mathcal{A}$ maps a state to an action where $\pi(a|s)$ is the probability of choosing action a at state s. Suppose $\mathcal{M}$ is the set of all MDPs and

$\rho^{\pi,M} = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$ is the episode discounted reward with policy π on MDP $M \in \mathcal{M}$. The optimal policy π^*_M on MDP M is the one that maximizes the expected value of $\rho^{\pi,M}$.

REvolveR and HERD Preliminary Liu et al. (2022b) proposed a technique named REvolveR for transferring policies from one robot to a different robot. Given a well-trained expert policy $\pi^*_{M_\text{S}}$ on a source robot $M_\text{S} \in \mathcal{M}$, its goal is to find the optimal policies $\pi^*_{M_\text{T}}$ on another target robot $M_\text{T} \in \mathcal{M}$. The core idea is to define an evolutionary sequence of intermediate robots that connects the two robots through linear interpolation of physical parameters and sequentially fine-tune the policy on each intermediate robot in the sequence. Liu et al. (2022a) proposed HERD, which extends the idea to robots represented in high-dimensional parameter space, and proposes to optimize the robot evolution path together with the policy. Concretely, given source and target robots $M_\text{S}, M_\text{T} \in \mathcal{M}$ that are parameterized in D-dimensional space respectively, HERD defines a continuous function $F : [0, 1]^D \to \mathcal{M}$ where $F(\mathbf{0}) = M_\text{S}, F(\mathbf{1}) = M_\text{T}$. Suppose the physical parameters of the source and target robots are $\boldsymbol{\theta}_\text{S}, \boldsymbol{\theta}_\text{T} \in \mathbb{R}^D$ respectively. For any evolution parameter $\boldsymbol{\alpha} \in [0, 1]^D$, $F(\boldsymbol{\alpha})$ defines an intermediate robot whose physical parameters are $\boldsymbol{\theta} = (1 - \boldsymbol{\alpha}) \odot \boldsymbol{\theta}_\text{S} + \boldsymbol{\alpha} \odot \boldsymbol{\theta}_\text{T}$. Then an expert policy $\pi^*_{F(\mathbf{0})}$ on the source robot $F(\mathbf{0})$ is optimized by sequentially interacting with each intermediate robot in the sequence $F(\boldsymbol{\alpha}_1), F(\boldsymbol{\alpha}_2), \ldots, F(\boldsymbol{\alpha}_K)$ where $\boldsymbol{\alpha}_K = \mathbf{1}$, until the policy is able to act (near) optimally on each intermediate robot. At robot $F(\boldsymbol{\alpha}_k)$, the optimization objective for finding the next best intermediate robot $F(\boldsymbol{\alpha}_{k+1}) = F(\boldsymbol{\alpha}_k + \boldsymbol{l}_k)$ is formulated as

$$\max_{\boldsymbol{l}_k, ||\boldsymbol{l}_k||_2 = \xi} \quad \max_{\pi} \quad L = \mathbb{E}[\rho^{\pi, F(\boldsymbol{\alpha}_k + \boldsymbol{l}_k)}] - \frac{\lambda}{2} ||\mathbf{1} - (\boldsymbol{\alpha}_k + \boldsymbol{l}_k)||_2^2 \tag{1}$$

which optimizes both the expected policy reward $\mathbb{E}[\rho^{\pi, F(\boldsymbol{\alpha}_k + \boldsymbol{l}_k)}]$ and the L^2 distance to the target robot $||\mathbf{1} - (\boldsymbol{\alpha}_k + \boldsymbol{l}_k)||_2$. For all k, the evolution step size $\xi = ||\boldsymbol{l}_k||_2$ is small enough so that each policy fine-tuning step is a much easier task. After K steps, the policy will eventually be transferred to the target robot $F(\mathbf{1})$. The idea is illustrated in Figure 1(a).

3 One-to-Many Robot-to-Robot Policy Transfer

3.1 Problem Statement

We investigate the problem of transferring a policy from one *source* robot to multiple *target* robots. Formally, we consider a source robot $M_\text{S} \in \mathcal{M}$ and N target robots $M_{\text{T},1}, M_{\text{T},2}, \ldots, M_{\text{T},N} \in \mathcal{M}$ respectively. We assume the state space $\mathcal{S}$, and action space $\mathcal{A}$, reward function $\mathcal{R}$ and discount factor γ of M_S and all $M_{\text{T},i}$ are shared and the difference is their transition dynamics $\mathcal{T}$. Given a well-trained expert policy $\pi^*_{M_\text{S}}$ on a source robot M_S, the goal is to find the optimal policies $\pi^*_{M_{\text{T},i}}$ on each of the target robot $M_{\text{T},i}$. We would like to investigate using the information in π_{M_S} to improve the sample efficiency as well as the final performance of $\pi_{M_{\text{T},i}}$ and also study how the learning of each individual $\pi_{M_{\text{T},i}}$ can help each other during the process.

We approach this problem by defining a *meta* robot $M_\text{Meta} \in \mathcal{M}$ that shares the same state space, action space, reward and discount factor. The robot M_Meta is designed such that M_Meta sits in the middle of M_S and all $M_{\text{T},i}$, i.e. in terms of transition dynamics, M_Meta is closer to all $M_{\text{T},i}$ than M_S while also being closer to M_S than all $M_{\text{T},i}$. Therefore, instead of repeating the process of directly transferring a source robot policy π_S from M_S to $M_{\text{T},i}$ for N times, we first transfer π_S to M_Meta and then from M_Meta to each individual $M_{\text{T},i}$.

3.2 Multi-robot Morphology Matching and Robot Representations

Our problem setting and our proposed solution are based on an assumption that the source robot M_S and target robots $M_{\text{T},i}$ share the same state and action space based on which the intermediate robots can be defined. The assumption is true for a pair of robots as shown in Liu et al. (2022a;b) where the intermediate robots are produced by robot morphology matching and kinematic interpolation. In this subsection, we show that this assumption can be extended to more than two robots.

Morphology Matching The kinematic tree of a robot describes the connection of the bodies and joints. The morphology of the robot, i.e. the topology of its kinematic tree, represents the kinematic behavior of the robot. Given two different robots with different morphology, it has been shown in Liu et al. (2022b) that their kinematic trees can be matched by adding extra nodes and edges. This step

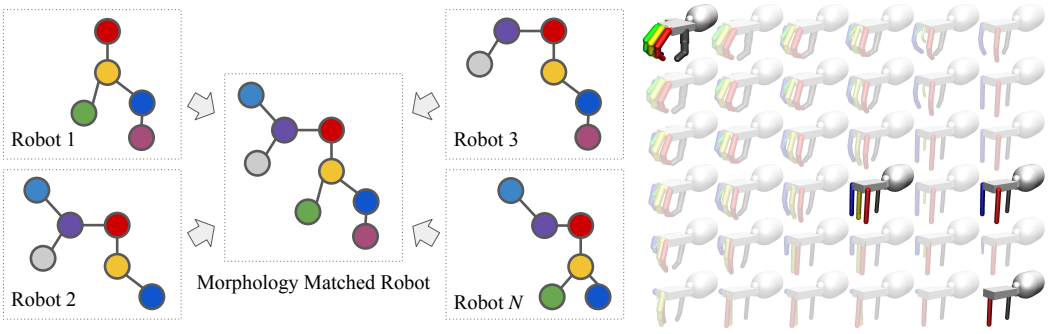


Figure 2: (a) **Morphology matching** of multiple robots. Colored circles denote corresponding robot bodies and straight lines denote robot joints. (b) An example of **robot evolution parameter space** after morphology matching of multiple robots. The four highlighted robots are the source and three target robots used in experiments in Section 5.2 respectively. Other semi-transparent robots are the interpolated robots sampled from the evolution parameter space.

can be easily extended to N robots where $N > 2$. As illustrated in Figure 2(a), by matching proper root nodes and key leaf nodes, the kinematic tree of the morphology-matched robot is essentially a graph union of the kinematic trees of all robots. This means each robot needs to create additional bodies, joints and motors, though they may be all zero in their numbers.

The above kinematic matching process can be automated by an algorithm that achieves the best matching of nodes and edges across N robots. However, in practice, we would like the matching process to include reasonable human intervention with enough robotics knowledge, e.g. matching human hand fingers to robot gripper fingers where the knuckle joints are matched correctly.

Kinematic Interpolation After morphology matching, the state and action space of the robots are matched. The difference in the robot transition dynamics is now only due to the differences in physical parameters, such as shapes and mass of robot bodies, gain and armature of joint motors, etc. Suppose there are D physical parameters. Then each $\boldsymbol{\theta} \in \mathbb{R}^D$ uniquely represents a new robot.

Suppose the physical parameters of the source robot and the N target robots are represented in D-dimensional space as $\boldsymbol{\theta}_{\text{S}} \in \mathbb{R}^D$ and $\boldsymbol{\theta}_{\text{T},1}, \boldsymbol{\theta}_{\text{T},2}, \ldots, \boldsymbol{\theta}_{\text{T},N} \in \mathbb{R}^D$ respectively. In each dimension, we compute the upper and lower bounds of the physical parameters

$$\begin{aligned} \boldsymbol{\theta}_u &= \text{MAX}(\{\boldsymbol{\theta}_{\text{S}}, \boldsymbol{\theta}_{\text{T},1}, \boldsymbol{\theta}_{\text{T},2}, \ldots, \boldsymbol{\theta}_{\text{T},N}\}) \\ \boldsymbol{\theta}_l &= \text{MIN}(\{\boldsymbol{\theta}_{\text{S}}, \boldsymbol{\theta}_{\text{T},1}, \boldsymbol{\theta}_{\text{T},2}, \ldots, \boldsymbol{\theta}_{\text{T},N}\}) \end{aligned} \tag{2}$$

where $\boldsymbol{\theta}_u$ and $\boldsymbol{\theta}_l$ essentially defines the *convex hull* of the set of robot physical parameters in D-dimensional space $\mathcal{H} = [\boldsymbol{\theta}_l(1), \boldsymbol{\theta}_u(1)] \times [\boldsymbol{\theta}_l(2), \boldsymbol{\theta}_u(2)] \times \cdots [\boldsymbol{\theta}_l(N), \boldsymbol{\theta}_u(N)] \in \mathbb{R}^D$ that encompasses the source and all target robots.

We can now use continuous function $F : [0, 1]^D \rightarrow \mathcal{M}$ to define an intermediate robot by interpolation between all pairs of kinematic parameters

$$\boldsymbol{\theta} = (\mathbf{1} - \boldsymbol{\alpha}) \odot \boldsymbol{\theta}_l + \boldsymbol{\alpha} \odot \boldsymbol{\theta}_u \tag{3}$$

where $\boldsymbol{\alpha} \in [0, 1]^D$ is the *evolution parameter* that describes the normalized position of a robot in the convex hull $\mathcal{H}$. Note that by limiting $\boldsymbol{\alpha}$ to be between $\mathbf{0}$ and $\mathbf{1}$, we assume the convex hull $\mathcal{H}$ is the set of all possible intermediate robots. This is a reasonable assumption since an out-of-range parameter can be physically dangerous and also unlikely to be useful in robot continuous interpolation. The convex hull $\mathcal{H}$ also serves as the metric space that measures the distance between any two robots.

In HERD (Liu et al., 2022a), the source and target robots are always represented as $\mathbf{0}$ and $\mathbf{1}$ respectively because when there are only two robots, one of them must be either lower or upper bound. Different from HERD, the source and target robots in our problem are not necessarily $\mathbf{0}$ and $\mathbf{1}$. An example of the resulting robot evolution space and the positions of the source and target robots in the evolution space is illustrated in Figure 2(b). More details can be found in Section B.

3.3 ONE-TO-MANY ROBOT EVOLUTION FOR POLICY TRANSFER

Suppose the source robot and the N target robots are represented by $F(\boldsymbol{\beta}_0) := M_{\text{S}}$ and $F(\boldsymbol{\beta}_1) := M_{\text{T},1}, \ldots, F(\boldsymbol{\beta}_N) := M_{\text{T},N}$ respectively where $\boldsymbol{\beta}_i \in [0,1]^D$. Similar to HERD (Liu et al., 2022a), we employ N robot evolution paths of intermediate robots $\tau_i = (F(\boldsymbol{\alpha}_{i,1})), F(\boldsymbol{\alpha}_{i,2})), \ldots, F(\boldsymbol{\alpha}_{i,K_i}))), i = 1, 2, \ldots, N$ where $K_i \in \mathbb{Z}^+$, $F(\boldsymbol{\alpha}_{i,1}) = F(\boldsymbol{\beta}_0)$ is the source robot and $F(\boldsymbol{\alpha}_{i,K_i}) = F(\boldsymbol{\beta}_i)$ is the target robot i. Following HERD (Liu et al., 2022a), we use intermediate robots $F(\boldsymbol{\alpha}_{i,k})$ as the bridge to transfer the policy to target robot i.

We use K_i training phases of policy optimizations. At training phase k, the policy is trained on policy rollouts from robots sampled on the line $\overline{\boldsymbol{\alpha}_k \boldsymbol{\alpha}_{k+1}}$. The sampling window gradually converges to $\boldsymbol{\alpha}_{k+1}$ during training until the policy is able to achieve enough success rate on $F(\boldsymbol{\alpha}_{k+1})$ before moving on to the next training phase $k+1$. For all k, we set the evolution step size $||\boldsymbol{\alpha}_{i,k} - \boldsymbol{\alpha}_{i,k+1}||_2 = \xi$ to be small enough so that each training phase is an easy sub-task with little policy fine-tuning. Naively following HERD would require training through all the τ_i for N time in total.

However, if the target robots $\boldsymbol{\beta}_i$ are mutually similar but are very different from the source robot, during the start of the training phases, the robot would evolve roughly in similar directions. Therefore, the N robot evolution paths would be close to each other near the start of the paths and the policy optimization might be redundant, as illustrated in Figure 1(a).

We propose to design the N evolution paths by forcing the first $m \in \mathbb{Z}^+$ intermediate robots to be shared among the N paths to address the redundancy issue at the start of the training. Formally,

$$\forall k \leq m, \boldsymbol{\alpha}_{1,k} = \boldsymbol{\alpha}_{2,k} = \cdots \boldsymbol{\alpha}_{N,k} \tag{4}$$

This means that all path will first reach a shared robot $F(\boldsymbol{\alpha}_{i,m})$ before splitting their ways towards each target robot. We name the last shared robot "**meta robot**", denoted by $F(\boldsymbol{\beta}_{\text{Meta}}) := F(\boldsymbol{\alpha}_{i,m})$. To summarize, given a defined meta robot, the policy transfer to N target robots consists of the following two steps:

1. Transfer source expert policy $\pi^*_{F(\boldsymbol{\beta}_0)}$ using HERD from the source robot $F(\boldsymbol{\beta}_0)$ to the meta robot $F(\boldsymbol{\beta}_{\text{Meta}})$ to obtain a well-trained meta robot policy $\pi_{F(\boldsymbol{\beta}_{\text{Meta}})}$;
2. For $i = 1, 2, \ldots, N$, transfer the meta robot policy $\pi_{F(\boldsymbol{\beta}_{\text{Meta}})}$ using HERD from the meta robot $F(\boldsymbol{\beta}_{\text{Meta}})$ to the target robot $F(\boldsymbol{\beta}_i)$ to obtain target robot policy $\pi_{F(\boldsymbol{\beta}_i)}$;

In this way, both simulation for exploration and training during policy transfer can be saved significantly. Theoretically, if the target robots are close enough to the target robots, or the policy transfer from the meta robot to the target robots is easy enough, we could expect our Meta-Evolve a speedup up to $O(N)$ compared to one-to-one policy transfer baselines.

3.4 META ROBOT DETERMINATION

Given the source and target robots, the meta robot significantly impacts the overall performance of the policy transfer. However, due to the complexity of the robots' physical parameter space and its relation to the actual MDP transition dynamics, it is extremely difficult to develop a universal solution. We hereby provide the following three heuristics for determining the meta robot.

1. Geometric Median of Source and Target Robots (denoted as "*Geom-Med All*") We aim to minimize the total L^2 travel distance in robot evolution parameter space from the source robot $F(\boldsymbol{\beta}_0)$ to the meta robot $F(\boldsymbol{\beta}_{\text{Meta}})$ and then to all target robots $\boldsymbol{\beta}_i$. Mathematically, a point that minimizes the sum of distances to a set of points is called the *geometric median* of the point set. Then the meta robot will be the geometric median of evolution parameter set $\{\boldsymbol{\beta}_0, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_N\}$:

$$\boldsymbol{\beta}_{\text{Meta, All}} = \underset{\boldsymbol{\beta} \in [0,1]^D}{\arg\min} \sum_{i=0}^{N} ||\boldsymbol{\beta} - \boldsymbol{\beta}_i||_2 \tag{5}$$

Note that by using geometric median, we assume that the training cost for transferring the policy from robot $F(\boldsymbol{\alpha}_{i,k})$ to robot $F(\boldsymbol{\alpha}_{i,k+1})$ is proportional to $||\boldsymbol{\alpha}_{i,k} - \boldsymbol{\alpha}_{i,k+1}||_2$. We believe this is a reasonable assumption since the training cost should be proportional to the distribution difference $\mathcal{D}_{KL}(F(\boldsymbol{\alpha}_{i,k}), F(\boldsymbol{\alpha}_{i,k+1}))$ of the MDPs of the two robots $F(\boldsymbol{\alpha}_{i,k}), F(\boldsymbol{\alpha}_{i,k+1})$ and should be proportional to the robot hardware difference $||\boldsymbol{\alpha}_{i,k} - \boldsymbol{\alpha}_{i,k+1}||_2$ when $||\boldsymbol{\alpha}_{i,k} - \boldsymbol{\alpha}_{i,k+1}||_2 \to 0, \forall k$.

2. Geometric Median of Target Robots *Only* (denoted as "*Geom-Med Target only*") At each training phase of the policy transfer, the algorithm should aim to reduce the expected **future** cost of training instead of the past. Therefore, at training phase k, the algorithm should act **greedily** to minimize the total L^2 travel distance from the current robot $F(\boldsymbol{\alpha}_{i,k})$ to the meta robot $F(\boldsymbol{\beta}_{\text{Meta}})$ and then to all target robots $F(\boldsymbol{\beta}_i)$, i.e. the temporary meta robot at phase k should be determined as the geometric median of evolution parameter set $\{\boldsymbol{\alpha}_{i,k}, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_N\}$:

$$\boldsymbol{\beta}_{\text{Meta},k} = \arg\min_{\boldsymbol{\beta}}(||\boldsymbol{\beta} - \boldsymbol{\alpha}_{i,k}||_2 + \sum_{i=1}^{N} ||\boldsymbol{\beta} - \boldsymbol{\beta}_i||_2) \tag{6}$$

As long as $\boldsymbol{\alpha}_{i,k}$ moves towards the goal of $\boldsymbol{\beta}_{\text{Meta},k}$ and reduces its distance to $\boldsymbol{\alpha}_{i,k}$, in training phase k, the $\boldsymbol{\beta}_{\text{Meta},k}$ will converge to the geometric median of evolution parameter set $\{\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_N\}$ as shown by Theorem 3.1. For simplicity and to avoid introducing additional noise due to the change of meta robot goal, we directly use the following $\boldsymbol{\beta}_{\text{Meta, Target}}$ as the meta robot in our algorithm.

$$\boldsymbol{\beta}_{\text{Meta, Target}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} ||\boldsymbol{\beta} - \boldsymbol{\beta}_i||_2 \tag{7}$$

Theorem 3.1. *Suppose $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_N$ are not collinear. Suppose $\boldsymbol{\beta}_{Meta,k}$ is defined by Equation (6), and for all k, $||\boldsymbol{\alpha}_{i,k+1} - \boldsymbol{\beta}_{Meta,k}||_2 < ||\boldsymbol{\alpha}_{i,k} - \boldsymbol{\beta}_{Meta,k}||_2$, then $\lim_{k\to\infty} \boldsymbol{\beta}_{Meta,k} = \boldsymbol{\beta}_{Meta,\ Target}$.*

The proof of Theorem 3.1 is in Section A.

3. Minimum Distance to Target Robots' Convex Hull (denoted as "*Min Dist to Cvx Hull*") In general, we cannot have complete knowledge about how the change of a certain parameter affects the MDP transition dynamics of a robot. Therefore, it may be wise to set the meta robot to be outside the ranges of target robot parameters and let the subsequent independently launched HERD runs figure out the optimized evolution path themselves. At the same time, we would prefer the meta robot to be as close to the target robots as possible. Thus, we can choose the point on the surface of the convex hull spanned by the target robots that has the minimum distance to the source robot:

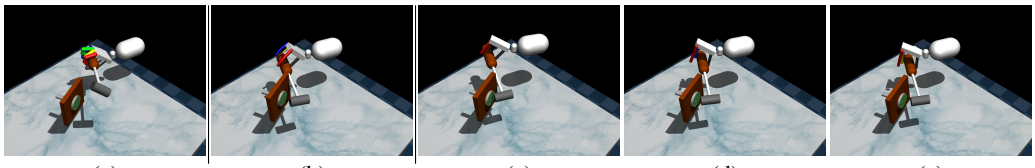
$$\forall d, \boldsymbol{\beta}_{\text{Meta, Min}}(d) = \begin{cases} \max\{\boldsymbol{\beta}_1(d), \boldsymbol{\beta}_2(d) \ldots, \boldsymbol{\beta}_N(d)\} & \text{if } \boldsymbol{\beta}_0(d) > \max\{\boldsymbol{\beta}_1(d), \boldsymbol{\beta}_2(d) \ldots, \boldsymbol{\beta}_N(d)\} \\ \min\{\boldsymbol{\beta}_1(d), \boldsymbol{\beta}_2(d) \ldots, \boldsymbol{\beta}_N(d)\} & \text{if } \boldsymbol{\beta}_0(d) < \min\{\boldsymbol{\beta}_1(d), \boldsymbol{\beta}_2(d) \ldots, \boldsymbol{\beta}_N(d)\} \\ \boldsymbol{\beta}_0(d) & \text{otherwise} \end{cases} \tag{8}$$

4 Related Work

Imitation Learning across Different Robots Traditional imitation learning is designed for learning on the same robots such as Ross et al. (2011); Ng et al. (2000); Ho & Ermon (2016); Duan et al. (2017); Ke et al. (2020). However, due to a huge mismatch in transition dynamics, these works often struggle in learning across different robots. Our work can be viewed as imitation learning across different robots. Compared to previous imitation learning methods that aim to learn across different robots directly (Radosavovic et al., 2020; Liu et al., 2019), we aim to employ robot evolution to gradually adapt the policy. Furthermore, our Meta-Evolve focuses on one-to-many imitation where the policy must work on multiple target robots.

Learning Controllers for Diverse Robot Morphology Recent work has studied the problem of learning a policy/controller for diverse robots. For instance, Pathak et al. (2019) uses dynamic graph neural networks to control and develop robots with different morphology simultaneously to build agents that can generalize to new scenarios. Besides, GNNs have been used to control diverse robot morphology in NerveNet (Wang et al., 2018) to control different robots. Huang et al. (2020) leverages modularity using graph neural networks across limbs of robots to train agent-agnostic policies. Hierarchical controllers have also been shown to be effective while transferring across morphology (Hejna et al., 2020). In contrast to these works, we do not co-develop the controller with morphology but transfer the policy from a source robot to multiple target robots by simulating an evolutionary process.

Meta Learning Our Meta-Evolve is closely related to the formulation of meta-learning (Finn et al., 2017; 2018; Rajeswaran et al., 2019). Different from meta reinforcement learning where only the policies π are meta learned, our formulation can be viewed as the continuous update of both the policy π as well as the transition dynamics $\mathcal{T}$ instantiated by different robot parameters $\boldsymbol{\theta}$. Moreover,



(a) (b) (c) (d) (e)

Figure 3: Visualization of `Hammer` task results. From left to right: (a) source robot; (b) meta robot; (c) 2-finger target robot; (d) 3-finger target robot; and (e) 4-finger target robot.

HERD		2-finger robot	3-finger robot	4-finger robot	total	speedup	
	# of train	10323 ± 1612	6301 ± 1418	6513 ± 1725	23138 ± 4366	1×	
	# of sim	393916 ± 44999	322192 ± 42403	382336 ± 112139	1098444 ± 198839	1×	
Ours (Geom-Med All)		source robot to meta robot	meta robot to 2-finger robot	meta robot to 3-finger robot	meta robot to 4-finger robot	total	speedup
	# of train	4785 ± 317	11080 ± 4935	1120 ± 488	1091 ± 487	18076 ± 4921	1.28×
	# of sim	242874 ± 20982	230217 ± 55268	53106 ± 13043	73224 ± 12145	599421 ± 60332	1.83×
Ours (Geom-Med Target only)		source robot to meta robot	meta robot to 2-finger robot	meta robot to 3-finger robot	meta robot to 4-finger robot	total	speedup
	# of train	5581 ± 647	6007 ± 642	208 ± 223	321 ± 275	12116 ± 594	1.91×
	# of sim	294196 ± 17396	109472 ± 15100	14184 ± 6503	33276 ± 13859	**451128 ± 12363**	**2.43**×
Ours (Min Dist to Cvx Hull)		source robot to meta robot	meta robot to 2-finger robot	meta robot to 3-finger robot	meta robot to 4-finger robot	total	speedup
	# of train	4859 ± 657	3999 ± 897	597 ± 328	235 ± 117	**9691 ± 719**	**2.39**×
	# of sim	304752 ± 15955	165824 ± 30551	58856 ± 11620	32792 ± 7814	562224 ± 64671	1.95

Table 1: `Hammer` policy transfer experiment results. We use "mean ± standard deviation" from runs of five different random seeds. "Geom-Med All", "Geom-Med Target only" and "Min Dist to Cvx Hull" denotes the three approaches presented in Section 3.4 respectively.

while meta-learning aims to learn a meta policy from scratch, in our problem, the source expert policy is provided and used in policy transfer. Closely related to our approach is task interpolation for meta-learning (Yao et al., 2021). Different from task interpolation, our policy transfer does not require working on multiple robots at the same time but only needs each transferred policy to work on one target robot.

5 Experiments

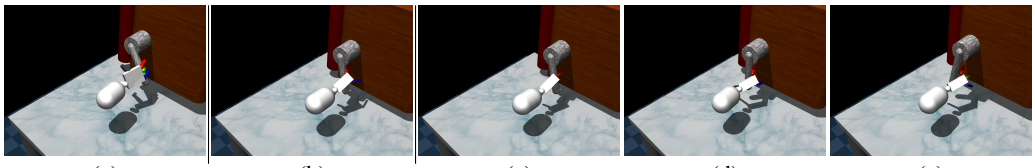
The design of our Meta-Evolve method is motivated by the hypothesis that by sharing the evolution paths among multiple robots through the design of the meta robot, the overall cost of one-to-many policy transfer can be reduced. To show this, we apply our Meta-Evolve on Hand Manipulation Suite (HMS) (Rajeswaran et al., 2018).

5.1 Experiment Settings

Source and Target Robots We utilize the five-finger dexterous hand provided in the ADROIT platform (Kumar et al., 2013) as the source robot and follow Rajeswaran et al. (2018) for the initial environment settings. The target robots are three robot grippers with two, three, and four fingers respectively. The target robots can be produced by gradually shrinking the fingers of the five-finger dexterous hand. The robot evolution is illustrated in Figures 1 and 2(b).

Task, Reward and RL Algorithm We use the three tasks from the the task suite in (Rajeswaran et al., 2018): `Hammer`, `Relocate` and `Door`. In `Hammer`, the task is to pick up the hammer and smash the nail into the board; in `Relocate`, the task is to pick up the ball and take it to the target position; in `Door`, the task is to turn the door handle and fully open the door. We use a challenging sparse reward function where only the task completion is rewarded. We use NPG (Rajeswaran et al., 2017) as the RL algorithm in all compared methods.

Evaluation Metrics For each compared method, the goal is to reach 80% success rate on all three target robots. Due to the nature of one-to-many policy transfer, the total number of RL iterations or simulation epochs it takes to reach this goal cannot be set beforehand. So we instead report the number of policy training iterations and simulation epochs needed to reach the goal.



(a) (b) (c) (d) (e)

Figure 4: Visualization of `Door` task results. From left to right: (a) source robot; (b) meta robot; (c) 2-finger target robot; (d) 3-finger target robot; and (e) 4-finger target robot.

		2-finger robot	3-finger robot	4-finger robot	total	speedup
HERD	# of train	2015 ± 478	1016 ± 75	1325 ± 170	4357 ± 394	1×
	# of sim	201184 ± 18338	188236 ± 6953	240284 ± 7980	629704 ± 5942	1×
Ours (Geom-Med All)		source robot to meta robot	meta robot to 2-finger robot	meta robot to 3-finger robot	meta robot to 4-finger robot	total	speedup
	# of train	1461 ± 336	833 ± 206	44 ± 31	127 ± 92	2464 ± 339	1.77
	# of sim	180219 ± 19489	104505 ± 22234	19635 ± 6288	31839 ± 10761	336198 ± 35192	1.87
Ours (Geom-Med Target only)		source robot to meta robot	meta robot to 2-finger robot	meta robot to 3-finger robot	meta robot to 4-finger robot	total	speedup
	# of train	1393 ± 250	540 ± 227	2 ± 3	28 ± 9	**1963 ± 490**	**2.22**×
	# of sim	185604 ± 15076	94684 ± 4543	10100 ± 5002	31284 ± 7972	**321672 ± 32593**	**1.96**×
Ours (Min Dist to Cvx Hull)		source robot to meta robot	meta robot to 2-finger robot	meta robot to 3-finger robot	meta robot to 4-finger robot	total	speedup
	# of train	1527 ± 238	782 ± 226	3 ± 4	8 ± 9	2319 ± 250	1.88
	# of sim	214875 ± 9954	114705 ± 27323	28524 ± 1946	14109 ± 905	372213 ± 22544	1.69

Table 2: `Door` policy transfer experiment results. We use "mean ± standard deviation" from runs of five different random seeds. "Geom-Med All", "Geom-Med Target only" and "Min Dist to Cvx Hull" denotes the three approaches presented in Section 3.4 respectively.

5.2 One-to-three Robot-to-robot Policy Transfer

We compare our Meta-Evolve against one-to-one policy transfer baseline HERD (Liu et al., 2022a). The results on `Hammer`, `Door`, `Relocate` are illustrated in Tables 1, 2 and 3 respectively. In terms of the total number of simulation epochs and training iterations needed, our method is able to achieve up to 2.4× improvement on these three tasks.

Breaking down each part of the evolution path, we noticed that in our method, the source robot to meta robot part of the path is usually the most costly and constitutes the largest portion of simulation epochs as well as training, e.g. on `Relocate` and `Door` tasks. However, the cost after splitting the path at the meta robot is much smaller which yields a smaller total cost.

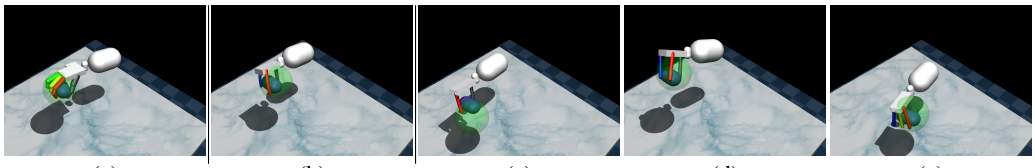
A more interesting observation is that, for some target robots and tasks, e.g. two- and four-finger target robots on `Hammer` task, the total cost of path "source to meta robot" plus the cost of path " meta robot to target robot" is even smaller than the cost of directly transferring the policy to that target robot using HERD. It shows that, transferring the policies to multiple related target through a meta robot can possibly help each robot improve their learning efficiency. We believe this phenomena deserve more future research attention.

5.3 Ablation Studies

We provide ablation studies on the design choice of meta robot. The results are illustrated in Tables 1, 2 and 3. Choosing the meta robot as the minimum distance point to the target robot convex hull (i.e. "Min Dist to Cvx Hull") and geometric median of only the target robots (i.e. "Geom-Med Target only") yield similar performance and are much better than choosing the geometric median of both source and target robots (i.e. "Geom-Med All"). After further analysis, a possible reason is that the first two approaches produce meta robots that are much closer to the targets than the last approach. It shows that the meta robot should be selected to be closer to the target robot if possible.

5.4 Visualizations

We provide visualizations of the transferred policies on each of the three target robots as well as the meta robot, on all three HMS tasks. The policy is able to be transferred to all target robots and successfully complete the task. Moreover, we observe that the meta robot obtained from the heuristics we introduced in Section 3.4 looks like a deformed three-finger gripper which is indeed



(a) (b) (c) (d) (e)

Figure 5: Visualization of `Relocate` task results. From left to right: (a) source robot; (b) meta robot; (c) 2-finger target robot; (d) 3-finger target robot; and (e) 4-finger target robot.

HERD		2-finger robot	3-finger robot	4-finger robot	total	speedup	
	# of train	7602 ± 1158	9656 ± 2228	9849 ± 932	27109 ± 4208	1×	
	# of sim	316600 ± 31284	363332 ± 23999	369868 ± 1836	1049800 ± 56510	1×	
Ours (Geom-Med All)		source robot to meta robot	meta robot to 2-finger robot	meta robot to 3-finger robot	meta robot to 4-finger robot	total	speedup
	# of train	6245 ± 1138	3458 ± 1106	4194 ± 1202	3404 ± 918	17301 ± 1702	1.57×
	# of sim	268818 ± 25115	117447 ± 14632	109356 ± 20365	107223 ± 24330	602844 ± 53470	1.74×
Ours (Geom-Med Target only)		source robot to meta robot	meta robot to 2-finger robot	meta robot to 3-finger robot	meta robot to 4-finger robot	total	speedup
	# of train	10744 ± 2402	1286 ± 1065	438 ± 327	316 ± 355	12784 ± 2399	2.12×
	# of sim	364254 ± 36482	66891 ± 22691	20280 ± 8499	30594 ± 9877	**482019 ± 52554**	**2.18**×
Ours (Min Dist to Cvx Hull)		source robot to meta robot	meta robot to 2-finger robot	meta robot to 3-finger robot	meta robot to 4-finger robot	total	speedup
	# of train	10031 ± 1857	1493 ± 462	494 ± 368	298 ± 502	**12316 ± 2569**	**2.20**×
	# of sim	357951 ± 30773	77382 ± 29341	46101 ± 12497	23454 ± 11721	504888 ± 27059	2.08×

Table 3: `Relocate` policy transfer experiment results. We use "mean ± standard deviation" from runs of five different random seeds. "Geom-Med All", "Geom-Med Target only" and "Min Dist to Cvx Hull" denotes the three approaches presented in Section 3.4 respectively.

close to all three target robots, considering the target robots include two-, three- and four-finger grippers respectively. For more details on the visualization, please refer to the supplementary video.

6 Discussions

When does Meta-Evolve Fail? One core assumption behind our Meta-Evolve method is that the target robots are mutually similar while very different from the source robot. If the evolution goal is in opposite directions, our Meta-Evolve will not work. For example, given the source robot of a five-finger hand, if one target robot is a ten-finger hand while another target robot is a two-finger gripper, the optimal meta robot might simply be the source robot itself, because the evolution directions toward the target robots are opposite (growing multiple fingers vs. shrinking multiple fingers). However, in practice, most industrial robots are indeed mutually similar in morphology and kinematics while very different from the human hand, so our assumption still stands and Meta-Evolve can still be useful in such cases.

Can the Meta Robot be Learned or Optimized? We envision the learning or optimization of the meta robot being very challenging. Policy transfer through robot evolution relies on local optimization in the robot evolution space. On the other hand, the optimization of the meta robot requires optimizing the robot evolution path globally and needs an accurate "guess" of the results of future policy transfer. We leave the problem of finding an optimized meta robot as the future work.

7 Conclusion

In this paper, we introduce a new research problem of transferring an expert policy from a source robot to multiple target robots. To solve this new problem, we introduce a new method name Meta-Evolve that utilizes continuous robot evolution to efficiently transfer the policy to a meta robot and then to each target robot. We also present three different heuristic approaches with theoretical results to determine the meta robot. We conduct experiments on Hand Manipulation Suite tasks and show that our one-to-many policy transfer method can significantly outperform the one-to-one policy transfer baseline.

REFERENCES

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 720–736, 2018.

Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18995–19012, 2022.

Donald Hejna, Lerrel Pinto, and Pieter Abbeel. Hierarchically decoupled imitation for morphological transfer. In *International Conference on Machine Learning*, pp. 4159–4171. PMLR, 2020.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29:4565–4573, 2016.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020.

Liyiming Ke, Sanjiban Choudhury, Matt Barnes, Wen Sun, Gilwoo Lee, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. In *International Workshop on the Algorithmic Foundations of Robotics*, pp. 313–329. Springer, 2020.

Vikash Kumar, Zhe Xu, and Emanuel Todorov. Fast, strong and compliant pneumatic actuation for dexterous tendon-driven hands. In *2013 IEEE international conference on robotics and automation*, pp. 1512–1519. IEEE, 2013.

Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. *arXiv preprint arXiv:1911.10947*, 2019.

Xingyu Liu, Deepak Pathak, and Kris M. Kitani. HERD: Continuous Human-to-Robot Evolution for Learning from Human Demonstration. In *The Conference on Robot Learning (CoRL)*, 2022a.

Xingyu Liu, Deepak Pathak, and Kris M. Kitani. REvolveR: Continuous Evolutionary Models for Robot-to-robot Policy Transfer. In *The International Conference on Machine Learning (ICML)*, 2022b.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Deepak Pathak, Chris Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: a study of generalization via modularity. *NeurIPS*, 2019.

Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. State-only imitation learning for dexterous manipulation. *arXiv preprint arXiv:2004.04650*, 2020.

Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham Kakade. Towards generalization and simplicity in continuous control. *arXiv preprint arXiv:1703.02660*, 2017.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.049.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

ShadowRobot. Shadowrobot dexterous hand, 2005. URL https://www.shadowrobot.com/dexterous-hand-series/.

Yehuda Vardi and Cun-Hui Zhang. The multivariate l 1-median and associated data depth. *Proceedings of the National Academy of Sciences*, 97(4):1423–1426, 2000.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. *ICLR*, 2018.

Huaxiu Yao, Linjun Zhang, and Chelsea Finn. Meta-learning with fewer tasks through task interpolation. In *International Conference on Learning Representations*, 2021.

A PROOF OF THEOREM 3.1

Proof. For all k, the definition of $\boldsymbol{\beta}_{\text{Meta},k}$ is

$$\boldsymbol{\beta}_{\text{Meta},k} = \arg\min_{\boldsymbol{\beta}}(||\boldsymbol{\beta} - \boldsymbol{\alpha}_{i,k}||_2 + \sum_{i=1}^{N} ||\boldsymbol{\beta} - \boldsymbol{\beta}_i||_2) \tag{9}$$

Thus we have

$$\begin{aligned} &||\boldsymbol{\beta}_{\text{Meta},k+1} - \boldsymbol{\alpha}_{i,k+1}||_2 + \sum_{i=1}^{N} ||\boldsymbol{\beta}_{\text{Meta},k+1} - \boldsymbol{\beta}_i||_2 \\ &\leq ||\boldsymbol{\beta}_{\text{Meta},k} - \boldsymbol{\alpha}_{i,k+1}||_2 + \sum_{i=1}^{N} ||\boldsymbol{\beta}_{\text{Meta},k} - \boldsymbol{\beta}_i||_2 \end{aligned} \tag{10}$$

Since for all k, $||\boldsymbol{\alpha}_{i,k+1} - \boldsymbol{\beta}_{\text{Meta},k}||_2 < ||\boldsymbol{\alpha}_{i,k} - \boldsymbol{\beta}_{\text{Meta},k}||_2$, therefore

$$\begin{aligned} &||\boldsymbol{\beta}_{\text{Meta},k+1} - \boldsymbol{\alpha}_{i,k+1}||_2 + \sum_{i=1}^{N} ||\boldsymbol{\beta}_{\text{Meta},k+1} - \boldsymbol{\beta}_i||_2 \\ &< ||\boldsymbol{\beta}_{\text{Meta},k} - \boldsymbol{\alpha}_{i,k}||_2 + \sum_{i=1}^{N} ||\boldsymbol{\beta}_{\text{Meta},k} - \boldsymbol{\beta}_i||_2 \end{aligned} \tag{11}$$

Let $d_k = ||\boldsymbol{\beta}_{\text{Meta},k} - \boldsymbol{\alpha}_{i,k}||_2 + \sum_{i=1}^{N} ||\boldsymbol{\beta}_{\text{Meta},k} - \boldsymbol{\beta}_i||_2$, Equation (11) can be re-written as

$$d_{k+1} < d_k \tag{12}$$

which means the sequence $\{d_k\}$ is a decreasing sequence. By definition, d_k is a summation of distance, so $\forall k, d_k > 0$. Therefore, according to the Monotone Convergence Theorem, the sequence $\{d_k\}$ must converge to its lower bound $\inf\{d_k\}$.

Since $\forall \boldsymbol{\beta}$,

$$\begin{aligned} &||\boldsymbol{\beta} - \boldsymbol{\alpha}_{i,k}||_2 + \sum_{i=1}^{N} ||\boldsymbol{\beta} - \boldsymbol{\beta}_i||_2 \\ &\geq \sum_{i=1}^{N} ||\boldsymbol{\beta} - \boldsymbol{\beta}_i||_2 \\ &\geq \sum_{i=1}^{N} ||\boldsymbol{\beta}_{\text{Meta, Target}} - \boldsymbol{\beta}_i||_2 \end{aligned} \tag{13}$$

The last inequality is due to the definition of $\boldsymbol{\beta}_{\text{Meta, Target}}$.

Since there exists a solution of $\boldsymbol{\alpha}_{i,k} = \boldsymbol{\beta}_{\text{Meta, Target}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} ||\boldsymbol{\beta} - \boldsymbol{\beta}_i||_2$ that can turn all two inequalities in Equation (13) into equality. Also considering that $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_N$ are not collinear, according to Vardi & Zhang (2000), the solution to the geometric median $\boldsymbol{\beta}_{\text{Meta, Target}}$ is unique given $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_N$.

This means the lower bound $\inf\{d_k\}$ is achieved if and only at $\boldsymbol{\beta}_{\text{Meta, Target}} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} ||\boldsymbol{\beta} - \boldsymbol{\beta}_i||_2$ which also means

$$\lim_{k \to \infty} \boldsymbol{\beta}_{\text{Meta},k} = \boldsymbol{\beta}_{\text{Meta, Target}} \tag{14}$$

□

B ROBOT EVOLUTION SPECIFICS

We illustrate the kinematic tree of the source robot in Figure 6. During evolution, all revolute joints gradually freeze to have a range of 0. On the other hand, the prismatic joints are initially frozen with a range of 0, and some of their ranges gradually increase until the same full range. During robot

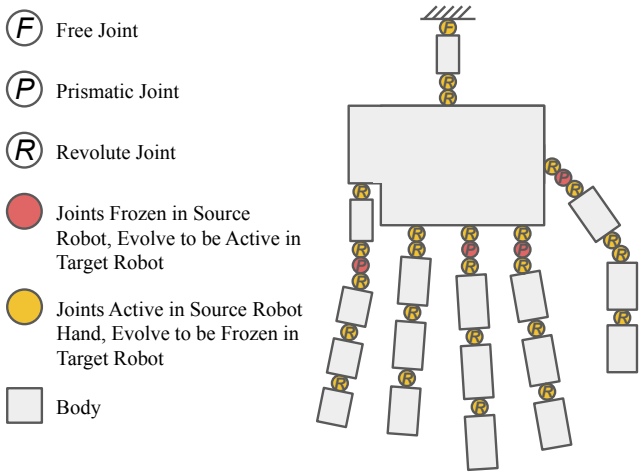


Figure 6: Kinematic tree of dexterous hand robot. All revolute and free joints will gradually freeze during evolution. The two prismatic joints are initially frozen and evolve to be active.

Hyperparameter	Value
RL Discount Factor γ	0.995
GAE	0.97
NPG Step Size	0.0001
Policy Network Hidden Layer Sizes	(32,32)
Value Network Hidden Layer Sizes	(32,32)
Simulation Epoch Trajectory Length	200
RL Traning Batch Size	12
Evolution Progression Step Size ξ	0.06
Number of Sampled $\boldsymbol{\delta}_i$ for Jacobian Estimation n	72
Evolution Direction Weight Factor λ	1.0
Sample Range Shrink Ratio λ_1	0.995
Success Rate Threshold q	0.667

Table 4: The value of hyperparametrs used in our experiments.

evolution, the body of the ring finger gradually shrinks to be zero-size and disappears. Our evolution solution includes the changing of $D = 65$ independent robot parameters.

C Hyperparameter and Training Details

We present the hyperparameters and training procedures of our robot evolution and policy optimization. We use PyTorch (Paszke et al., 2019) as our deep learning framework and NPG (Rajeswaran et al., 2017) as the RL algorithm in all experiments. To fairly compare against HERD (Liu et al., 2022a), we use the same evolution progression step size ξ as HERD in the experiments. The hyperparameters are illustrated in Table 4.