Robust Contact-rich Manipulation through Implicit Motor Adaptation

Journal Title
XX(X):1–17

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Teng Xue[1, 2], Amirreza Razmjoo[1, 2], Suhan Shetty[1, 2] and Sylvain Calinon [1, 2]

Abstract

Contact-rich manipulation plays an important role in daily human activities. However, uncertain physical parameters often pose significant challenges for both planning and control. A promising strategy is to develop policies that are robust across a wide range of parameters. Domain adaptation and domain randomization are widely used, but they tend to either limit generalization to new instances or perform conservatively due to neglecting instance-specific information. *Explicit motor adaptation* addresses these issues by estimating system parameters online and then retrieving the parameter-conditioned policy from a parameter-augmented base policy. However, it typically requires precise system identification or additional training of a student policy, both of which are challenging in contact-rich manipulation tasks with diverse physical parameters. In this work, we propose *implicit motor adaptation*, which enables parameter-conditioned policy retrieval given a roughly estimated parameter distribution instead of a single estimate. We leverage tensor train as an implicit representation of the base policy, facilitating efficient retrieval of the parameter-conditioned policy by exploiting the separable structure of tensor cores. This framework eliminates the need for precise system estimation and policy retraining while preserving optimal behavior and strong generalization. We provide a theoretical analysis to validate the approach, supported by numerical evaluations on three contact-rich manipulation primitives. Both simulation and real-world experiments demonstrate its ability to generate robust policies across diverse instances. Project website: https://sites.google.com/view/implicit-ma.

Keywords

Contact-rich manipulation, robust control, sim-to-real transfer, tensor train decomposition

1 Introduction

Robot manipulation usually involves multiple contact-rich manipulation primitives, such as `Push` and `Pivot`, leading to a long-horizon problem characterized by hybrid types of variables (discrete and continuous). The resulting combinatorial complexity poses significant challenges to most planning and control approaches. Instead of treating long-horizon manipulation as a whole, one way is to decompose the long-horizon process into several simple contact-rich manipulation primitives and then sequence them using PDDL planners (Ghallab et al. 1998; Xue et al. 2024b; Cheng and Xu 2023) or Large Language Models (Driess et al. 2023). Since manipulation primitives are typically sequenced by naive symbolic planners that lack geometric or motion-level information, it is crucial to develop primitives that are robust to diverse instances with varying physical parameters, such as shape, mass, and friction coefficient. For example, once a `push` primitive is scheduled by a high-level symbolic planner, it should be capable of pushing objects with different physical properties from any initial configuration to their targets.

Many approaches have been proposed for robust planning and control through contacts, such as H-∞ control (Franco et al. 2006), Sliding Mode Control (SMC) (Shtessel et al. 2014; Edwards and Spurgeon 1998) or Model Predictive Control (MPC) (Morari and Lee 1999; Garcia et al. 1989; Lopez et al. 2019). However, these methods often overlook instance-specific information and focus primarily on worst-case scenarios, resulting in conservative behaviors. To address this limitation, online system identification can be integrated to enable adaptive MPC (Adetola et al. 2009; Adetola and Guay 2011). However, the diverse parameters and non-smooth contact dynamics result in computationally expensive parameter estimation and trajectory optimization, leading to poor performance for real-time control.

Learning manipulation policies that can quickly react to the physical world is thus required to advance further. Widely used approaches include reinforcement learning (RL) (Kaelbling et al. 1996; Kober et al. 2013) and imitation learning (IL) (Billard et al. 2008; Schaal 1999). Due to the high cost of data collection in real-world contact-rich scenarios, policy learning in simulation and transferring it to the real world has become a common strategy. To enable robust sim-to-real transfer, commonly adopted methods include domain randomization (DR) (Tobin et al. 2017; Muratore et al. 2018) and domain adaptation (DA) (Bousmalis et al. 2018; Arndt et al. 2020; Chebotar et al.

[1] Idiap Research Institute, Martigny, Switzerland
[2] École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

Corresponding author:
Teng Xue, Idiap Research Institute, Rue Marconi 19, 1920 Martigny, Switzerland.
Email: teng.xue@idiap.ch

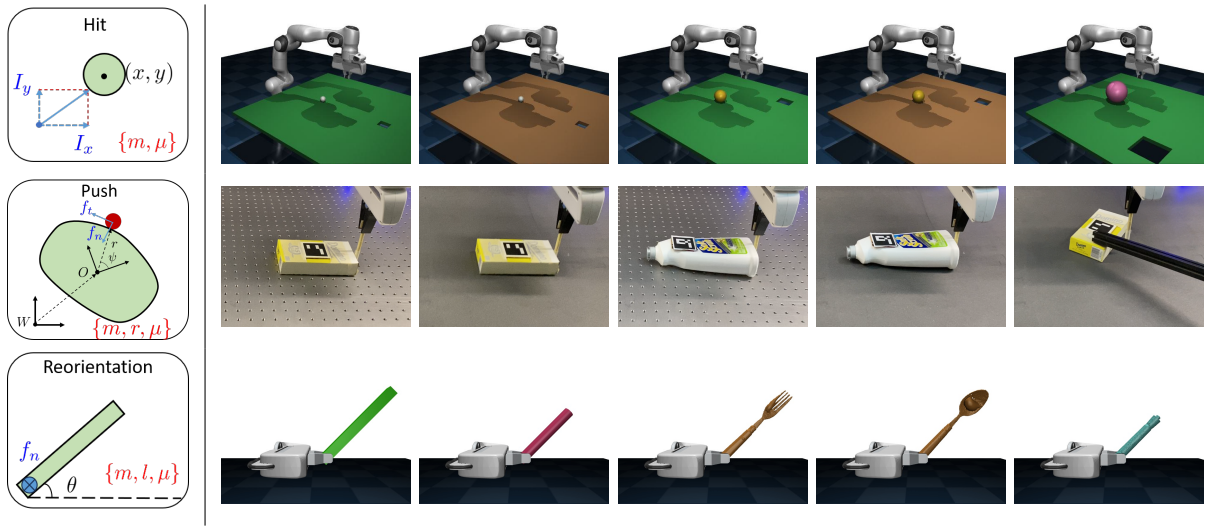


Figure 1. Deployment of the learned policy in a variety of contact-rich manipulation tasks. The left images illustrate the state and action spaces for each primitive (`Hit`, `Push`, `Reorientation`) in black and blue, respectively, along with the parameter spaces represented by the red variables: m for mass, μ for the friction coefficient, r for the radius, and l for the length. A single policy is trained for each primitive and deployed directly across a wide range of objects with varying shapes, weights, and friction parameters, while preserving instance-specific optimal behaviors.

2019). DR assumes that environment parameters follow specific distributions (e.g., normal or uniform), and the goal is to maximize (or minimize) the expected reward (or cost) under randomly sampled parameters during training. While this method provides good generalization to diverse instances, it often results in suboptimal and high-variance behaviors due to restrictive assumptions about the parameter distributions. In contrast, DA incorporates more instance-specific data during policy training, leading to optimal behaviors in the specific target domain but sacrificing generalization to other instances without additional fine-tuning.

Combining DR and DA to harness the benefits of both is a sensible strategy (Muratore et al. 2022; Qi et al. 2023), but how to effectively balance them is still an open question. It typically involves online system identification to capture domain-specific information and multi-domain learning to account for diverse instances. One promising approach to address this trade-off is to learn a base policy that uses diverse privileged environmental information as input and then relies on proprioceptive history to estimate the system parameters or a latent low-dimensional embedding online. For simplicity, we refer to both as the system parameters in this article, which are then used as input to the base policy to generate domain-specific optimal behaviors. This framework, referred to as Teacher-Student Policy in (Lee et al. 2020a) and Rapid Motor Adaptation in (Kumar et al. 2021) and (Qi et al. 2023), is more broadly termed *explicit motor adaptation (EMA)* in this article, as the policies are typically represented as explicit feedforward functions that directly output actions given observations and estimated parameters. It can be viewed as a variant of DA that enhances generalization by including online parameter estimation.

This approach has demonstrated impressive performance in quadrupedal locomotion across diverse terrains (Lee et al. 2020a; Kumar et al. 2021) and robust in-hand manipulation (Qi et al. 2023). However, it typically relies on high-quality system identification (Qi et al. 2023) or additional training of a student policy to mimic the teacher policy (Lee et al. 2020a). These requirements pose significant challenges in contact-rich manipulation tasks, as they demand extensive data collection and considerable effort in model design and training, but remain difficult to achieve.

Instead, we propose *implicit motor adaptation (IMA)* in this work. IMA can be seen as an elaborated version of DR, leveraging instance-aware distributions instead of being parameter-blind, while also retaining the advantage of DR in not relying heavily on good-quality system identification. It addresses sim-to-real transfer as a stochastic problem, pushing the limits for finding an optimal policy under sim-to-real uncertainty without requiring policy retraining for new instances. The policy is implicitly represented as $\arg\max$ of the parameter-conditioned advantage function

$$\boldsymbol{u}^* = \arg\max_{\boldsymbol{u}\in\mathcal{U}} A(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{u}) \quad \text{instead of} \quad \boldsymbol{u}^* = \pi_\theta(\boldsymbol{h}, \boldsymbol{x}), \tag{1}$$

where $\boldsymbol{h}$ is the proprioceptive history. EMA estimates the system parameter from $\boldsymbol{h}$ and directly inputs it into the base policy π_θ to output the action. In contrast, thanks to the implicit policy representation, IMA can retrieve the parameter-conditioned policy using a probabilistic estimate based on $\boldsymbol{h}$. This makes it particularly effective for contact-rich manipulation tasks, in which contact parameters identification is often challenging. Although implicit representations have been explored in imitation learning (Florence et al. 2022) and as components of reinforcement learning (Haarnoja et al. 2017; Wang et al. 2022), to the best of our knowledge, our work is the first to investigate this idea in the context of robust policy learning.

However, computing the parameter-conditioned advantage function and retrieving the policy can be computationally expensive, making it difficult to apply in real-world settings, particularly in contact-rich manipulation tasks where the robot has to continuously interact with its surroundings. To address this issue, we employ Tensor Train (TT) (Oseledets

2011) as function representation, enabling fast computation of the advantage function and efficient online policy retrieval.

This article builds on our previous work (Xue et al. 2024a), where we introduced domain contraction, a mechanism for retrieving policies with a known probabilistic distribution. Here, we extend this concept by integrating domain contraction into a comprehensive sim-to-real transfer framework, namely *implicit motor adaptation*. The framework incorporates an online system identification module that probabilistically estimates environmental parameters from proprioceptive history and retrieves the parameter-conditioned policy for real-world manipulation from a base policy trained in simulation.

In summary, beyond employing Tensor Train as an implicit policy representation and utilizing domain contraction for probabilistic policy retrieval, this paper introduces the following new contributions compared to (Xue et al. 2024a):

1) A novel framework called *implicit motor adaptation (IMA)* for robust contact-rich manipulation, highlighting the promise of implicit representation for robust policy learning compared with the widely used *explicit motor adaptation (EMA)* framework.

2) A probabilistic system adaptation module allowing the robot to adapt its behavior in a probabilistic manner without relying on high-quality system identification or additional student policy training.

3) Theoretical proofs and numerical experiments demonstrating the effectiveness of IMA compared to EMA.

2 Related Work

2.1 Learning for Contact-rich Manipulation

Many approaches have been proposed to learn manipulation policies for contact-rich manipulation, including Behavior Cloning (BC) (Florence et al. 2022; Torabi et al. 2018), Deep Reinforcement Learning (DRL) (Pertsch et al. 2021; Qi et al. 2023; Lee et al. 2020a), and Approximate Dynamic Programming (ADP) (Powell 2007; Werbos 1992). In our work, we employ an ADP approach called Tensor Train Policy Iteration (TTPI) (Shetty et al. 2024b) for policy learning, where the state space is augmented with system parameters to enable subsequent parameter-conditioned policy retrieval given different instances. However, our proposed approach is flexible and can be integrated with any policy learning technique, provided the policy can be implicitly represented by advantage functions or energy-based functions.

2.2 Implicit Policy Representation

Instead of being expressed as an explicit function, the action policy can be described implicitly through function optimization. Implicit representations have been widely used in various approaches, including energy-based models (EBMs) (LeCun et al. 2006; Du and Mordatch 2019), diffusion models (Yang et al. 2023; Ho et al. 2020), and tensor networks (Orús 2019; Stoudenmire and Schwab 2016). In imitation learning, behavior cloning can be formulated using EBMs (Florence et al. 2022), highlighting several advantages such as multimodal representation and long-horizon visuomotor policy learning. In methods based on dynamic programming, the control policy is naturally obtained by optimizing objective functions, making it more intuitive to represent the policy implicitly. For example, reactive policy learning can be reformulated as solving sequence-of-constraints model predictive control (MPC) (Toussaint et al. 2022), demonstrating strong generalization and improved compositionality using implicit functions. Similarly, implicit models can be utilized as policy representations in reinforcement learning, emphasizing their expressiveness in action generation (Haarnoja et al. 2017; Wang et al. 2022). For high-dimensional action spaces, Tensor Train (TT) can be emplyed as a low-rank representation for state-value and advantage functions, enabling implicit retrieval of control actions (Shetty et al. 2024b; Tal et al. 2018). Our work is inspired by the performance of implicit representation in control policy learning and aims to explore its potential for robust policy learning, which has not been investigated so far.

2.3 Sim-to-real transfer

Obtaining large amounts of real-world data for primitive learning is challenging. Therefore, robot learning in simulation and transferring to the real world is a promising idea (Peng et al. 2018). However, the reality gap between simulation and the real world poses a significant challenge to such idea. If the target domain is known and specific, Domain Adaptation (DA) (Bousmalis et al. 2018; Arndt et al. 2020; Chebotar et al. 2019) can be an effective method, but its reliance on domain-specific data limits the generalization to other scenarios. Alternatively, Domain Randomization (DR) (Tobin et al. 2017) seeks to develop a robust policy by introducing random variations into the simulation parameters. While this method offers good generalization, it often results in suboptimal and high-variance behaviors due to the restrictive assumptions about environment parameter distribution (e.g., normal or uniform).

Combining DA and DR can be a promising strategy to balance generalization and optimal performance. The basic idea is to adapt the parameter distribution by leveraging differences between simulated and reference environment (Mehta et al. 2020; Muratore et al. 2022; Ramos et al. 2019). However, policies developed through such methods are often tailored to the reference environment or target domain. Recently, explicit motor adaptation (EMA) (Yu et al. 2017) has demonstrated remarkable success in learning robust locomotion (Lee et al. 2020a; Kumar et al. 2021) and manipulation primitives (Qi et al. 2023). In this approach, a teacher policy is trained in simulation with various domain parameters, and a student policy learns to replicate this behavior, with a low-dimensional embedding of proprioceptive history as input. Our work closely follows this paradigm but introduces a more efficient way for learning the parameter-augmented teacher policy using tensor approximation. Additionally, our method eliminates the need for high-quality system identification and subsequent student policy training, as the parameter-conditioned robust policy can be directly derived from the teacher policy given a rough parameter distribution.

2.4 Tensor Train for function approximation

A multidimensional function can be approximated by a tensor, where each element in the tensor is the value of the function given the discretized inputs. The continuous value of the function can then be obtained by interpolating among tensor elements. However, storing the full tensor for a high-dimensional function can be challenging. To address this issue, Tensor Train (TT) was proposed to approximate the tensor using several third-order tensor cores. The widely used methods include TT-SVD (Oseledets 2011) and TT-cross (Oseledets and Tyrtyshnikov 2010). Furthermore, TTGO (Shetty et al. 2024a) was proposed for finding globally optimal solutions given functions in TT format. TTPI (Shetty et al. 2024b) was then introduced to learn control policies through tensor approximation, showing superior performance on several hybrid control problems. Logic-Skill Programming (LSP) (Xue et al. 2024b) expands the operational space of TTPI by incorporating first-order logic to sequence policies. Building on TTGO and TTPI, our work extends these methods to learn robust policies, which further enhances the capabilities of LSP. We additionally demonstrate that the TT format is a suitable structure for efficient parameter-conditioned policy retrieval.

3 Background

3.1 Tensors as Discrete Analogue of a Function

A multivariate function $F(x_1, \dots, x_d)$ defined over a rectangular domain made up of the Cartesian product of intervals (or discrete sets) $I_1 \times \cdots \times I_d$ can be discretized by evaluating it at points in the set $\mathcal{X} = \{(x_1^{i_1}, \dots, x_d^{i_d}) : x_k^{i_k} \in I_k, i_k \in \{1, \dots, n_k\}\}$. This gives us a tensor $\boldsymbol{\mathcal{F}}$, a discrete version of F, where $\boldsymbol{\mathcal{F}}_{(i_1,\dots,i_d)} = F(x_1^{i_1}, \dots, x_d^{i_d}), \forall (i_1, \dots, i_d) \in \mathcal{I}_{\mathcal{X}}$, and $\mathcal{I}_{\mathcal{X}} = \{(i_1, \dots, i_d) : i_k \in \{1, \dots, n_k\}, k \in \{1, \dots, d\}\}$. The value of F at any point in the domain can then be approximated by interpolating between the elements of the tensor $\boldsymbol{\mathcal{F}}$.

3.2 Tensor Networks and Tensor Train Decomposition

Naively approximating a high-dimensional function using a tensor is intractable due to the combinatorial and storage complexities of the tensor ($\mathcal{O}(n^d)$). Tensor networks mitigate the storage issue by decomposing the tensor into factors with fewer elements, akin to using Singular Value Decomposition (SVD) to represent a large matrix. In this paper, we explore the use of Tensor Train (TT), a type of Tensor Network that represents a high-dimensional tensor using several third-order tensors called *cores*, as shown in Fig. 2.

We can access the element indexed $(i_1, \dots, i_d)$ of the tensor in this format simply by multiplying matrix slices from the cores:

$$\boldsymbol{\mathcal{F}}_{(i_1,\dots,i_d)} = \boldsymbol{\mathcal{F}}^1_{:,i_1,:} \boldsymbol{\mathcal{F}}^2_{:,i_2,:} \cdots \boldsymbol{\mathcal{F}}^d_{:,i_d,:}, \tag{2}$$

where $\boldsymbol{\mathcal{F}}^k_{:,i_k,:} \in \mathbb{R}^{r_{k-1} \times r_k}$ represents the i_k-th frontal slice (a matrix) of the third-order tensor $\boldsymbol{\mathcal{F}}^k$. For any given tensor, there always exists a TT decomposition (Oseledets 2011). This low-rank structure further facilitates sampling and optimization for robot planning and control.

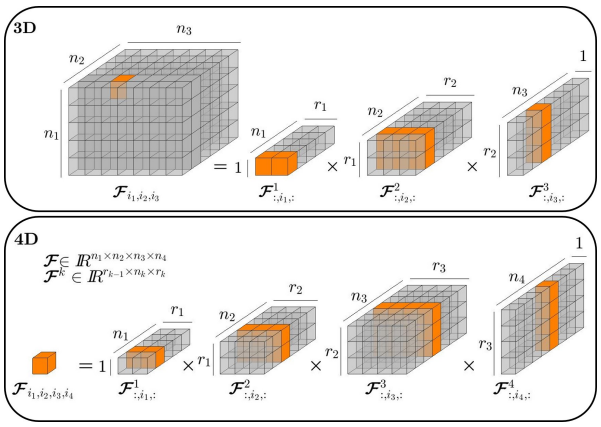


Figure 2. TT decomposition generalizes matrix decomposition techniques to higher-dimensional arrays. In TT format, an element in a tensor can be obtained by multiplying specific slices of the core tensors. The figure presents examples of third-order, and fourth-order tensors. Image adapted from Shetty et al. (2024a).

There are several ways to acquire a TT model, including TT-SVD (Oseledets 2011) and TT-Cross (Oseledets and Tyrtyshnikov 2010; Savostyanov and Oseledets 2011). TT-SVD extends the SVD decomposition from matrix level to a high-dimensional tensor level. However, it needs to store the full tensor first, which is impractical to very high-dimensional functions. TT-Cross solves this limitation by selectively evaluating function F on a subset of elements, avoiding the need to store the entire tensor.

3.3 Function approximation using Tensor Train

Given the discrete analogue tensor $\boldsymbol{\mathcal{F}}$ of a function F, we obtain the continuous approximation by spline-based interpolation of the TT cores corresponding to the continuous variables. For example, we can use linear interpolation for the cores (i.e., between the matrix slices of the core) and define a matrix-valued function corresponding to each core $k \in \{1, \dots, d\}$,

$$\boldsymbol{F}^k(x_k) = \frac{x_k - x_k^{i_k}}{x_k^{i_k+1} - x_k^{i_k}} \boldsymbol{\mathcal{F}}^k_{:,i_k+1,:} + \frac{x_k^{i_k+1} - x_k}{x_k^{i_k+1} - x_k^{i_k}} \boldsymbol{\mathcal{F}}^k_{:,i_k,:}, \tag{3}$$

where $x_k^{i_k} \le x_k \le x_k^{i_k+1}$ and $\boldsymbol{F}^k : I_k \subset \mathbb{R} \to \mathbb{R}^{r_{k-1} \times r_k}$ with $r_0 = r_d = 1$. This induces a continuous approximation of F given by

$$F(x_1, \dots, x_d) \approx \boldsymbol{F}^1(x_1) \cdots \boldsymbol{F}^d(x_d). \tag{4}$$

This allows us to selectively do the interpolation only for the cores corresponding to continuous variables, and hence we can represent functions in TT format whose variables could be a mix of continuous and discrete elements.

3.4 Global Optimization using Tensor Train (TTGO)

In optimization problems, the goal is to find the decision variables $\boldsymbol{x}$ that maximize the objective function $f(\boldsymbol{x})$.

TTGO (Shetty et al. 2024a) frames this problem as the maximization of an unnormalized probability density function (PDF) $F(\boldsymbol{x})$, which is derived from $f(\boldsymbol{x})$ through a monotonically non-increasing transformation. The TT-Cross algorithm is then used to compute the TT approximation of $F(\boldsymbol{x})$, denoted by $\boldsymbol{\mathcal{F}}$:

$$F(x_1, \ldots, x_d) \approx \sum_{\gamma_1=1}^{r_1} \sum_{\gamma_2=1}^{r_2} \cdots \sum_{\gamma_{d-1}=1}^{r_{d-1}} \boldsymbol{\mathcal{F}}^1(1,\, x_1,\, \gamma_1)\boldsymbol{\mathcal{F}}^2(\gamma_1,\, x_2,\, \gamma_2)\cdots\boldsymbol{\mathcal{F}}^d(\gamma_{d-1},\, x_d,\, 1), \quad (5)$$

where each $\boldsymbol{\mathcal{F}}^k$ is a TT core of size $r_{k-1} \times n_k \times r_k$, with n_k denoting the number of discretization points for x_k, and the TT ranks r_k typically remaining small even in high dimensions.

Given the TT representation of $F(\boldsymbol{x})$ as $\boldsymbol{\mathcal{F}}$, we can exploit its separable structure to perform *coordinate-wise* optimization by iteratively selecting and refining promising candidates across dimensions. In contrast to naive methods that require exhaustive search over an exponentially large grid (i.e., $\mathcal{O}(n^d)$), the TT format enables localized and efficient search for a global (or near-global) optimum with significantly lower complexity (i.e., $\mathcal{O}(ndr^2)$). Furthermore, this approach is entirely gradient-free, making it well-suited for high-dimensional and non-convex landscapes. For further technical details on these procedures, we refer the readers to (Oseledets 2011; Sozykin et al. 2022; Shetty et al. 2024a).

3.5 Generalized Policy Iteration using Tensor Train (TTPI)

Optimal control of dynamical systems with nonlinear dynamics poses a substantial challenge in robotics. To address this, Generalized Policy Iteration using Tensor Train (TTPI) was proposed (Shetty et al. 2024b), leveraging tensor approximation and approximate dynamic programming. This method approximates state-value and advantage functions using Tensor Train (TT), effectively mitigating the curse of dimensionality. The low-rank structure of TT enables the use of TTGO to find near-global solutions during policy retrieval from the advantage function, even under complex nonlinear system constraints, surpassing the capabilities of existing neural network-based algorithms. TTPI has demonstrated superior performance on several hybrid control problems. In this article, we further extend this approach to learning robust manipulation primitives for contact-rich tasks involving diverse contact parameters.

4 Problem formulation

Let us consider a discrete-time dynamical system:

- **State** $\boldsymbol{x} \in \Omega_{\boldsymbol{x}} \subseteq \mathbb{R}^m$: The system's state space.
- **Action** $\boldsymbol{u} \in \Omega_{\boldsymbol{u}} \subseteq \mathbb{R}^n$: The system's action space.
- **Parameters** $\boldsymbol{\alpha} \in \Omega_{\boldsymbol{\alpha}} \subseteq \mathbb{R}^d$: The unknown or uncertain physical parameters of the system, such as mass, friction coefficient, etc.
- **Dynamics Model** $f : \Omega_{\boldsymbol{\alpha}} \times \Omega_{\boldsymbol{x}} \times \Omega_{\boldsymbol{u}} \to \Omega_{\boldsymbol{x}}$: The system's next state depends on current state $\boldsymbol{x}_t$, action $\boldsymbol{u}_t$ and the system parameters $\boldsymbol{\alpha}$.
- **Reward function** $R : \Omega_{\boldsymbol{x}} \times \Omega_{\boldsymbol{u}} \to \mathbb{R}$: The immediate reward, which depends on the current state $\boldsymbol{x}_t$, action $\boldsymbol{u}_t$.
- **Discount factor** $\gamma \in [0, 1]$.

This system can be described by a Markov Decision Process (MDP) $\mathcal{M} = \{\Omega_{\boldsymbol{x}}, \Omega_{\boldsymbol{u}}, \Omega_{\boldsymbol{\alpha}}, f, R, \gamma\}$. The objective is to find a policy π that maximizes the expected cumulative reward

$$\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R(\boldsymbol{x}_t, \pi(\boldsymbol{\alpha}, \boldsymbol{x}_t)) \mid \boldsymbol{x}_0 = \boldsymbol{x}]. \quad (6)$$

However, knowing the exact value of $\boldsymbol{\alpha}$ in the real world is intractable. This information can, however, be estimated using multimodal sensors (Lee et al. 2020b) or proprioceptive history (Bianchini et al. 2023; Lee et al. 2020a). Given the sensor noise and model inaccuracy, it is better to estimate the system parameters as a probabilistic distribution, denoted as $\hat{\boldsymbol{\alpha}} \sim P(\hat{\boldsymbol{\alpha}})$. Therefore, our approach aims to find a policy π that maximizes

$$\mathbb{E}_{\hat{\boldsymbol{\alpha}} \sim P(\hat{\boldsymbol{\alpha}})} \left[\mathbb{E}_\pi \left(\sum_{t=0}^{\infty} \gamma^t R(\boldsymbol{x}_t, \pi(\hat{\boldsymbol{\alpha}}, \boldsymbol{x}_t)) \mid \boldsymbol{x}_0 = \boldsymbol{x} \right) \right]. \quad (7)$$

Solving (7) involves knowing the probability distribution $P(\hat{\boldsymbol{\alpha}})$ and then finding the policy π that maximizes the expected cumulative reward over this distribution. One approach to address this problem is to learn control policies separately for each instance with its specific parameter distribution, akin to performing domain adaptation within each target domain. However, a single manipulation primitive typically involves numerous instances, making this approach too time-consuming and impractical. An alternative is to learn an augmented base policy that encompasses multiple instances and retrieve instance-specific policies using estimated instance information, similarly to the *explicit motor adaptation (EMA)* strategy. However, EMA cannot solve (7) as it focuses only on one specific parameter instance rather than a distribution.

In this work, we present an *implicit motor adaptation (IMA)* approach that addresses the problem of probabilistic system identification and, more importantly, the retrieval of a robust policy conditioned on the distribution of parameters. Additionally, to reduce the computational burden of online control, we employ tensor factorization as an implicit representation of the base policy, enabling efficient parameter-conditioned policy retrieval through tensor core products.

5 Implicit Motor Adaptation (IMA)

In this section, we provide a detailed introduction to *implicit motor adaptation (IMA)*, focusing on its three key components: 1) parameter-augmented base policy learning, 2) probabilistic system adaptation conditioned on proprioceptive history, and 3) parameter-conditioned policy retrieval via domain contraction. The full pipeline is shown in Fig. 3.

5.1 Parameter-augmented Base Policy Learning

Similarly to the multi-goal setting in DRL (Liu et al. 2022), we first train a parameter-augmented policy by augmenting

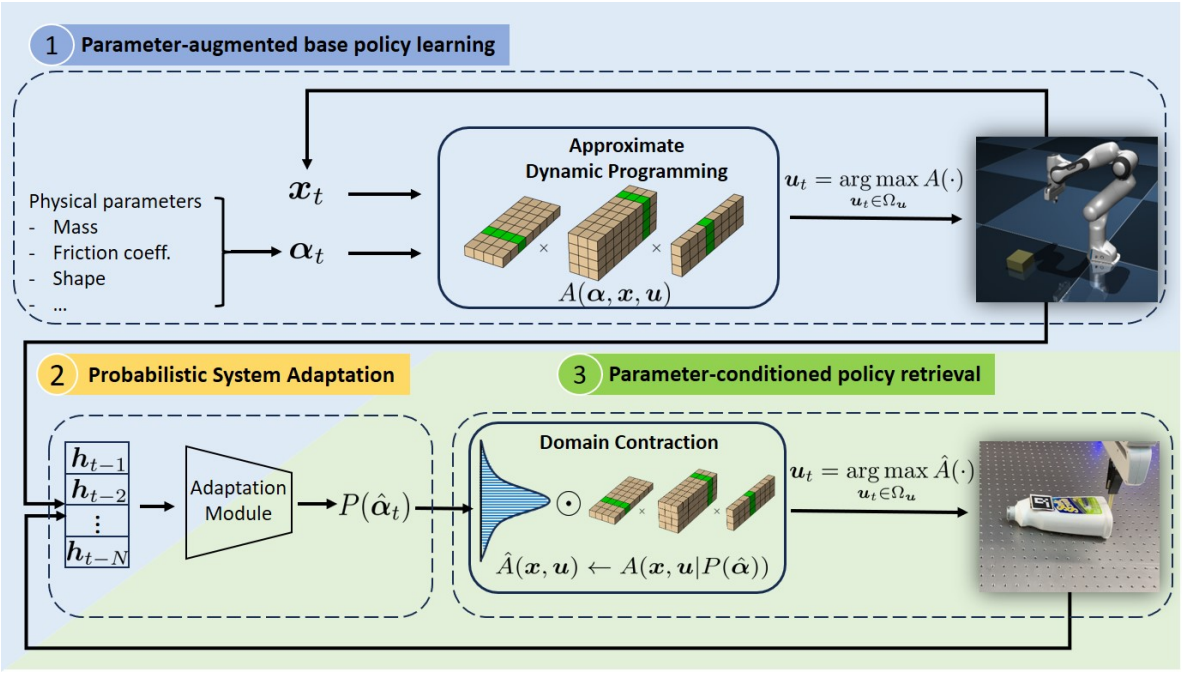


Figure 3. Pipeline of the proposed approach, including (1) parameter-augmented base policy learning, (2) probabilistic system adaptation with proprioceptive history, and (3) parameter-conditioned policy retrieval. The base policy and parameter-conditioned policy are implicitly represented by the corresponding advantage functions $A(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u})$ and $\hat{A}(\boldsymbol{x}, \boldsymbol{u})$, respectively. Blue-shaded modules are trained in simulation, and green-shaded ones are used in deployment, with the probabilistic system adaptation module bridging both stages.

the state space with parameters, treating both the state $\boldsymbol{x}$ and parameter $\boldsymbol{\alpha}$ as inputs and the action $\boldsymbol{u}$ as the output. This policy serves as the base policy for parameter-conditioned policy retrieval in Section 5.3, and is also used for data collection to train the probabilistic adaptation module discussed in Section 5.2. The resulting parameter-augmented bellman equation is

$$\begin{aligned} V(\boldsymbol{\alpha}, \boldsymbol{x}) &= \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(\boldsymbol{x}_t, \boldsymbol{u}_t) \mid \boldsymbol{x}_0 = \boldsymbol{x} \right], \\ A(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u}) &= R(\boldsymbol{x}, \boldsymbol{u}) + \gamma(V(f(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u})) - V(\boldsymbol{\alpha}, \boldsymbol{x})), \\ \pi(\boldsymbol{\alpha}, \boldsymbol{x}) &= \arg\max_{\boldsymbol{u} \in \Omega_{\boldsymbol{u}}} A(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u}). \end{aligned} \tag{8}$$

where $V(\boldsymbol{\alpha}, \boldsymbol{x})$ is the parameter-augmented state-value function, and $A(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u})$ is the parameter-augmented advantage function.

Solving (8) is typically a challenging ADP problem due to the curse of dimensionality. In this article, we tackle this challenge by building on our prior work, TTPI (Shetty et al. 2024b), which utilizes tensor factorization techniques for function approximation. The Value Iteration algorithm (Pashenkova et al. 1996) is used to determine the optimal value function. At any iteration k, the $(k+1)$-th value function approximation is computed as

$$\begin{aligned} V^{k+1} &= \text{TT-Cross}(\mathcal{B}^{\pi_k} V^k, \epsilon), \\ \mathcal{B}^{\pi_k} V^k(\boldsymbol{\alpha}, \boldsymbol{x}) &= R(\boldsymbol{x}, \pi^k(\boldsymbol{\alpha}, \boldsymbol{x})) + \gamma V^k(f(\boldsymbol{\alpha}, \pi^k(\boldsymbol{\alpha}, \boldsymbol{x}))) \\ \pi^k(\boldsymbol{x}) &= \operatorname*{argmax}_{\boldsymbol{u} \in \Omega_{\boldsymbol{u}}} A^k(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u}), \\ A^k(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u}) &= R(\boldsymbol{x}, \boldsymbol{u}) + \gamma(V^k(f(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u})) - V^k(\boldsymbol{\alpha}, \boldsymbol{x})), \end{aligned} \tag{9}$$

where ϵ is the accuracy threshold of TT-cross approximation.

To compute V^{k+1} in TT format, the function $\mathcal{B}^{\pi_k} V^k$ is queried iteratively using TT-Cross$(\mathcal{B}^{\pi_k} V^k, \epsilon)$, with batches of states (usually ranging from 1000 to 100,000 in practice). This requires computing the policy $\pi^k(\boldsymbol{\alpha}, \boldsymbol{x}) = \arg\max_{\boldsymbol{u} \in \Omega_{\boldsymbol{u}}} A^k(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u})$ numerous times across several iterations, making a fast computation of $\operatorname{argmax}_{\boldsymbol{u} \in \Omega_{\boldsymbol{u}}} A^k(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u})$ in batch form crucial.

To resolve the bottleneck, the advantage function A^k is computed in TT format using TT-Cross. This is efficient as the calculation only requires evaluating V^k and R, which are cheap to compute. This enables the use of TTGO (Shetty et al. 2024a), an efficient optimization technique for a function in TT format. As a result, solutions for $\pi^k(\boldsymbol{\alpha}, \boldsymbol{x}) = \operatorname{argmax}_{\boldsymbol{u} \in \Omega_{\boldsymbol{u}}} A^k(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u})$ over batches of states can be obtained quickly, leading to efficient policy retrieval and value iteration. The optimal value function and advantage function are obtained upon algorithm convergence, resulting in functions in TT format:

$$\begin{aligned} V(\boldsymbol{\alpha}, \boldsymbol{x}) &\approx \boldsymbol{\mathcal{V}}(\alpha_{1:d}, \boldsymbol{x}_{1:m}) \\ &= \boldsymbol{\mathcal{V}}^1_{:,i_1,:} \cdots \boldsymbol{\mathcal{V}}^d_{:,i_d,:} \, \boldsymbol{\mathcal{V}}^{d+1}_{:,i_{d+1},:} \cdots \boldsymbol{\mathcal{V}}^{d+m}_{:,i_{d+m},:}, \end{aligned} \tag{10}$$

$$\begin{aligned} A(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u}) &\approx \boldsymbol{\mathcal{A}}(\alpha_{1:d}, \boldsymbol{x}_{1:m}, \boldsymbol{u}_{1:l}) \\ &= \boldsymbol{\mathcal{A}}^1_{:,i_1,:} \cdots \boldsymbol{\mathcal{A}}^d_{:,i_d,:} \, \boldsymbol{\mathcal{A}}^{d+1}_{:,i_{d+1},:} \cdots \boldsymbol{\mathcal{A}}^{d+m}_{:,i_{d+m},:} \\ &\quad \boldsymbol{\mathcal{A}}^{d+m+1}_{:,i_{d+m+1},:} \cdots \boldsymbol{\mathcal{A}}^{d+m+l}_{:,i_{d+m+l},:}. \end{aligned} \tag{11}$$

5.2 Probabilistic System Adaptation

The system parameters $\boldsymbol{\alpha}$ (also called privileged environment information) is typically not accessible during execution in the real world. However, we can probabilistically

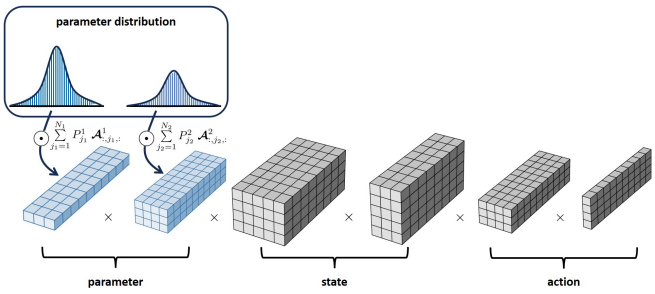


Figure 4. Domain contraction in TT format.
The parameter-augmented advantage function in TT format typically includes separate 3rd-order cores for different dimensionality, such as parameter, state and action. Given a probabilistic parameter distribution, we can retrieve the parameter-conditioned policy by making product of parameter distributions and corresponding TT cores.

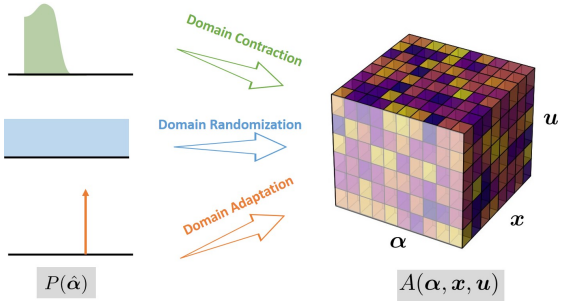


Figure 5. Domain contraction unifies domain randomization and domain adaptation by giving different parameter distributions.

estimate it as $\hat{\boldsymbol{\alpha}}$, drawn from a probability distribution $P(\hat{\boldsymbol{\alpha}})$. This distribution can be inferred from the proprioceptive history, namely

$$P(\hat{\boldsymbol{\alpha}}_t) = \phi(\boldsymbol{x}_{t-k:t-1}, \boldsymbol{u}_{t-k,t-1}). \tag{12}$$

We name this process as probabilistic system adaptation. To learn the mapping between the proprioceptive history and $P(\hat{\boldsymbol{\alpha}})$, a dataset is required containing the state-action history $\boldsymbol{h} = \{\boldsymbol{x}_{t-k:t-1}, \boldsymbol{u}_{t-k:t-1}\}$ and the corresponding system parameter $\boldsymbol{\alpha}_t$, which is accessible in simulation. Various approaches, such as energy-based models (EBMs) LeCun et al. (2006) and diffusion models Ho et al. (2020), can be employed to learn this distribution. In this article, the subsequent probabilistic policy retrieval (Section 5.3) reduces the strong reliance on precise parameter estimation. We use a straightforward multilayer perceptron (MLP) to approximate ϕ, while noting that more sophisticated models could be explored in future extensions.

In practice, the loss function is defined as MSE $(\boldsymbol{\nu}_t, \boldsymbol{\alpha}_t) = \|\boldsymbol{\nu}_t - \boldsymbol{\alpha}_t\|^2$, where $\boldsymbol{\nu}_t$ is the output of MLP. We assume that $P(\hat{\boldsymbol{\alpha}}_t)$ follows a uniform distribution $\mathbb{U}(\boldsymbol{\nu}_t - \boldsymbol{w}/2, \boldsymbol{\nu}_t + \boldsymbol{w}/2)$, where $\boldsymbol{w}$ is a hyperparameter representing the bandwidth of $\mathbb{U}$. Note that our framework is flexible and can incorporate other distributions as well.

5.3 Parameter-conditioned policy retrieval

Without loss of generality, we assume the contact-rich manipulation task involves d types of different parameters, such as mass, size and friction coefficient. The parameter space Ω_α is therefore composed of d subspaces, namely $\Omega_\alpha = \Omega_{\alpha_1} \times \cdots \times \Omega_{\alpha_d}$. We assume each subspace is discretized by N_i points. Let $\boldsymbol{\alpha}_j = (\alpha_{j_1}, \cdots, \alpha_{j_d})$ represents one instance of domain parameter, at the discretization index j across all d dimensions. Similarly, the parameter distribution $P(\hat{\boldsymbol{\alpha}})$ is defined within Ω_P, also composed of d subspaces, namely $\Omega_P = \Omega_{P_1} \times \cdots \times \Omega_{P_d}$, and each subspace is discretized by N_i points. Let $P_j = P_1(\hat{\alpha}_{j_1}) P_2(\hat{\alpha}_{j_2}) \cdots P_d(\hat{\alpha}_{j_d})$ represents the probability of the estimated value $\hat{\boldsymbol{\alpha}}_j$, where P_i is the probability distribution at dimension i, $i \in \{1, \cdots, d\}$.

Given the estimated parameter distribution $P(\hat{\boldsymbol{\alpha}})$, obtaining the corresponding parameter-conditioned policy $\pi(\boldsymbol{x}|P(\hat{\boldsymbol{\alpha}}))$ is not free. It is neither simply averaging the parameter estimates $\sum_{j=1}^{N} P_j \hat{\boldsymbol{\alpha}}_j$ and directly feeding it into $\pi(\boldsymbol{\alpha}, \boldsymbol{x})$, nor merely taking the weighted sum of parameter-specific policies $\sum_{j=1}^{N} P_j \pi(\hat{\boldsymbol{\alpha}}_j, \boldsymbol{x})$. Instead, it involves finding the $\arg\max$ of the weighted sum of parameter-specific advantage functions, namely

$$A(\boldsymbol{x}, \boldsymbol{u}|P(\hat{\boldsymbol{\alpha}})) = \sum_{j_1=1}^{N_1} \cdots \sum_{j_d=1}^{N_d} P_j A_{\hat{\boldsymbol{\alpha}}_j}(\boldsymbol{x}, \boldsymbol{u}). \tag{13}$$

We call this process as domain contraction in our previous work (Xue et al. 2024a), as illustrated in Fig. 4. Intuitively, domain contraction randomizes over a contracted domain based on instance-specific parameter distribution, instead of randomizing over the full domain by blindly ignoring all instance-specific information (as in domain randomization). We further provide the theoretical proof below:

Proposition 1. *Given the estimated parameter distribution $P(\hat{\boldsymbol{\alpha}})$, the parameter-conditioned policy can be retrieved from the weighted sum of parameter-specific advantage functions.*

Proof. Obtaining the parameter-conditioned policy involves optimizing the parameter-augmented function

$$\begin{aligned} &A(\boldsymbol{x}, \boldsymbol{u}|P(\hat{\boldsymbol{\alpha}})) \\ =&R(\boldsymbol{x}, \boldsymbol{u}) + \gamma\Big(V\big(f(\boldsymbol{x}, \boldsymbol{u}|P(\hat{\boldsymbol{\alpha}}))\big) - V(\boldsymbol{x}|P(\hat{\boldsymbol{\alpha}}))\Big), \\ &\forall (P, \boldsymbol{x}, \boldsymbol{u}) \in \Omega_P \times \Omega_{\boldsymbol{x}} \times \Omega_{\boldsymbol{u}}, \end{aligned} \tag{14}$$

with

$$\begin{aligned} &V(\boldsymbol{x}|P(\hat{\boldsymbol{\alpha}})) = \sum_{j_1=1}^{N_1} \cdots \sum_{j_d=1}^{N_d} P_{(j_1,\cdots,j_d)}\, V(\boldsymbol{x}|\hat{\alpha}_{j_1}, \cdots, \hat{\alpha}_{j_d}), \\ &\sum_{j_1=1}^{N_1} \cdots \sum_{j_d=1}^{N_d} P_{(j_1,\cdots,j_d)} = 1. \end{aligned} \tag{15}$$

For simplicity, we use $\hat{\boldsymbol{\alpha}}_j$ to represent $(\hat{\alpha}_{j_1}, \cdots, \hat{\alpha}_{j_d})$ and P_j to represent $P_(j_1, \cdots, j_d)$. Given that each subspace Ω_{α_i} of the parameter space Ω_α is discretized by N_i points, the parameter-conditioned advantage function can be written as

$$\begin{aligned} &A(\boldsymbol{x}, \boldsymbol{u}|P(\hat{\boldsymbol{\alpha}})) \\ =&R(\boldsymbol{x}, \boldsymbol{u}) + \sum_{j_1=1}^{N_1} \cdots \sum_{j_d=1}^{N_d} P_j\, \gamma\big(V\big(f(\boldsymbol{x}, \boldsymbol{u}|\hat{\boldsymbol{\alpha}}_j)\big) - V(\boldsymbol{x}|\hat{\boldsymbol{\alpha}}_j)\big). \end{aligned} \tag{16}$$

The right side of (16) can be further rewritten as

$$\sum_{j_1=1}^{N_1}\cdots\sum_{j_d=1}^{N_d} P_j\Big(R(\boldsymbol{x},\boldsymbol{u})+\gamma\big(V\big(f(\boldsymbol{x},\boldsymbol{u}|\hat{\boldsymbol{\alpha}}_j)\big)-V(\boldsymbol{x}|\hat{\boldsymbol{\alpha}}_j)\big)\Big), \tag{17}$$

and the parameter-specific advantage function is defined as

$$A_{\hat{\boldsymbol{\alpha}}_j}(\boldsymbol{x},\boldsymbol{u}) = R(\boldsymbol{x},\boldsymbol{u})+\gamma\big(V\big(f(\boldsymbol{x},\boldsymbol{u}|\hat{\boldsymbol{\alpha}}_j)\big)-V(\boldsymbol{x}|\hat{\boldsymbol{\alpha}}_j)\big). \tag{18}$$

From (16), (17), and (18), we can derive that the parameter-conditioned advantage function is the weighted sum of parameter-specific advantage functions, as shown in (13), and the parameter-conditioned primitive policy can then be computed by

$$\pi(\boldsymbol{x}|P(\hat{\boldsymbol{\alpha}})) = \arg\max_{\boldsymbol{u}\in\Omega_{\boldsymbol{u}}} A(\boldsymbol{x},\boldsymbol{u}|P(\hat{\boldsymbol{\alpha}})). \tag{19}$$

Note that the parameter-conditioned policy cannot be directly computed as the weighted sum of parameter-specific policies, since the $\arg\max$ operation does not have an associative property with respect to addition.

Computing (13) and then retrieve the parameter-conditioned policy can be computationally expensive due to the combinatorial complexity and $\arg\max$ over an arbitray function. We address this issue by leveraging the separable structure of TT format for efficient algebraic operation, and its advantage of finding optimal solutions for functions in TT format. In Sec. 5.1, we have approximated the parameter-augmented advantage functions in TT format. We define the tensor cores related to $\boldsymbol{x}$ and $\boldsymbol{u}$ in (11) as

$$\boldsymbol{\mathcal{A}}(\boldsymbol{x}_{1:m},\boldsymbol{u}_{1:l}) = \boldsymbol{\mathcal{A}}^{d+1}_{:,i_{d+1},:}\cdots\boldsymbol{\mathcal{A}}^{d+m}_{:,i_{d+m},:}\cdots\boldsymbol{\mathcal{A}}^{d+m+l}_{:,i_{d+m+l},:}, \tag{20}$$

thus the parameter-specific advantage functions can be extracted as

$$\begin{aligned} A_{\hat{\boldsymbol{\alpha}}_j}(\boldsymbol{x},\boldsymbol{u}) &\approx \boldsymbol{\mathcal{A}}(\boldsymbol{x}_{1:m},\boldsymbol{u}_{1:l}|\hat{\boldsymbol{\alpha}}_j) \\ &= \boldsymbol{\mathcal{A}}^1_{:,j_1,:}\cdots\boldsymbol{\mathcal{A}}^d_{:,j_d,:}\,\boldsymbol{\mathcal{A}}(\boldsymbol{x}_{1:m},\boldsymbol{u}_{1:l}), \end{aligned} \tag{21}$$

The parameter probability P_j for parameter instance $\hat{\boldsymbol{\alpha}}_j$ is given by

$$P_{(j_1,\ldots,j_d)} = P_1(\hat{\alpha}_{j_1})P_2(\hat{\alpha}_{j_2})\cdots P_d(\hat{\alpha}_{j_d}). \tag{22}$$

For simplicity, we denote $P_d(\hat{\alpha}_{j_d})$ as $P^d_{j_d}$. The parameter-conditioned advantage function in (13) can then be efficiently computed as the weighted sum of parameter-specific advantage functions for each dimension, utilizing the separable structure of the TT model, namely

$$\begin{aligned} A(\boldsymbol{x},\boldsymbol{u}|P(\hat{\boldsymbol{\alpha}})) &\approx \sum_{j_1=1}^{N_1}\cdots\sum_{j_d=1}^{N_d} P_{(j_1,\ldots,j_d)}\boldsymbol{\mathcal{A}}(\boldsymbol{x}_{1:m},\boldsymbol{u}_{1:l}|\hat{\boldsymbol{\alpha}}_j) \\ &= \sum_{j_1=1}^{N_1} P^1_{j_1}\,\boldsymbol{\mathcal{A}}^1_{:,j_1,:}\cdots\sum_{j_d=1}^{N_d} P^d_{j_d}\,\boldsymbol{\mathcal{A}}^d_{:,j_d,:}\,\boldsymbol{\mathcal{A}}(\boldsymbol{x}_{1:m},\boldsymbol{u}_{1:l}). \end{aligned} \tag{23}$$

The computation of weighted sum is in tensor core level, which is much more efficient than in the function level. After obtaining the parameter-conditioned advantage function, we can retrieve the parameter-conditioned policy using TTGO (Shetty et al. 2024a), a specialized method for efficiently finding globally optimal solutions given functions in TT format.

To retrieve such parameter-conditioned policies, knowing the parameter distribution $P(\hat{\boldsymbol{\alpha}})$ is crucial. Domain randomization (DR) and domain adaptation (DA) rely on assumed distributions during training, while domain contraction (DC) offers a better way to leverage domain knowledge during execution, enabling optimal behaviors while preserving generalization capability. We further demonstrate that DR and DA are two special cases of DC, as shown in Figure 5 and detailed in (Xue et al. 2024a).

5.4 Theoretical analysis of IMA compared with EMA

EMA is a commonly used approach for robust policy learning. Given that precise system identification is intractable in real-world scenarios, we present a theoretical proof demonstrating that the control policy obtained by IMA is theoretically superior to that of EMA for practical applications.

Proposition 2. *Given an estimate of domain parameter $\boldsymbol{\alpha}$, the parameter-conditioned policy derived through implicit motor adaptation is more optimal than that obtained through explicit motor adaptation.*

Proof. Knowing the exact parameter $\boldsymbol{\alpha}$ for the real-world domain is intractable, but we can represent it probabilistically. Let $P(\hat{\boldsymbol{\alpha}})$ denote the probabilistic estimation obtained using any off-the-shelf approach. Therefore, the objective of robust policy learning is to find a policy π that maximizes

$$\begin{aligned} &J_\pi(\boldsymbol{x},P(\hat{\boldsymbol{\alpha}})) = \mathbb{E}_{\hat{\boldsymbol{\alpha}}\sim P(\hat{\boldsymbol{\alpha}})}\left[\mathbb{E}_\pi\Big[\sum_{t=0}^{\infty}\gamma^t R(\boldsymbol{x}_t,\boldsymbol{u}_t)\Big|\boldsymbol{x}_0=\boldsymbol{x}\Big]\right], \\ &\text{s.t.}\quad \boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t,\boldsymbol{u}_t,\hat{\boldsymbol{\alpha}}). \end{aligned} \tag{24}$$

The key difference between IMA and EMA lies in the method used to retrieve the parameter-conditioned policy. EMA determines the domain parameter $\boldsymbol{\alpha}$ with a single estimate, defined as $\hat{\boldsymbol{\alpha}} = f_\theta(\boldsymbol{h})$. The obtained $\hat{\boldsymbol{\alpha}}$ is then used as input to the parameter-augmented base policy. The resulting parameter-conditioned policy π_e aims to maximize

$$V(\boldsymbol{x}|\hat{\boldsymbol{\alpha}}) = V(\hat{\alpha}_{j_1},\cdots,\hat{\alpha}_{j_d},\boldsymbol{x}). \tag{25}$$

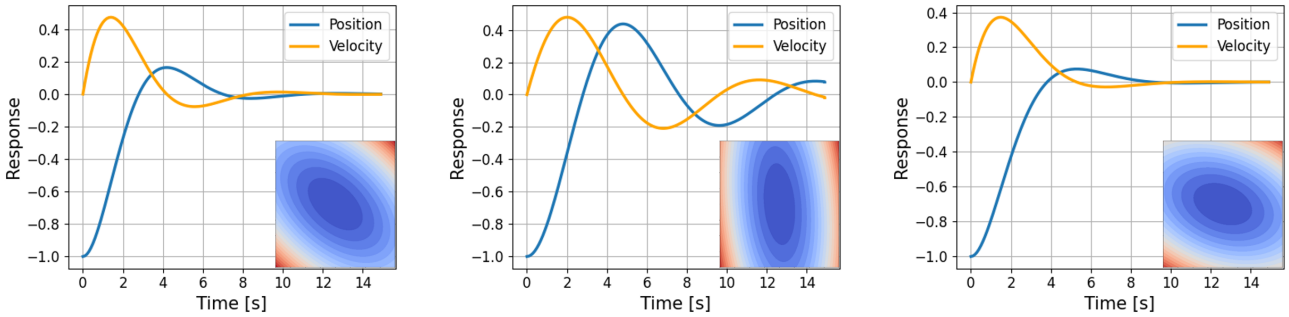
IMA estimates the true parameter $\boldsymbol{\alpha}$ using a probability distribution $\hat{\boldsymbol{\alpha}}\sim P(\hat{\boldsymbol{\alpha}})$, and then retrieves the parameter-conditioned policy π_i through domain contraction, with the objective of maximizing

$$V(\boldsymbol{x}|P(\hat{\boldsymbol{\alpha}})) = \sum_{j_1=1}^{N_1}\cdots\sum_{j_d=1}^{N_d} P_{(j_1,\cdots,j_d)}V(\hat{\alpha}_{j_1},\cdots,\hat{\alpha}_{j_d},\boldsymbol{x}), \tag{26}$$

where $\hat{\boldsymbol{\alpha}}\sim P(\hat{\boldsymbol{\alpha}})$.

Given the same dataset used in EMA and IMA adaptation module training, $\hat{\boldsymbol{\alpha}}$ can be considered a single sample from $P(\hat{\boldsymbol{\alpha}})$. Therefore, we have

$$J_{\pi_i}(\boldsymbol{x},P(\hat{\boldsymbol{\alpha}})) \geq J_{\pi_e}(\boldsymbol{x},P(\hat{\boldsymbol{\alpha}})). \tag{27}$$



(a) Ground truth **(b)** EMA result **(c)** IMA result

Figure 6. Position and velocity responses of a spring-damper system, along with the approximated value functions (figure insets as colormaps), for the policies derived from the ground truth, EMA, and IMA.

This indicates that the policy obtained by EMA typically demonstrates poorer performance compared to that obtained by IMA.

Example 1. *Spring-damper System*

To numerically illustrate the difference, we provide a spring-damper system with parameters $\boldsymbol{\alpha} = (m, k, b)$ as a toy example, where m represents the mass, k the spring stiffness, and b the damping coefficient. The system is described by the following state-space equations

$$\dot{\boldsymbol{x}}_t = \mathbf{A}\boldsymbol{x}_t + \mathbf{B}u_t,$$

where $\boldsymbol{x}_t$ includes the displacement and velocity, and u_t is the control input. The system matrices are expressed as

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}.$$

The optimal control law and value function can be obtained by solving the Algebraic Riccati Equation (ARE) (Lancaster 1995), and they are strongly influenced by the parameter values.

We assume that the real system parameters $\boldsymbol{\alpha} = [m, k, b]$ are unknown but can be estimated as a probability distribution $\hat{\boldsymbol{\alpha}} \sim P(\hat{\boldsymbol{\alpha}})$. Fig. 6 compares the results of EMA and IMA with the ground-truth optimal policy. The value function obtained by IMA closely aligns with the ground truth, effectively stabilizing the mass at the desired equilibrium position. In contrast, the EMA policy shows significantly poorer performance. This toy example highlights the effectiveness of probabilistic system adaptation and implicit policy retrieval in a dynamical system with unknown parameters.

6 Experimental Results

We validate the effectiveness of the proposed method on three contact-rich manipulation tasks: `Hit`, `Push`, and `Reorientation`, as shown in Fig. 1. All the primitives are highly dependent on the physical parameters between the object, the robot, and the surroundings.

6.1 Base policy learning

We first learn the parameter-augmented base policy over a range of system parameters using tensor approximation. Since the physical dynamics of `Hit` are fully known, the control policy can be derived analytically. For `Push` and `Reorientation`, TTPI (Shetty et al. 2024b) is used to compute the parameter-augmented value functions and advantage functions.

Hit: Hitting involves manipulating objects through impact, which is a representative one-shot manipulation task. This means the object can only be hit once, with no additional actions allowed. Determining an appropriate impact is therefore crucial and varies significantly depending on instance-specific domain parameters. In this work, we focus on the planar hitting primitive. The state is the object position, denoted as $\boldsymbol{x} = [x, y]$, and the control input is the applied impact $\boldsymbol{u} = [I_x, I_y]$. The physical parameters $\boldsymbol{\alpha}$ include the object mass m and the friction coefficient μ.

Given the mass m, the friction coefficient μ, the initial state $\boldsymbol{x}_0$, and the target $\boldsymbol{x}^{\text{des}}$, we define the following single-step reward:

$$r(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u}) = -\left(\|\boldsymbol{x} - \boldsymbol{x}^{\text{des}}\|^2 \; + \; 0.01\,\|\boldsymbol{u}\|^2\right), \tag{28}$$

where

$$\boldsymbol{x} = \boldsymbol{x}_0 + \frac{\boldsymbol{u}}{m}\,t - 0.5\,\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}\,\mu\,g\,t^2.$$

Since the hitting task terminates after a single action, this immediate reward function r effectively serves the same role that an "advantage function" would in a multi-step setting, differing only by a constant. We use TT-cross to approximate $r(\boldsymbol{\alpha}, \boldsymbol{x}, \boldsymbol{u})$ across diverse parameter instances in TT format.

Push: Pushing is widely recognized as a challenging task in robot planning and control, primarily due to its hybrid and under-actuated nature (Mason 1986; Lynch and Mason 1996; Mason 1999). In this work, we further complicate the problem by considering diverse parameters and aiming to find the instance-aware optimal policy for each. The state of the planar pushing task is characterized by $[s_x, s_y, s_\theta, \psi, \phi]$, and the action is represented as $[f_x, f_y, \dot{\psi}, \dot{\phi}]$. Here, $[s_x, s_y, s_\theta] \in SE(2)$ denotes the position and orientation of the object in the world frame. ψ is the relative angle of the contact point in the object frame, and ϕ represents the distance between the contact point and the object surface. The forces exerted on the object are denoted by $\boldsymbol{f} = [f_x, f_y]^\top$, while $\boldsymbol{v}_p = [\dot{\psi}, \dot{\phi}]^\top$ represents the angular and translational velocities of the robot's end-effector. The physical parameters include the object mass m, radius r, and the friction coefficient μ between the object and the table.

The robot dynamics is defined based on the quasi-static approximation and the limit surface, resulting in a similar expression as (Hogan and Rodriguez 2020a). To learn the parameter-augmented advantage function, the reward function is defined as

$$r = -(\rho c_p + c_o + 0.01 c_f + 0.01 c_v), \tag{29}$$

with

$$\begin{aligned} c_p &= \|\boldsymbol{x}_p - \boldsymbol{x}_p^{\text{des}}\|/l_p, \quad c_o = \|x_o - x_o^{\text{des}}\|/l_o, \\ c_f &= \|\boldsymbol{f}\|, \quad c_v = \|\boldsymbol{v}_p\|, \end{aligned} \tag{30}$$

where $\boldsymbol{x}_p = [s_x, s_y]$ and $x_o = \theta$ denote the object's position and orientation. Without loss of generality, we set $\boldsymbol{x}^{\text{des}} = \boldsymbol{0}$ as the target configuration. $\boldsymbol{f}$ is the control force, while $\boldsymbol{v}_p$ is the velocity of the robot's end-effector. l_p and l_o are set to 0.005 and 0.01π, respectively.

Reorientation: In this task, we aim to enable the robot to reorient an object using parallel fingers. The state is the orientation angle θ, and the control input is the normal force f_n between the gripper and object. The physical parameters are the object mass m, length l and torsional friction coefficient μ. The gravitational torque and normal force f_n are used as braking mechanisms to slow down the object motion. We build the dynamics model of the reorientation primitive based on (Viña Barrientos et al. 2016) as

$$\begin{aligned} I\ddot{\theta} &= \tau_g + 2\tau_f, \\ \dot{\theta} &= \dot{\theta}_0 - \ddot{\theta}\Delta t, \end{aligned} \tag{31}$$

where $\tau_f = \mu_t f_n^{1+\xi}$ is the torsional sliding friction between robot gripper and the object. In this work, we set $\xi = 0$. μ_t is the torsional friction coefficient, which is related to the materials and normal force distribution. $\tau_g = mgl\sin(\theta)$ is the gravity torque. We therefore include μ_t, object mass m and length l as the model parameters. The task is to rotate any object from its initial configuration to a vertically upward configuration. To achieve this, the object is given an initial angular velocity $\dot{\theta}_0$ by swinging the robot arm.

The reward function for `Reorientation` primitive learning is defined as

$$r = -(\beta c_g + c_f), \tag{32}$$

with $c_g = \|x_o - x_o^{\text{des}}\|$, $c_f = \|f_n\|$. x_o is the orientation angle θ, and f_n is the normal force between the gripper and object. x_o^{des} is set to π as the reorientation goal, and β is set to 10^4.

In our experiments, we employed an NVIDIA GeForce RTX 3090 GPU with 24GB of memory. The accuracy parameter for cross approximation was set to $\epsilon = 10^{-3}$ for TT-Cross approximation. The maximum rank $r_{\max}$ and discount factor were set to 100 and 0.99, respectively. The continuous variables of state, action and parameter domains were discretized as 50 to 500 points using uniform discretization.

6.2 Results of domain contraction for policy retrieval

Given the parameter-augmented advantage functions learned in Section 6.1, we can now retrieve the parameter-conditioned policy for each specific instance through domain contraction (DC). This section aims to demonstrate the unification and flexibility of DC compared to domain randomization (DR) and domain adaptation (DA). To illustrate this, we make a simplifying assumption, without loss of generality, that the system adaptation module provides a uniform distribution as the probabilistic representation of parameters. This distribution spans a range defined by the discretization indices of each dimension (denoted as w), as shown in Table 1 and Fig. 7. Specifically, $w = 1$ indicates that the exact value of the physical parameter is known, corresponding to DA. $w = N$ reflects no prior knowledge about the model parameters, corresponding to DR. These two variants can be seen as extreme cases of DC, where w determines how finely the distribution is utilized for policy retrieval. Nonetheless, our framework is general and flexible, and can accommodate arbitrary parameter distributions, such as those produced by diffusion models (Ho et al. 2020).

Table 1. Cumulative reward of three manipulation tasks

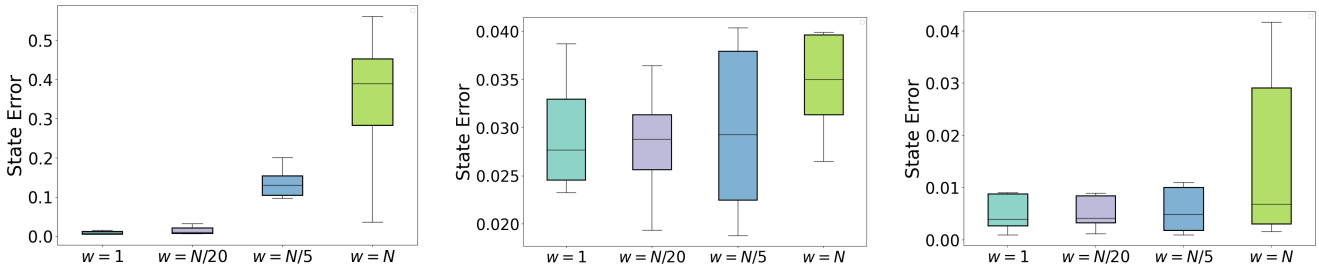
	$w = 1$	$w = N/20$	$w = N/5$	$w = N$
Hit	1.0	0.65 ± 0.21	0.01 ± 0.01	0.02 ± 0.05
Push	1.0	0.99 ± 0.01	0.99 ± 0.03	0.93 ± 0.11
Reori.	1.0	0.99 ± 0.04	0.99 ± 0.07	0.85 ± 0.19

Table 1 and Fig. 7 demonstrate the comparisons of cumulative reward and final state error, respectively. The state error is quantified as the ℓ_2 norm of the difference between the final state and the target state. The cumulative reward is normalized using the value obtained through DA. We can observe that DA ($w = 1$) typically performs the best in all these three tasks, in particular for the `Hit` primitive, as it resembles an open-loop control. Once the impact is given from the robot to the object, there is no way to adjust control inputs to influence the object movements further. However, knowing the exactly correct domain parameter is intractable (as shown in Fig. 10). Instead, domain contraction does not require one specific value and accommodate probabilistic estimation. For example, results show that a rough range ($w = N/20$) is sufficient to achieve the target for the `Hit` primitive. Furthermore, `Push` and `Reorientation` can be considered more akin to closed-loop control, which allows for much more rough parameter estimation. As depicted in Table 1 and Fig. 7, a rough distribution with $w = N/5$ is adequate.

Moreover, based on Table 1, we observe that $w = N$ results in the lowest cumulative reward. This is consistent with our assertion that DR typically leads to conservative behaviors. Although DA($w = 1$) yields the highest cumulative reward, obtaining precise parameter values is often challenging in the real world. DC bridges the gap between domain adaptation and DR, offering greater flexibility to generate optimal behaviors while leveraging instance-specific rough parameter distributions. This is much practical for real-world contact-rich manipulation tasks.

6.3 Results of implicit v.s. explicit motor adaptation

In this section, we compare the performance of IMA and EMA on each task with 10 different instances, by randomizing system parameters and initial conditions while using the same base policy. As shown in Fig. 8, IMA



(a) Hit **(b)** Push **(c)** Reorientation

Figure 7. Comparison of final state error given different estimated parameter distribution.

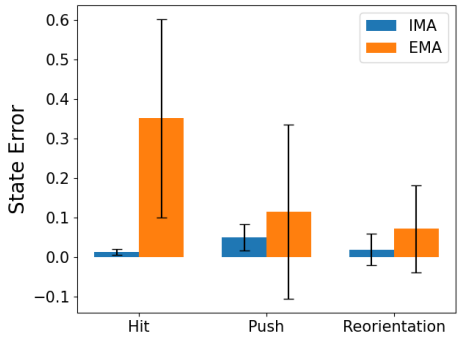


Figure 8. Comparison between IMA and EMA on three manipulation primitives in terms of the final state error, given the same base policy.

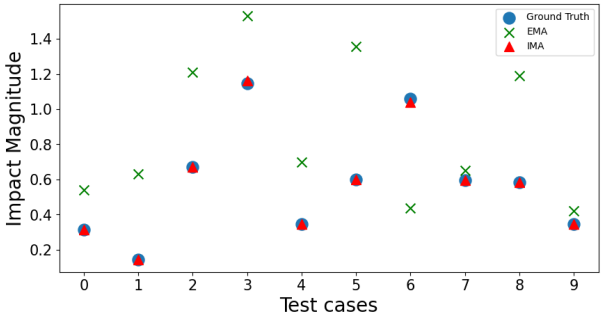


Figure 9. Resulting hitting impact of the policies derived from the ground truth, EMA, and IMA.

generally outperforms EMA on the three contact-rich manipulation tasks, particularly in the open-loop hitting task. In this paper, we employ a simple MLP with a structure of $512 \times 256 \times 128 \times 64$ for parameter estimation. EMA directly uses the output (denoted as $\hat{\boldsymbol{\alpha}}_t$) to retrieve the parameter-conditioned policy from the base policy. In contrast, IMA assumes $\hat{\boldsymbol{\alpha}}_t$ as the mean $\boldsymbol{\nu}_t$ of a uniform distribution $\mathbb{U}(\boldsymbol{\nu}_t - \boldsymbol{w}/2, \boldsymbol{\nu}_t + \boldsymbol{w}/2)$, where the range w is set to $N/20$ for `Hit` and $N/5$ for `Push` and `Reorientation`.

Fig. 9 illustrates the computed impacts for 10 hitting instances with different physical parameters, where the ground truth values are represented by blue dots. By utilizing a probabilistic representation of the estimated parameters and domain contraction for policy retrieval, IMA typically produces values closely aligned with the ground truth, whereas EMA often yields suboptimal results. We further analyze the performance of EMA and IMA under varying levels of parameter estimation accuracy, as shown in Fig. 10, and study the impact of different choices for w in DC. When parameters are precisely estimated, EMA achieves the lowest final state error, demonstrating optimal performance. However, as parameter estimation becomes less accurate, greater randomization (a larger w) is required to obtain a better policy. The dotted line and gray shaded area in the figure indicate the estimation accuracy achieved in our work, with IMA ($w = N/3$) providing the best results. This analysis highlights the effectiveness of IMA in real-world scenarios where system parameters are unknown and can only be roughly estimated. It also demonstrates that domain randomization ($w = N$) is not always good when some domain knowledge is available, even if imperfect. We believe IMA offers a promising approach to find control policies when we have limited access to the domain knowledge.

Figures 11 and 12 further illustrate the performance of IMA and EMA in the pushing and reorientation tasks, respectively. In Fig. 11, the same base policy is used to push a sugar box and a mustard bottle toward their target configurations, including both position and orientation. While both approaches eventually reach the target positions, IMA outperforms EMA in terms of orientation. This is primarily due to the underactuated dynamics of the planar pushing task, where system parameters such as the friction coefficient and object mass play a critical role. IMA estimates these parameters probabilistically and retrieves parameter-conditioned policies implicitly. This allows for increased tolerance of typical system identification errors, enabling robust contact-rich manipulation with unknown parameters. Similarly, Fig. 12 demonstrates the superior performance of IMA over EMA in reorienting an arbitrary object vertically from the bottom to the top.

6.4 Results of robust manipulation primitive learning

We also compared our full pipeline with two widely used robust policy learning approaches: reinforcement learning with domain randomization (RL+DR) (Tobin et al. 2017) and reinforcement learning with explicit motor adaptation (RL+EMA) (Yu et al. 2017; Qi et al. 2023). We utilize Soft Actor-Critic (SAC) (Haarnoja et al. 2018) for policy learning in both RL+DR and RL+EMA, as SAC has been shown to be the state-of-the-art RL algorithm for such contact-rich manipulation tasks, especially planar pushing, as demonstrated in (Potters 2022; Cong et al. 2022). Our implementation is based on Stable-Baselines3 (Raffin et al.

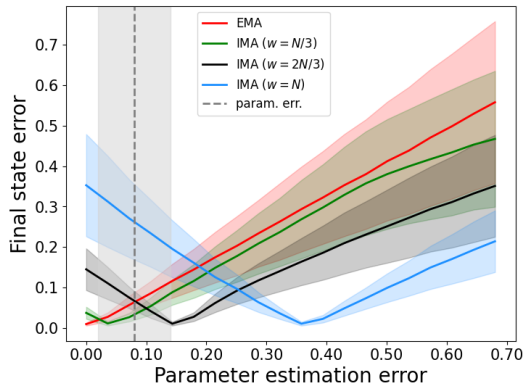


Figure 10. Resulting final state error of different motor adaptation strategies under varying levels of parameter estimation accuracy. The dotted line and gray shaded area represent the mean and standard deviation of the parameter estimation error achieved using the system estimator in this article.

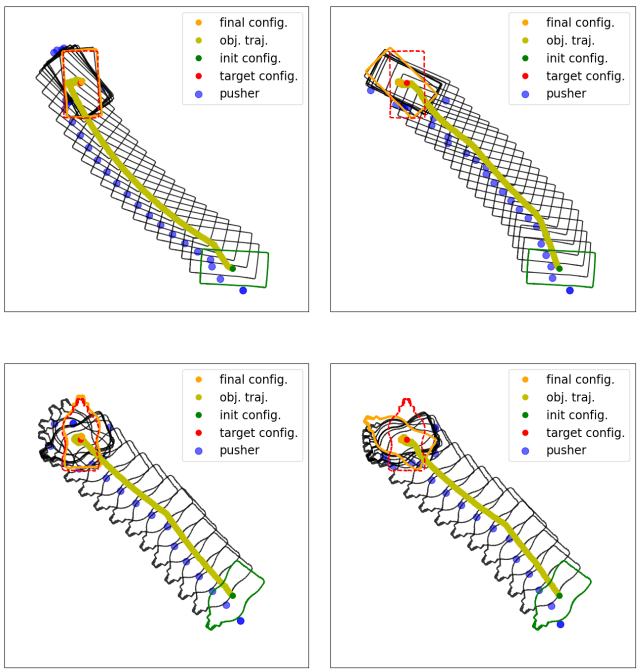


Figure 11. Planar pushing tasks with a sugar box (*Top*) and a mustard bottle (*Bottom*). **Left**: Object trajectory produced by IMA; **Right**: Object trajectory produced by EMA.

2021), using a Multilayer Perceptron (MLP) with a 64 × 64 × 64 architecture as the policy network. The discount factor is set to 0.99, and the learning rate to 0.001.

The quantitative results are presented in Fig. 13, including the time required for policy training and the state error of the retrieved policy. The *base_error* and *adaptation_error* refer to the final state error of the base policy with the true parameter and the parameter-conditioned policy after motor adaptation, respectively. For RL+DR, the *base_error* and *adaptation_error* are the same since no motor adaptation is performed.

Notably, although the `Hit` task is a one-shot manipulation task in which proprioceptive history is unavailable, the motor adaptation module works by observing the trajectories of past rollouts, similarly to how humans perform such tasks.

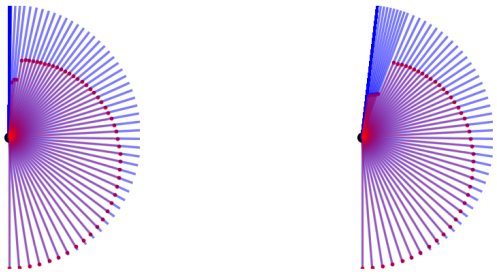


Figure 12. Trajectories produced by IMA (**Left**) and EMA (**Right**) for the reorientation task. The blue lines depict the trajectory of an arbitrary object, with the objective of reorienting it from the bottom configuration to a perfectly vertical top configuration. The lengths of the red lines indicate the angular velocity magnitude, which is indirectly controlled by leveraging gravity and friction forces.

Table 2. Time required for computing parameter-conditioned advantage function using core-level or function-level operations

	core-level	function-level
Hit	0.016s ± 0.002s	0.720s ± 0.236s
Push	0.075s ± 0.003s	18.56s ± 4.748s
Reorientation	0.018s ± 0.003s	8.494s ± 0.561s

The results show that our method requires significantly less time for base policy training compared to the other methods, while achieving the lowest *base_error*, highlighting the advantages of leveraging TT for learning control policies in contact-rich manipulation tasks.

Furthermore, our method (TT+IMA) achieves the lowest final state error among the three approaches, demonstrating the overall effectiveness of our complete approach for robust contact-rich manipulation tasks, with implicit action representation playing a crucial role in retrieving robust policies. An additional observation is that RL+EMA often exhibits the highest standard deviation, indicating that its control policies perform extremely poorly in some instances, whereas IMA has a better statistical performance over the diverse instances.

6.5 Sensitivity analysis and ablation study

Our framework involves several hyperparameters, such as the maximum TT-rank, the discretization granularity, and the bandwidth of the uniform distribution. To analyze their impact on policy performance, we conduct a sensitivity analysis across a range of values for each hyperparameter. Figure 15 shows how policy performance, measured by the ℓ_2 error between the final and target states, varies with the maximum TT-rank and discretization granularity. For comparability across tasks, the errors are normalized by the worst-case performance in each task.

From Figure 15, we observe that setting the maximum rank $r_{\max}$ above 20 leads to consistently low errors across all tasks. For `Push` and `Hit`, even lower ranks (e.g., $r_{\max} = 10$ or 5) are sufficient, indicating that the underlying policies lie in a low-rank functional space. A relatively low tensor rank is sufficient to capture the essential structure, and policy performance remains stable when the rank is increased further. Empirically, we observe that the piecewise-smooth dynamics (Toussaint et al. 2020)

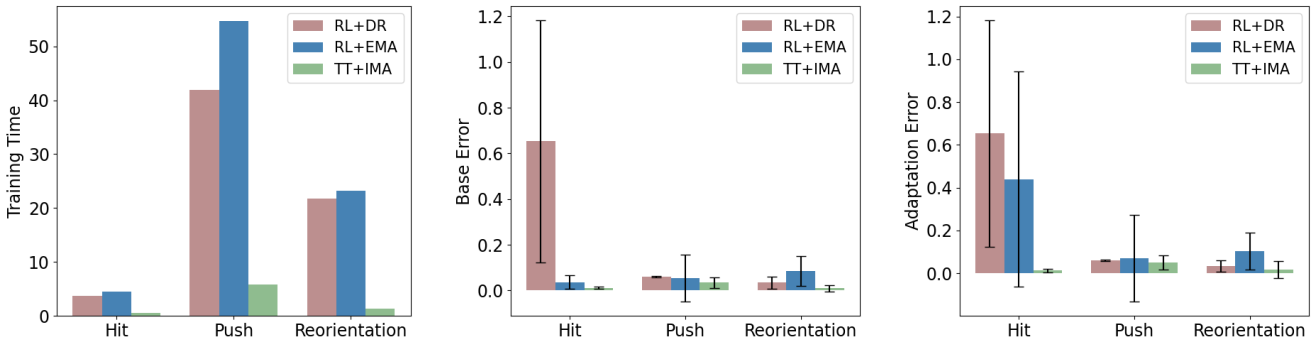
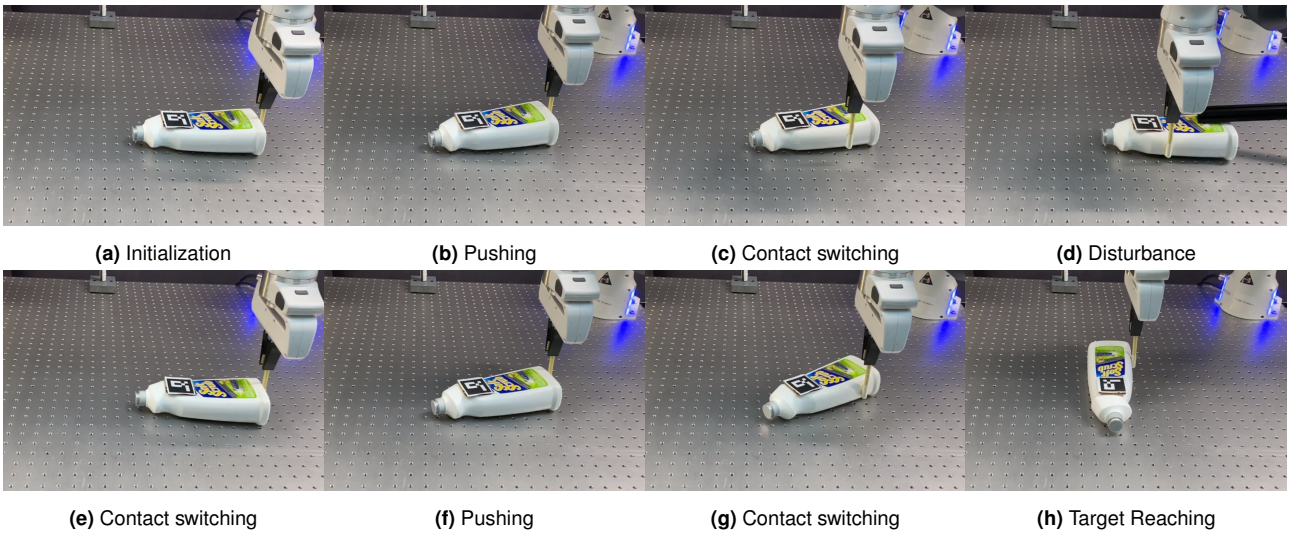


Figure 13. Comparison of different robust primitive learning approaches in terms of training time, and state errors of base policies and adapted policies. The unit of training time is seconds for `Hit` and minutes for `Push` and `Reorientation`. Errors are calculated based on the ℓ_2 norm between the final configuration and the target configuration.



(a) Initialization **(b)** Pushing **(c)** Contact switching **(d)** Disturbance

(e) Contact switching **(f)** Pushing **(g)** Contact switching **(h)** Target Reaching

Figure 14. An example of planar pushing task: bleach bottle on a metal surface. The system begins in the initial configuration (a), aiming to manipulate the bottle to reach the target configuration (h). The robot first makes contact with the bottle and pushes it slightly (b), then switches the contact point (c). When an external disturbance (d) is introduced by a human, the robot adapts by switching the contact point again (e) to accommodate the unexpected change. It continues pushing the bottle further (f), performs another contact switch (g), and ultimately reaches the target configuration (h).

commonly encountered in contact-rich manipulation tasks induce structured, low-dimensional variations in the state-action space. The corresponding advantage functions often exhibit a low-rank structure, which can be efficiently captured using the TT representation.

Similarly, the policy is robust to the choice of discretization granularity. As shown in Figure 15, when the number of discretization intervals exceeds 8, performance does not change much with further refinement, indicating insensitivity to this hyperparameter. This robustness can be attributed to the fact that low-rank TT models provide smooth function approximations, which support effective interpolation between discretized points (as discussed in Section 3.3). While extremely coarse discretization may degrade performance, in practice, setting the number of intervals to 10 or 20 generally yields stable and accurate results.

For the influence of the bandwidth for the uniform distribution used in probabilistic domain adaptation, we refer the readers to Section 6.2 for details. In summary, a coarse bandwidth ($w = N/5$, where N is the number of discretization points) is sufficient for closed-loop tasks such as `Push` and `Reorientation`. In contrast, `Hit`, as a one-shot (open-loop) task, benefits from a more precise distribution, with $w = N/20$ being adequate. In general, the bandwidth parameter for other contact-rich tasks can be determined similarly based on whether the task is open-loop or closed-loop, without requiring significant tuning effort.

To further demonstrate the efficiency of the TT representation, we compared it against non-TT function approximation methods, particularly Neural Networks (NN). In our framework, both TT and NN can be used to learn parameter-augmented base policies and to approximate parameter-augmented advantage functions. The key question is which method offers greater efficiency in the subsequent steps: computing the parameter-conditioned advantage function and retrieving the corresponding policy via an $\arg\max$ operation.

Table 2 highlights the significant computational advantage of TT in computing the advantage function. Specifically, TT models support efficient *core-level* operations, which avoid combinatorial computation across the multi-dimensional parameter space, leading to a complexity of $\mathcal{O}(Ndr^2)$, where N is the number of discretization points per dimension, d is the dimensionality of the parameter space, and r is the TT rank, which typically remains small in practice,

Table 3. Comparison of TT and NN representation for parameter-conditioned policy retrieval. Error refers to the task execution error using the retrieved policy, while Opt. Time indicates the time required to perform the $\arg\max$ operation.

	TT Optimization		TT Refinement		NN Optimization	
	Task Error	Opt. Time (s)	Task Error	Opt. Time (s)	Task Error	Opt. Time (s)
Hit	0.010 ± 0.008	0.003 ± 0.001	0.002 ± 0.001	0.015 ± 0.001	0.002 ± 0.004	0.346 ± 0.021
Push	0.034 ± 0.024	0.013 ± 0.001	0.033 ± 0.024	0.146 ± 0.001	1.469 ± 0.574	2.020 ± 0.002
Reorientation	0.061 ± 0.104	0.002 ± 0.001	0.061 ± 0.104	0.062 ± 0.001	0.751 ± 0.680	1.514 ± 0.092

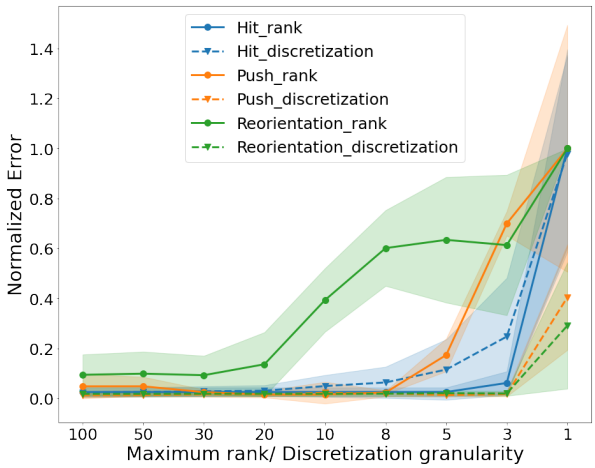


Figure 15. Sensitivity analysis of policy performance with respect to the maximum TT-rank (solid lines) and discretization granularity (dashed lines). The y-axis shows the normalized ℓ_2 error between the final and target states, scaled by the worst-case performance in each task. The x-axis denotes the maximum TT-rank and the number of discretization intervals (non-uniformly spaced to enhance readability in the low-rank and coarse-discretization regions). The results demonstrate that policy performance is not highly sensitive to these hyperparameters.

as demonstrated in our experiments. In contrast, NN-based methods (or any non-TT models) rely on *function-level* operations, resulting in a complexity of $\mathcal{O}(N^d)$, which makes the computation substantially slower.

After computing the parameter-conditioned advantage function, the next step is policy retrieval via an $\arg\max$ operation. As shown in Table 3, TT again demonstrates superior efficiency. Direct TT-based optimization (TT Optimization) yields low-error policies with minimal computation time, achieving near-global solutions within milliseconds. In contrast, NN-based approaches (NN Optimization) typically rely on gradient-based solvers. In this work, we specifically adopt the Adam optimizer (Kingma and Ba 2015) as a representative method for comparison. The results indicate that this approach is slower and more susceptible to suboptimal convergence, particularly in complex tasks such as `Push` and `Reorientation`.

To further improve accuracy, we also evaluated a hybrid approach (TT Refinement), which uses TT optimization for initialization, followed by a gradient-based Newton-type method. While this refinement slightly improves performance, the gain is often marginal compared to the additional computational cost. In the tasks considered in this work, TT optimization alone is sufficient to achieve high-quality results. These findings highlight the strong potential of TT representation for efficient and robust policy learning in complex, contact-rich manipulation tasks.

6.6 Real-robot experiments: planar push

We validated the proposed method for the planar pushing task using a 7-axis Franka robot and a RealSense D435 camera. The manipulated objects included a sugar box and a bleach cleanser from the YCB dataset (Calli et al. 2015), each with different shapes and masses, as shown in Fig. 1. The friction coefficients between the objects and the table were varied by using a metal surface and plywood, respectively. Note that it is easier to control the robot kinematically rather than using force control. We therefore leveraged the ellipsoidal limit surface to convert the applied force to velocity, resulting in the motion equations shown in (Hogan and Rodriguez 2020b; Xue et al. 2023). We trained a parameter-augmented policy in simulation and then applied it in the real world through domain contraction. The experiments demonstrate the effectiveness of the obtained parameter-conditioned policies in manipulating instances with diverse parameters. Additionally, external disturbances were introduced by humans, showcasing the reactivity of the retrieved policy. Figure 14 presents keyframes of the learned planar pushing primitive applied to a bleach bottle on a metal surface, exemplifying a specific case within a variety of diverse pushing scenarios. Further results are demonstrated in the accompanying video.

7 Conclusion and Future Work

In this article, we propose *implicit motor adaptation* for robust manipulation primitive learning, which enables parameter-conditioned policy retrieval given a probabilistic parameter distribution rather than a single estimate. We prove that, to achieve this, the policy must be represented implicitly as the $\arg\max$ of the advantage function, rather than treated as an explicit feed-forward function. The state-value and advantage functions are represented in tensor train (TT) format, facilitating efficient policy retrieval through operations at the tensor core level instead of at the function level. Our approach differs from the well-known *explicit motor adaptation* framework, which either requires precise system identification or additional student policy training. Theoretical analysis and numerical results demonstrate the superior performance of *implicit motor adaptation* over *explicit motor adaptation*. Simulation and real-world experiments further validate the effectiveness of our approach in contact-rich manipulation tasks under parameter uncertainty and external disturbances.

Our work paves the way for robust sim-to-real transfer. It can be easily integrated with more general policy learning approaches, such as reinforcement learning methods

and classical approaches (e.g., LQR), provided the base policy can be implicitly represented with a state-action value function. The key insight is to represent the base policy implicitly and retrieve the parameter-conditioned policy probabilistically. This concept also holds promise for achieving robust behavior cloning, where the policy can be represented through energy functions (as in implicit behavior cloning (Florence et al. 2022)) or denoising functions (as in diffusion policy (Chi et al. 2024)).

In this article, we used TT to represent the value function and the advantage function, both obtained via TT-Cross approximation (Oseledets and Tyrtyshnikov 2010; Usvyatsov et al. 2022). Although TT-Cross enables efficient global approximation by actively querying function values, it struggles to scale to very high-dimensional settings. In the future, we plan to combine TT with neural networks in a data-driven manner (Dolgov et al. 2023; Shetty 2024) to address this limitation.

Additionally, we employed a simple MLP for system identification, assuming that the parameters follow a uniform distribution. This approach could be extended by adopting more advanced models, such as diffusion models (Ho et al. 2020). Large Visual-Language Models (Gao et al. 2024) could also be explored to leverage their commonsense understanding of domain knowledge from scenario images.

Acknowledgments

This work was supported by the China Scholarship Council (grant No.202106230104), and by the State Secretariat for Education, Research and Innovation in Switzerland for participation in the European Commission's Horizon Europe Program through the INTELLIMAN project (https://intelliman-project.eu/, HORIZON-CL4-Digital-Emerging Grant 101070136) and the SESTOSENSO project (http://sestosenso.eu/, HORIZON-CL4-Digital-Emerging Grant 101070310). We thank Jiacheng Qiu for suggestions about the implementation of RL baselines.

References

Veronica Adetola and Martin Guay. Robust adaptive mpc for constrained uncertain nonlinear systems. *International Journal of Adaptive Control and Signal Processing*, 25(2):155–167, 2011.

Veronica Adetola, Darryl DeHaan, and Martin Guay. Adaptive model predictive control for constrained nonlinear systems. *Systems & Control Letters*, 58(5):320–326, 2009.

Karol Arndt, Murtaza Hazara, Ali Ghadirzadeh, and Ville Kyrki. Meta reinforcement learning for sim-to-real domain adaptation. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 2725–2731, 2020.

Bibit Bianchini, Mathew Halm, and Michael Posa. Simultaneous learning of contact and continuous dynamics. In *Conference on Robot Learning*, pages 3966–3978. PMLR, 2023.

Aude Billard, Sylvain Calinon, Ruedige Dillmann, and Stefan Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*, pages 1371–1394. Springer, Secaucus, NJ, USA, 2008.

Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 4243–4250, 2018.

Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The YCB object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 8973–8979, 2019.

Shuo Cheng and Danfei Xu. League: Guided skill learning and abstraction for long-horizon manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 8(10):6451–6458, 2023.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *International Journal of Robotics Research (IJRR)*, 0(0), 2024.

Lin Cong, Hongzhuo Liang, Philipp Ruppel, Yunlei Shi, Michael Görner, Norman Hendrich, and Jianwei Zhang. Reinforcement learning with vision-proprioception model for robot planar pushing. *Frontiers in Neurorobotics*, 16:829437, 2022.

Sergey Dolgov, Dante Kalise, and Luca Saluzzi. Data-driven tensor train gradient cross approximation for hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 45 (5):A2153–A2184, 2023.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Chebotar Yevgen, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model. In *International Conference on Machine Learning*, pages 8469–8488. PMLR, 2023.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 32, 2019.

Christopher Edwards and Sarah Spurgeon. *Sliding mode control: theory and applications*. CRC Press, 1998.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

Ana Lùcia D Franco, Henri Bourles, Edson R De Pieri, and Herve Guillard. Robust nonlinear control associating robust feedback linearization and H_∞ control. *IEEE Transactions on Automatic Control*, 51(7):1200–1207, 2006.

Jensen Gao, Bidipta Sarkar, Fei Xia, Ted Xiao, Jiajun Wu, Brian Ichter, Anirudha Majumdar, and Dorsa Sadigh. Physically grounded vision-language models for robotic manipulation. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 12462–12469. IEEE, 2024.

Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*,

25(3):335–348, 1989.

Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins, Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, David E Smith, Ying Sun, and Daniel Weld. PDDL|The Planning Domain Definition Language. Technical report, Yale Center for Computational Vision and Control, 1998.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Francois R Hogan and Alberto Rodriguez. Reactive planar non-prehensile manipulation with hybrid model predictive control. *International Journal of Robotics Research (IJRR)*, 39(7):755–773, 2020a.

François Robert Hogan and Alberto Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pages 800–815. Springer, 2020b.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. Intl Conf. on Learning Representations (ICLR)*, 2015.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Proc. Robotics: Science and Systems (RSS)*, 2021.

P Lancaster. *Algebraic Riccati Equations*. Oxford Science Publications/The Clarendon Press, Oxford University Press, 1995.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020a.

Michelle A Lee, Yuke Zhu, Peter Zachares, Matthew Tan, Krishnan Srinivasan, Silvio Savarese, Li Fei-Fei, Animesh Garg, and Jeannette Bohg. Making sense of vision and touch: Learning multimodal representations for contact-rich tasks. *IEEE Transactions on Robotics*, 36(3):582–596, 2020b.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. In *Proc. Intl Joint Conf. on Artificial Intelligence (IJCAI)*, pages 5502–5511, 2022.

Brett T Lopez, Jean-Jacques E Slotine, and Jonathan P How. Dynamic tube mpc for nonlinear systems. In *2019 American Control Conference (ACC)*, pages 1655–1662. IEEE, 2019.

Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *International Journal of Robotics Research (IJRR)*, 15(6):533–556, 1996.

Matthew T Mason. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research (IJRR)*, 5(3):53–71, 1986.

Matthew T Mason. Progress in nonprehensile manipulation. *International Journal of Robotics Research (IJRR)*, 18(11): 1129–1141, 1999.

Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & chemical engineering*, 23(4-5):667–682, 1999.

Fabio Muratore, Felix Treede, Michael Gienger, and Jan Peters. Domain randomization for simulation-based policy optimization with transferability assessment. In *Conference on Robot Learning*, pages 700–713. PMLR, 2018.

Fabio Muratore, Theo Gruner, Florian Wiese, Boris Belousov, Michael Gienger, and Jan Peters. Neural posterior domain randomization. In *Conference on Robot Learning*, pages 1532–1542. PMLR, 2022.

Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9):538–550, 2019. ISSN 2522-5820.

Ivan Oseledets and Eugene Tyrtyshnikov. TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88, 2010.

Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Elena Pashenkova, Irina Rish, and Rina Dechter. Value iteration and policy iteration algorithms for Markov decision problem. In *AAAI'96: Workshop on Structural Issues in Planning and Temporal Reasoning*, volume 39. Citeseer, 1996.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 3803–3810, 2018.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pages 188–204. PMLR, 2021.

Susan Potters. Learning-based control for pushing with a non-holonomic mobile robot. Master's thesis, Delft University of Technology, Delft, 2022.

Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. In *Proc. Robotics: Science and Systems (RSS)*, 2019.

Dmitry V. Savostyanov and Ivan V. Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. *The 2011 International Workshop on Multidimensional (nD) Systems*, pages 1–8, 2011.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

Suhan Shetty, Teguh Lembono, Tobias Löw, and Sylvain Calinon. Tensor train for global optimization problems in robotics. *International Journal of Robotics Research (IJRR)*, 43(6):811–839, 2024a. doi: 10.1177/02783649231217527.

Suhan Shetty, Teng Xue, and Sylvain Calinon. Generalized policy iteration using tensor approximation for hybrid control. In *Proc. Intl Conf. on Learning Representations (ICLR)*, 2024b.

Suhan Narayana Shetty. *Robot Learning using Tensor Networks*. PhD thesis, EPFL, Lausanne, 2024.

Yuri Shtessel, Christopher Edwards, Leonid Fridman, and Arie Levant. *Sliding mode control and observation*, volume 10. Springer, 2014.

Konstantin Sozykin, Andrei Chertkov, Roman Schutski, Anh-Huy Phan, Andrzej S Cichocki, and Ivan Oseledets. Ttopt: A maximum volume quantized tensor train-based optimization and its application to reinforcement learning. *Advances in neural information processing systems*, 35:26052–26065, 2022.

Edwin Stoudenmire and David J Schwab. Supervised learning with tensor networks. *Advances in neural information processing systems*, 29, 2016.

Ezra Tal, Alex Gorodetsky, and Sertac Karaman. Continuous tensor train-based dynamic programming for high-dimensional zero-sum differential games. In *2018 Annual American Control Conference (ACC)*, pages 6086–6093. IEEE, 2018.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proc. Intl Joint Conf. on Artificial Intelligence (IJCAI)*, pages 4950–4957, 2018.

Marc Toussaint, Jung-Su Ha, and Danny Driess. Describing physics for physical reasoning: Force-based sequential manipulation planning. *IEEE Robotics and Automation Letters (RA-L)*, 5(4): 6209–6216, 2020.

Marc Toussaint, Jason Harris, Jung-Su Ha, Danny Driess, and Wolfgang Hönig. Sequence-of-constraints mpc: Reactive timing-optimal control of sequential manipulation. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 13753–13760, 2022.

Mikhail Usvyatsov, Rafael Ballester-Ripoll, and Konrad Schindler. tntorch: Tensor network learning with pytorch. *Journal of Machine Learning Research*, 23(208):1–6, 2022.

Francisco Viña Barrientos, Yiannis Karayiannidis, Christian Smith, and Danica Kragic. Adaptive control for pivoting with visual and tactile feedback. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, 2016.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *Proc. Intl Conf. on Learning Representations (ICLR)*, 2022.

Paul Werbos. Approximate dynamic programming for real-time control and neural modeling. *Handbook of intelligent control*, 1992.

Teng Xue, Hakan Girgin, Teguh Santoso Lembono, and Sylvain Calinon. Demonstration-guided optimal control for long-term non-prehensile planar manipulation. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 4999–5005, 2023.

Teng Xue, Amirreza Razmjoo, Suhan Shetty, and Sylvain Calinon. Robust manipulation primitive learning via domain contraction. In *Proc. Conference on Robot Learning (CoRL)*, 2024a.

Teng Xue, Amirreza Razmjoo, Suhan Shetty, and Sylvain Calinon. Logic-Skill Programming: An Optimization-based Approach to Sequential Skill Planning. In *Proc. Robotics: Science and Systems (RSS)*, 2024b.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Proc. Robotics: Science and Systems (RSS)*, 2017.