# Zero-Shot Sim-to-Real Robot Learning: A Dexterous Manipulation Study on Reactive Catching

Kejia Ren[1], Gaotian Wang[1], Andrew S. Morgan[2] and Kaiyu Hang[1]
[1]Department of Computer Science, Rice University
[2]Robotics and AI Institute

***Abstract*—Dexterous manipulation is physics-intensive and highly sensitive to modeling errors and perception noise, making sim-to-real transfer prohibitively challenging. Domain randomization (DR) is commonly used to improve the robustness of learned policies for such tasks, but conventional DR randomizes one instance per episode, offering very limited exposure to the variability of real-world dynamics. To this end, we propose Domain-Randomized Instance Set (DRIS), which represents and propagates a set of randomized instances simultaneously, providing richer approximation of uncertain dynamics and enabling policies to learn actions that account for multiple possible outcomes. Supported by theoretical analysis, we show that DRIS yields more robust policies and alleviates the need for real-world fine-tuning, even with a modest number of instances (e.g., 10). We demonstrate this on a challenging reactive catching task. Unlike traditional catching setups that use end-effectors designed to mechanically stabilize the object (e.g., curved or enclosing surfaces), our system uses a flat plate that offers no passive stabilization, making the task highly sensitive to noise and requiring rapid reactive motions. The learned policies exhibit strong robustness to uncertainties and achieve reliable zero-shot sim-to-real transfer.**

## I. Introduction

Dexterous manipulation refers to the robot's ability to skillfully manipulate objects through coordinated and intentional contact interactions, e.g., in-hand manipulation [4, 10], tool use [25], pushing [13], catching [24], etc. Such tasks often require well-planned contact, timing, and motion to achieve stable and adaptive control over object behavior. Moreover, they often introduce a substantial sim-to-real gap, as the success of such tasks can be highly sensitive to inaccuracies in estimating the underlying dynamics and physical properties such as contact geometry, frictions, object inertia, and compliance. Even small discrepancies between simulation and the real world can lead to large deviations in manipulation behavior and ultimately cause task failure.

Traditional model-based frameworks depend on known system parameters to achieve accurate prediction and control [8]. However, in practical manipulation scenarios, properties such as contact stiffness, friction coefficients, and object geometries can greatly vary between environments and are difficult to precisely identify. Learning-based methods address these limitations by training policies with extensive domain randomization or data augmentation over uncertain physical parameters in simulation [32], improving the robustness to these uncertainties when deployed in the real world. Yet, tasks involving very dynamic, contact-rich interactions remain difficult to learn, as their success hinges on precise coordination of contact timing and force modulation under uncertainty.

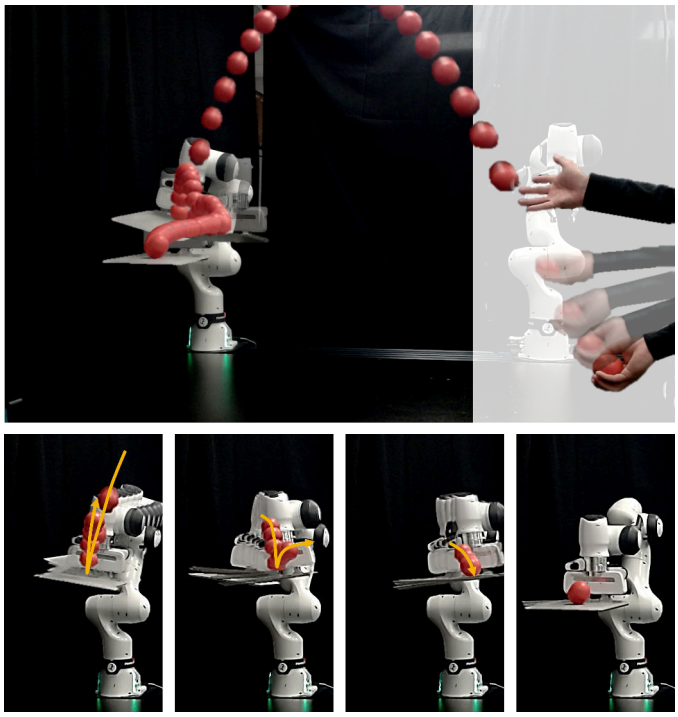


Fig. 1: Robot reactive catching of a rubber ball using a flat plate. The second row shows sequential impacts during the catching motion, where the final frame depicts the ball successfully stabilized on the plate; the yellow arrows indicate the ball's trajectory at the moments of impact.

Domain randomization is traditionally performed by perturbing relevant physical parameters during data collection, e.g., sampling one object instance with randomized properties in each rollout and training policy on the collected rollouts. While this exposes the policy to diverse experiences, it lacks a structured mechanism for the policy to reason about how uncertainties affect possible manipulation outcomes. In contrast, if such uncertainties are explicitly incorporated into the task representation itself, the policy can learn to generate actions that compensate for them and plan with respect to a distribution of possible outcomes, thereby mitigating the sim-to-real gap introduced by these uncertainties.

To this end, rather than training a policy on individual

manipulation instances with randomly sampled physical parameters (as is done in conventional domain randomization), we propose *Domain-Randomized Instance Set (DRIS)* that explicitly models a distribution of possible physical variations encountered during manipulation. Our proposed DRIS contains multiple parallel instances (e.g., objects with different physical properties), all of which are propagated simultaneously. Their state evolutions under a shared action jointly inform the policy update. As a result, the learned policy becomes inherently tolerant to the uncertainties encoded in DRIS, leading to improved robustness against the sim-to-real gap (e.g., enabling reliable zero-shot transfer).

To demonstrate the effectiveness of the proposed learning strategy, we consider a highly challenging reactive catching task illustrated in Fig. 1. Unlike traditional catching setups that employ curved or concave end-effectors such as cups [14], nets [16], or articulated hands [6], to mechanically stabilize the object after impact, our setup uses a flat and low-friction plate rigidly attached to the robot's end-effector. As a result, the system becomes highly sensitive to physical uncertainties or perception errors, where even small errors in contact timing, plate orientation, or unwanted lateral forces can cause the ball to bounce off or roll away from the plate.

In this work, we propose a robust learning strategy for dynamic manipulation, demonstrated through a challenging robot reactive catching case study. Our key technical contributions are summarized as follows:

1) We propose Domain-Randomized Instance Set (DRIS) as a representation to capture the uncertainty in system dynamics induced by physical parameter variations;
2) We propose a DRIS-based data collection and policy learning paradigm that enables simultaneous handling of multiple task instances, yielding more stable and robust policy learning and improved zero-shot sim-to-real transfer than the traditional paradigm, supported by theoretical analysis;
3) We instantiate the framework on a challenging reactive catching problem with a tailored task representation and policy architecture, and validate our framework through both simulation and real-world experiments.

## II. Related Work

*Reinforcement Learning for Dexterous Manipulation.* Unlike traditional analysis-based approaches sensitive to modeling errors, Reinforcement Learning (RL) learns directly from data, making it advantageous for the complex contact dynamics of dexterous manipulation. RL has successfully solved tasks including in-hand manipulation [4, 10], catching [1], juggling [35], deformable object manipulation [11], and regrasping [34]. However, these data-intensive methods typically require large-scale high-fidelity simulation or expert demonstrations to facilitate real-world transfer.

*Sim-to-Real Transfer in Robotics.* Bridging the sim-to-real gap caused by sensing and contact discrepancies is a central challenge. Beyond Domain Randomization (DR), researchers utilize high-fidelity simulators [47, 39, 26] or explicitly bridge the gap via adaptive system identification [50, 22, 42] and hybrid dynamics learning [3, 23]. While these techniques enable zero-shot transfer [45], reliable deployment for dynamic manipulation remains difficult due to the discontinuous, stochastic nature of high-speed contact dynamics.

*Domain Randomization in Robot Learning.* Since real-world data collection is costly, DR is essential for reducing the simulation gap in contact-rich tasks [29, 20]. Randomization strategies have evolved from visual and geometric properties [45, 46] to kinematic [15] and dynamic parameters [32]. While active DR optimizes sampling distributions to improve transfer [37, 9, 27, 28, 44], it typically requires real-world rollouts. In this work, we instead randomize multiple instances simultaneously and train over their joint evolution to achieve zero-shot transfer for challenging tasks such as catching.

*Representing Uncertainty in States and Dynamics.* Robotics has extensively studied uncertainty representation. Classical methods employ belief-space planning via POMDPs [36, 7, 21]. Learning-based approaches incorporate uncertainty through probabilistic latent dynamics [17, 18], history-based parameter inference [30, 41], or ensembles for model-based RL [12]. Complementary to these, recent work learns belief embeddings to explicitly encode uncertainty for policy conditioning in partially observable settings [48].

*Dexterous Robot Catching.* Catching typically requires fast prediction and reactive motion. Prior work evolved from optimization-based planning [6] and probabilistic models for uneven objects [24] to deep learning-based policies on mobile manipulators [51]. However, most rely on mechanically assisting end-effectors like cups [31, 14, 1], nets [16], or articulated hands [38, 19]. In contrast, we use a flat plate without mechanical stabilization. Unlike similar flat-surface work [5] limited to specific objects and a number of rebounds, our approach handles diverse conditions without such restrictions.

## III. Learning Manipulation Policy with Domain-Randomized Instance Set

### A. Dexterous Manipulation as Policy Learning

We consider the dexterous manipulation problem in which a robotic manipulator interacts with an object through contact-rich dynamics. We define the domain space $\mathcal{C}$ as the set of all relevant physical parameters that influence the system dynamics, such as robot characteristics (e.g., stiffness), object geometry and physical properties (e.g., friction). Each $\boldsymbol{c} \in \mathcal{C}$ represents a specific domain instance that induces a particular realization of the system dynamics. These parameters are often difficult to measure accurately, but they can substantially affect the object's motion during manipulation.

The problem is formulated as a discrete-time Markov Decision Process (MDP). At time step $t$, the system observes a state $\boldsymbol{s}_t \in \mathcal{S}$, executes an action $\boldsymbol{a}_t = \pi(\boldsymbol{s}_t) \in \mathcal{A}$ according to a robot policy $\pi(\cdot)$, and transitions to the next state $\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{c})$, where the dynamics $f$ depend on the domain parameter $\boldsymbol{c}$. Different values of $\boldsymbol{c}$ (e.g., object properties) lead to distinct motion behaviors even under the

same robot action. At each step, the system receives a scalar reward $r_t = r(\boldsymbol{s}_t, \boldsymbol{a}_t) \in \mathbb{R}$.

The objective is to find an optimal policy $\pi_{\boldsymbol{\theta}} : \mathcal{S} \mapsto \mathcal{A}$, that maximizes the expected return (i.e., cumulative reward), where $\gamma \in [0, 1)$ is a discount factor:

$$\begin{aligned} \max_{\pi_{\boldsymbol{\theta}}} \quad & \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{T-1} \gamma^t \, r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right] \\ \text{s.t.} \quad & \boldsymbol{a}_t = \pi_{\boldsymbol{\theta}}(\boldsymbol{s}_t) \in \mathcal{A}, \quad \forall t = 0, \cdots, T-1, \\ & \boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{c}) \in \mathcal{S}, \quad \boldsymbol{c} \in \mathcal{C} \end{aligned} \tag{1}$$

In deep reinforcement learning (DRL), the policy $\pi_{\boldsymbol{\theta}}$ is modeled by a neural network parameterized by $\boldsymbol{\theta}$, and optimized via gradient-based update using data collected from robot-object interactions.

### B. Domain-Randomized Instance Set (DRIS)

In practice, physical parameters $\boldsymbol{c}$ are difficult to identify accurately, introducing uncertainty into manipulation dynamics. To improve robustness, most approaches model the dynamics stochastically as a probability $P(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ and apply domain randomization to approximately capture this stochasticity, typically by sampling a single parameter instance per episode in standard model-free RL [32]. More recent methods further adapt the sampling distribution using real-world data [37, 44]. However, these strategies still predict only individual next states, and do not explicitly model how the distribution of possible states evolves over time.

Inspired by prior work on set-based representations [49], rather than propagating a single state $\boldsymbol{s}_t$, we instead propagate a set of possible states $\mathcal{S}_t \subset \mathcal{S}$ simultaneously under a shared robot action, reflecting the effects of physical variations across different domain parameters. Specifically, we construct a discrete set $\hat{\mathcal{C}} \subset \mathcal{C}$ of $N$ manipulation instances (indexed by $i$) by uniformly sampling from the domain space:

$$\hat{\mathcal{C}} = \left\{\boldsymbol{c}^{(i)}\right\}_{i=1}^{N}, \; \boldsymbol{c}^{(i)} \sim \mathcal{U}(\mathcal{C}) \tag{2}$$

We then define the *Domain-Randomized Instance Set (DRIS)* as the collection of state-parameter pairs associated with these sampled instances:

$$\mathcal{D}_t = \left\{\left(\boldsymbol{s}_t^{(i)}, \boldsymbol{c}^{(i)}\right) \,\middle|\, \boldsymbol{c}^{(i)} \in \hat{\mathcal{C}}\right\}_{i=1}^{N} \subset \mathcal{S} \times \mathcal{C} \tag{3}$$

where $\boldsymbol{s}_t^{(i)}$ in each element denotes the state of the $i$-th manipulation instance, as it evolves at time $t$ under its corresponding domain parameter $\boldsymbol{c}^{(i)}$. For clarity, we also extract the state of all instances in the DRIS to form another set (i.e., the projection of $\mathcal{D}_t$ onto the state space $\mathcal{S}$):

$$\mathcal{S}_t = \text{proj}_{\mathcal{S}}(\mathcal{D}_t) = \left\{\boldsymbol{s}_t^{(i)} \,\middle|\, \left(\boldsymbol{s}_t^{(i)}, \boldsymbol{c}^{(i)}\right) \in \mathcal{D}_t\right\} \subset \mathcal{S} \tag{4}$$

We extend the system dynamics by introducing a new function $\mathcal{F}$ to evolve the entire instance set:

$$\begin{aligned} \mathcal{D}_{t+1} &= \left\{\left(\boldsymbol{s}_{t+1}^{(i)}, \boldsymbol{c}^{(i)}\right)\right\}_{i=1}^{N} = \mathcal{F}(\mathcal{D}_t, \boldsymbol{a}_t) \\ &= \left\{\left(f\left(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t, \boldsymbol{c}^{(i)}\right), \boldsymbol{c}^{(i)}\right) \,\middle|\, \left(\boldsymbol{s}_t^{(i)}, \boldsymbol{c}^{(i)}\right) \in \mathcal{D}_t\right\} \end{aligned} \tag{5}$$

### C. Zero-Shot Sim-to-Real Learning

Since DRIS defined in Sec. III-B inherently embeds uncertainties into its representation, integrating DRIS into policy learning in simulation enables more robust sim-to-real transfer without fine-tuning. Because the DRIS size $N$ may vary and because the robot will manipulate only a single instance (although trained with multiple instances in DRIS) in real deployment, the policy must take an input whose dimensionality is independent of $N$. Thereafter, we require a compact and size-agnostic representation of DRIS. To this end, we employ an encoder $\psi(\cdot)$ that maps the DRIS state (i.e., $\mathcal{S}_t$) to a fixed-dimensional latent vector $\boldsymbol{z}_t = \psi(\mathcal{S}_t) \in \mathbb{R}^{d_z}$, where $d_z$ is the latent dimension and remains constant regardless of the size $N$ of $\mathcal{D}_t$. A policy that directly operates in this latent space is then learned by solving:

$$\begin{aligned} \max_{\pi_{\boldsymbol{\theta}}} \quad & \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\sum_{t=0}^{T-1} \gamma^t \cdot \frac{1}{N} \sum_{\boldsymbol{s}_t^{(i)} \in \mathcal{S}_t} r(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t)\right] \\ \text{s.t.} \quad & \boldsymbol{a}_t = \pi_{\boldsymbol{\theta}}(\boldsymbol{z}_t) \in \mathcal{A}, \quad \forall t = 0, \cdots, T-1, \\ & \boldsymbol{z}_t = \psi(\mathcal{S}_t) \in \mathbb{R}^{d_z}, \quad \forall t = 0, \cdots, T-1, \\ & \mathcal{S}_t = \text{proj}_{\mathcal{S}}(\mathcal{D}_t), \quad \forall t = 0, \cdots, T-1, \\ & \mathcal{D}_{t+1} = \mathcal{F}(\mathcal{D}_t, \boldsymbol{a}_t) \subset \mathcal{S} \times \mathcal{C} \end{aligned} \tag{6}$$

Compared to conventional DR, we summarize the benefits of DRIS in Theorem III.1:

**Theorem III.1** (DRIS improves stability and robustness)**.** *Relative to conventional domain randomization, DRIS is an exact particle approximation for the system belief propagation. Consequently, DRIS yields a more stable optimization, more robust policies, and improved sim-to-real generalization. (See Appendix B for detailed analysis and proofs.)*

As illustrated in Fig. 2 (left) with a catching task, DRIS can typically be implemented by a set of multiple object instances, each initialized with different physical properties (domain parameters) such as shape, weight, and friction. The instances evolve independently, i.e., they do not interact with one another, and their respective state transitions under the same robot action are evaluated in parallel for policy update.

After training, the policy can be deployed in the real world without fine-tuning: The observed single real-world state $\boldsymbol{s}$ (equivalent to a DRIS of size $N = 1$) is encoded and provided as input to the trained policy, repeatedly until the task is finished (see Appendix A for detailed algorithmic pipeline).

## IV. Reactive Catching Case Study

We instantiate the proposed DRIS-based learning strategy on the problem of reactive ball catching with an $M$-DoF robot manipulator. As illustrated in Fig. 1, the robot is equipped with a flat plate rigidly attached to its end-effector and is required to catch a ball thrown toward it. To successfully perform this task, the robot must approach and make contact with the incoming ball, absorb its kinetic energy through controlled motions, and subsequently stabilize and balance the ball once it comes to rest on the plate.

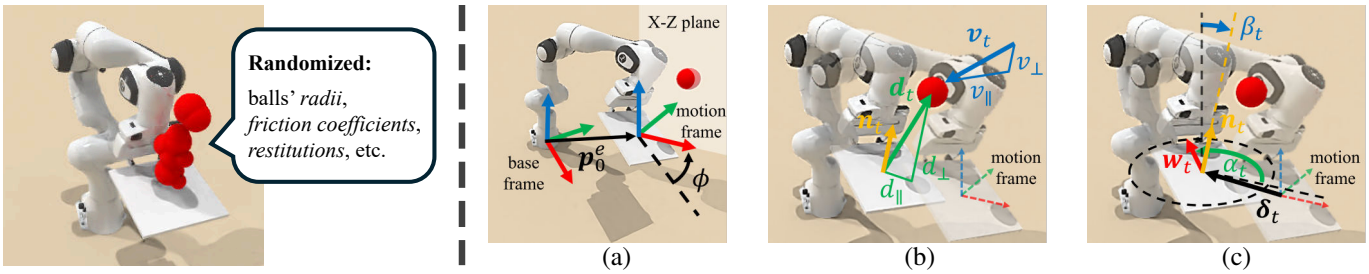



Fig. 2: The reactive catching problem instantiation. *Left:* Randomized parameters in DRIS. *Right:* Components used to represent the problem: (a) The motion frame is defined at the plate's initial position $\boldsymbol{p}_0^e$ and rotated about the Z-axis by angle $\phi$ so that its X-Z plane passes through the incoming ball. (b) The robot has moved from the initial ghosted to the current opaque configuration. The system state consists of the ball's relative displacement $\boldsymbol{d}_t$ (green arrow) to the plate and its linear velocity $\boldsymbol{v}_t$ (blue arrow), both expressed in the motion frame; these two quantities are decomposed with respect to the plate's normal vector $\boldsymbol{n}_t$ (yellow arrow) to define the reward. (c) The action comprises a translation of the plate $\boldsymbol{\delta}_t$ (black arrow) and a tilting configuration: angle $\alpha_t$ (green) specifies a horizontal axis $\boldsymbol{w}_t$ (red arrow), about which the plate is rotated by angle $\beta_t$ (blue).

### A. Problem Representation

To enable a more compact problem representation and ensure the catching policy is invariant to the ball's incoming direction, we define a fixed motion frame relative to the ball at $t = 0$ (the beginning of manipulation). All task-related quantities (e.g., state and action) are expressed in this motion frame. As shown in Fig. 2 (right-a), the motion frame is constructed with its origin at the plate's center, its $Z$-axis aligned vertically upward, and its $X$-$Z$ plane passing through the ball's center. The transformation from the robot's base frame to the motion frame is denoted by ${}^b\boldsymbol{T}_m = (\boldsymbol{R}_z(\phi), \boldsymbol{p}_0^e) \in SE(3)$, where $\boldsymbol{R}_z(\phi) \in SO(3)$ denotes a rotation about the $Z$-axis by angle $\phi$ and $\boldsymbol{p}_0^e \in \mathbb{R}^3$ represents the plate's center position at $t = 0$, expressed in the robot's base frame.

**State.** The state of the system consists of the ball's displacement relative to the plate's center and its linear velocity, both expressed in the motion frame. Specifically, as shown in Fig. 2 (right-b), the state is denoted as $\boldsymbol{s}_t = (\boldsymbol{d}_t, \boldsymbol{v}_t) \in \mathbb{R}^6$, where $\boldsymbol{d}_t = \boldsymbol{R}_z(\phi)^\top (\boldsymbol{p}_t^o - \boldsymbol{p}_t^e) \in \mathbb{R}^3$ is the displacement of the ball (with position $\boldsymbol{p}_t^o$) relative to the plate's center (with position $\boldsymbol{p}_t^e$) at time $t$, expressed in the motion frame; $\boldsymbol{v}_t = \boldsymbol{R}_z(\phi)^\top \boldsymbol{v}_t^o \in \mathbb{R}^3$ is the ball's linear velocity $\boldsymbol{v}_t^o$ transformed into the motion frame.

**Action.** The action is defined as $\boldsymbol{a}_t = (\boldsymbol{\delta}_t, \boldsymbol{u}_t) \in \mathbb{R}^5$, where $\boldsymbol{\delta}_t \in \mathbb{R}^3$ denotes the displacement command of the plate's center expressed in the motion frame, i.e., the corresponding target position of the plate's center in the robot's base frame is computed as $\tilde{\boldsymbol{p}}_t^e = \boldsymbol{R}_z(\phi)\,\boldsymbol{\delta}_t + \boldsymbol{p}_t^e$. The remaining action component $\boldsymbol{u}_t = (\alpha_t, \beta_t) \in \mathbb{R}^2$ represents the target tilting configuration of the plate in 3D. Specifically, the plate is rotated about a horizontal axis $\boldsymbol{w}_t = (\cos\alpha_t, \sin\alpha_t, 0)^\top$ by an angle $\beta_t$, where $\alpha_t \in [0, 2\pi)$ and $\beta_t \in [0, \pi/4)$. This rotation tilts the plate's normal vector $\boldsymbol{n}_t \in \mathbb{R}^3$ as illustrated in Fig. 2 (right-c). During execution, the complete action $\boldsymbol{a}_t$ is converted into a desired plate pose, which is then used to command the robot via joint torque control, as will be detailed in Sec. IV-D.

**Reward.** Given the observed state $\boldsymbol{s}_t = (\boldsymbol{d}_t, \boldsymbol{v}_t)$ and the plate normal $\boldsymbol{n}_t \in \mathbb{R}^3$ with $\|\boldsymbol{n}_t\| = 1$ (all in the motion frame), we decompose the ball's displacement and velocity into components perpendicular and parallel to the plate surface $d_\perp, d_\parallel, v_\perp, v_\parallel \in \mathbb{R}$, as in Fig. 2 (right-b):

$$\begin{aligned} d_\perp &= \boldsymbol{d}_t \cdot \boldsymbol{n}_t, \quad d_\parallel = \|\boldsymbol{d}_t - d_\perp \cdot \boldsymbol{n}_t\| \\ v_\perp &= \boldsymbol{v}_t \cdot \boldsymbol{n}_t, \quad v_\parallel = \|\boldsymbol{v}_t - v_\perp \cdot \boldsymbol{n}_t\| \end{aligned} \tag{7}$$

The reward is the sum of a velocity term $r_v \in (0, 1]$ (which yields a higher value when the ball's velocity is kept lower) and a constant penalty term $r_p = -1$ applied when the ball moves beneath or outside the plate:

$$\begin{aligned} r_t &= r(\boldsymbol{s}_t, \boldsymbol{a}_t) = r_v + r_p, \\ r_v &= \frac{1}{2}\exp\left(-\frac{v_\parallel^2}{\eta^2}\right) + \frac{1}{2}\exp\left(-\frac{\max\{v_\perp, -0.1\}^2}{\eta^2}\right), \\ r_p &= -\mathbb{1}\left\{d_\perp < 0 \vee d_\parallel > l_e\right\} \end{aligned} \tag{8}$$

where $\eta \in \mathbb{R}$ in the $r_v$ term is a decay coefficient controlling the sensitivity to ball velocity, and $l_e \in \mathbb{R}$ in $r_p$ term denotes the plate's size (e.g., half length).

### B. Training with DRIS

For the reactive catching problem, we consider four physical properties as the domain parameters: the ball's radius, static friction, dynamic friction, and restitution coefficient, i.e., $\mathcal{C} \subset \mathbb{R}^4$. As shown in Fig. 3, in each episode, we spawn $N$ balls whose physical properties are independently sampled from $\mathcal{C}$ to construct the DRIS. In a vectorized representation, the DRIS state is formed by concatenating the states (i.e., relative displacements and velocities) of these ball instances, i.e., $\mathcal{S}_t = \left(\boldsymbol{s}_t^{(1)}, \cdots, \boldsymbol{s}_t^{(N)}\right) \in \mathbb{R}^{N\times 6}$.

As described in Sec. III-C, an encoder $\psi(\cdot)$ is required to transform the DRIS state into a latent vector in a compact feature space. The encoder must be size-agnostic to the DRIS size $N$, mapping any set to a fixed-dimensional latent feature $\boldsymbol{z}_t \in \mathbb{R}^{d_z}$ where $d_z$ denotes the dimension of the latent feature space. We adopt a point cloud-based Autoencoder (AE) [2] and extend its input dimensionality from 3D to 6D by incorporating both the positions and velocities of balls.

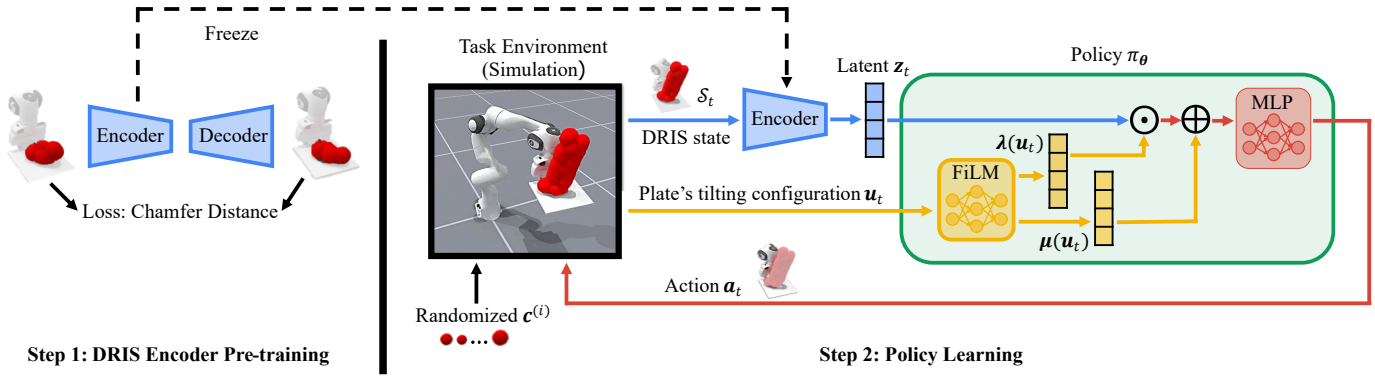



Fig. 3: Schematic overview of the DRIS-based learning pipeline. *Step 1 (Left):* The DRIS encoder is pre-trained to reconstruct the input DRIS state. *Step 2 (Right):* The policy network is trained via RL using the pre-trained (and frozen) DRIS encoder. At the beginning of each simulation episode, the physical property $\boldsymbol{c}^{(i)}$ of each ball instance in the DRIS is randomized; at each time step, the DRIS state $\mathcal{S}_t$ is encoded and fed to the policy $\pi_{\boldsymbol{\theta}}$ to generate an action $\boldsymbol{a}_t$.

The point cloud AE consists of an encoder built from 1D convolutional layers followed by a feature-wise max-pooling layer to extract features of the input set, and an MLP-based decoder that reconstructs the original set from the latent feature. We collect a dataset of DRIS samples by executing random robot actions, and pre-train this AE to reconstruct DRIS state using the Chamfer distance as reconstruction loss:

$$\mathcal{L}_\psi(\mathcal{S}_t, \widetilde{\mathcal{S}}) = \frac{1}{N}\sum_{\boldsymbol{s}\in\mathcal{S}_t}\min_{\boldsymbol{s}'\in\widetilde{\mathcal{S}}}\|\boldsymbol{s}-\boldsymbol{s}'\|^2 + \frac{1}{|\widetilde{\mathcal{S}}|}\sum_{\boldsymbol{s}'\in\widetilde{\mathcal{S}}}\min_{\boldsymbol{s}\in\mathcal{S}_t}\|\boldsymbol{s}-\boldsymbol{s}'\|^2 \tag{9}$$

where $\mathcal{S}_t$ and $\widetilde{\mathcal{S}}$ denote the input and reconstructed DRIS state, respectively. The encoder from the pre-trained AE is then used as $\psi(\cdot)$ for downstream policy learning.

With RL, at each training step, the pre-trained encoder $\psi$ transforms the observed DRIS state $\mathcal{S}_t$ into a latent vector $\boldsymbol{z}_t$; the policy, operating in the latent feature space (as detailed in Sec. IV-C), is trained using a policy gradient algorithm such as Proximal Policy Optimization (PPO) [40].

### C. FiLM-Conditioned Policy Network

The policy $\pi_{\boldsymbol{\theta}}$ consists of two components: a Feature-wise Linear Modulation (FiLM) [33] module and an MLP-based action generation network, as illustrated in Fig. 3.

At each step $t$, given the encoded state feature $\boldsymbol{z}_t$ and the observed plate tilting orientation $\boldsymbol{u}_t$ (which serves as the conditional signal), the FiLM module applies a feature-wise affine transformation:

$$\tilde{\boldsymbol{z}}_t = \text{FiLM}(\boldsymbol{z}_t, \boldsymbol{u}_t) = \boldsymbol{\lambda}(\boldsymbol{u}_t) \odot \boldsymbol{z}_t + \boldsymbol{\mu}(\boldsymbol{u}_t) \tag{10}$$

where $\boldsymbol{\lambda}(\boldsymbol{u}_t), \boldsymbol{\mu}(\boldsymbol{u}_t) \in \mathbb{R}^{d_z}$ are outputs of two small neural networks that generate scaling and bias coefficients, and $\odot$ denotes element-wise multiplication. The FiLM-modulated feature $\tilde{\boldsymbol{z}}_t \in \mathbb{R}^{d_z}$ is then passed through an MLP-based action-generation network to produce the action $\boldsymbol{a}_t$:

$$\boldsymbol{a}_t = \pi_{\boldsymbol{\theta}}(\boldsymbol{z}_t) = \text{MLP}(\tilde{\boldsymbol{z}}_t) = \text{MLP}(\text{FiLM}(\boldsymbol{z}_t, \boldsymbol{u}_t)) \tag{11}$$

The policy parameters $\boldsymbol{\theta}$ include the learnable weights of the FiLM modulation networks $\boldsymbol{\lambda}(\cdot)$ and $\boldsymbol{\mu}(\cdot)$ as well as those of the action-generation MLP. During training, all parameters are jointly optimized through gradient-based updates.

The use of FiLM conditioning is motivated by the contact physics of the catching task. When the ball rests on the plate, the plate's orientation (tilting angle) directly alters the tangential component of gravity along the plate surface and the contact direction, thereby affecting the ball's acceleration. By modulating the latent feature with the plate's orientation, the policy can dynamically adapt its action generation to account for these gravity-induced variations, enabling more physically consistent control across different tilt configurations.

### D. Control Implementation on High-DoF Manipulator

As described in Sec. IV-A, the action $\boldsymbol{a}_t$ consists of a desired plate displacement $\boldsymbol{\delta}_t$ and a tilting configuration $\boldsymbol{u}_t = (\alpha_t, \beta_t)$ which corresponds to a rotation of the plate about a horizontal axis $\boldsymbol{w}_t = (\cos\alpha_t, \sin\alpha_t, 0)^\top$ by an angle $\beta_t$ (all quantities expressed in the motion frame). The desired plate pose (expressed in the robot's base frame) is given by: $\widetilde{\boldsymbol{T}}_t = \left(\widetilde{\boldsymbol{R}}_t^e, \tilde{\boldsymbol{p}}_t^e\right) \in SE(3)$, where the orientation and position components are computed using Rodrigues' formula ($\hat{\boldsymbol{w}}_t \in \mathbb{R}^{3\times3}$ is the skew-symmetric matrix of $\boldsymbol{w}_t$):

$$\begin{aligned}\widetilde{\boldsymbol{R}}_t^e &= \boldsymbol{R}_z(\phi)\left(\mathbb{I} + \hat{\boldsymbol{w}}_t\sin\beta_t + \hat{\boldsymbol{w}}_t{}^2(1-\cos\beta_t)\right) \in SO(3)\\ \tilde{\boldsymbol{p}}_t^e &= \boldsymbol{R}_z(\phi)\,\boldsymbol{\delta}_t + \boldsymbol{p}_t^e \in \mathbb{R}^3\end{aligned} \tag{12}$$

The desired joint configuration of the robot $\tilde{\boldsymbol{q}}_t$ is obtained via inverse kinematics ($M$ being the robot's DoF): $\tilde{\boldsymbol{q}}_t = \text{IK}\left(\widetilde{\boldsymbol{T}}_t\right) \in \mathbb{R}^M$. The desired joint configuration is tracked using a joint-space torque controller that generates real-time joint torques (no inertial feedforward term is included, as the desired joint acceleration is zero):

$$\boldsymbol{\tau} = \boldsymbol{K}(\tilde{\boldsymbol{q}}_t - \boldsymbol{q}) - \boldsymbol{D}\dot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^M \tag{13}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}} \in \mathbb{R}^M$ are the observed joint angles and velocities from sensor readings; $\boldsymbol{K}, \boldsymbol{D} \in \mathbb{R}^{M\times M}$ are the stiffness and damping gain matrices; $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}}), \boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^M$ are the Coriolis and gravity terms, respectively.

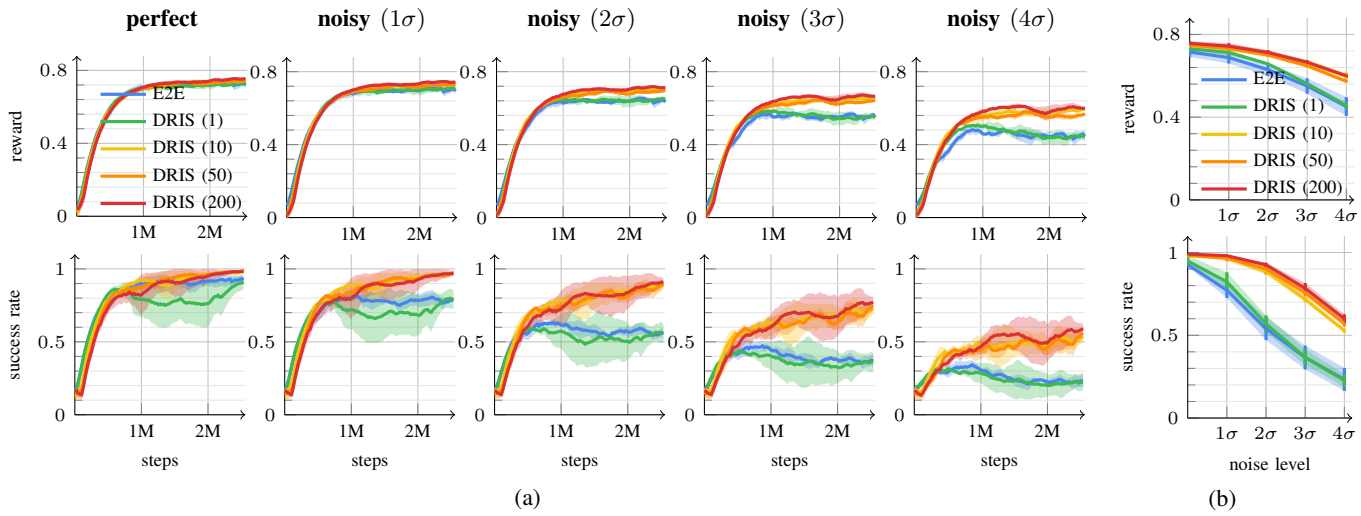



Fig. 4: Reward (top) and success rate (bottom) under varying observation noise. (a) Training curves across simulation steps, evaluated under progressively larger noise levels (left to right); (b) Evaluation of each policy at each noise level, averaged over the last 100 epochs.

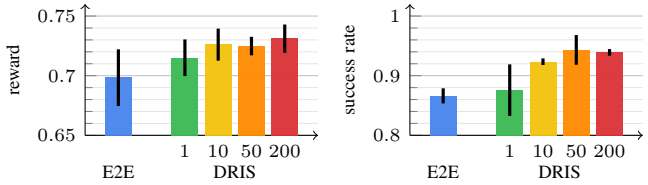



Fig. 5: Reward (left) and success rate (right) of each policy under execution errors, averaged over the last 100 epochs.

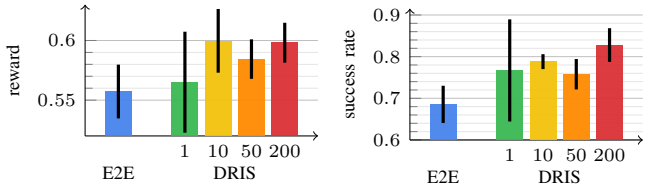



Fig. 6: Reward (left) and success rate (right) of each policy when manipulating balls with unseen restitution values (higher than those in training), averaged over the last 100 epochs.

## V. Training and Validation in Simulation

In this section, we implement the reactive catching task, train the policies purely in simulation using our proposed strategy, and evaluate its performance under various sources of uncertainty.

### A. Training in Simulation

The task simulation were implemented in ManiSkill [43]. We instantiated 128 parallel simulation environments for training, each containing a robot manipulating a set of multiple different balls (i.e., DRIS) for up to 20 steps (corresponding to one second of simulated time). When launching each environment instance, the balls are randomly positioned at a horizontal distance between $[1.0,\ 2.0]\,m$ away from the plate and assigned an initial velocity such that they reach a predefined catching region after $[1.0,\ 1.5]\,s$, approaching from a random incoming direction. The catching region is defined as a zone at the same height as the plate and within a horizontal radius of $0.2m$ from its center. Each episode begins when the ball is about to enter this region, specifically within a random lead time of $[0.08, 0.12]\,s$ before arrival, at which point the robot starts executing generated actions. The initial joint angles of the robot are sampled from a Gaussian distribution centered at the home configuration with a standard deviation of $0.02$ radians per joint. We disable collision detection in simulation between balls within the same DRIS and between balls and all robot links except the plate (i.e., only ball-plate collisions are enabled) to eliminate unmodeled contacts for stable training.

We first collected training data for the DRIS encoder by allowing the robot to interact with the DRIS containing 200 balls using random actions for 50 episodes. Across all 128 parallel environment instances, this resulted in a total of $128,000$ recorded DRIS state samples (i.e., the states of all balls at each step). The encoder was then trained on this dataset for 100 epochs, which took approximately 10 minutes. We then froze the pre-trained DRIS encoder and used it to train the FiLM-conditioned policy for 1000 epochs with PPO, which took approximately 2 hours.

### B. Evaluation against Uncertainties

To evaluate the effect of DRIS size on policy performance, we trained separate policies with different numbers of balls in the DRIS, specifically $N = 1, 10, 50, 200$. Moreover, to compare training with the proposed DRIS-based strategy against a standard RL setting, we trained an additional baseline policy using the same neural network architecture but learned end-to-end directly from the observed single-ball state to the desired action. We refer to this baseline as *E2E*. During evaluation, the DRIS-trained policies, although trained with multiple balls in each DRIS, were always tested on catching a single ball to

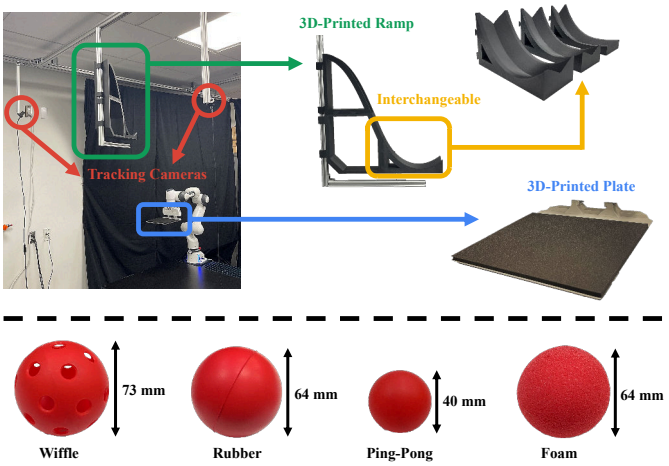
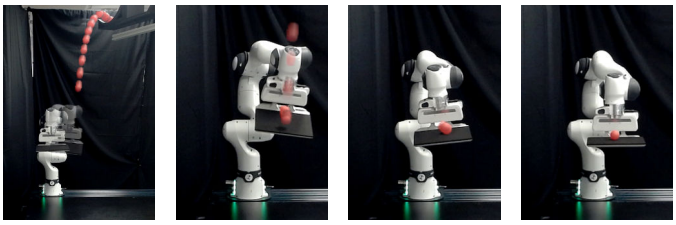



Fig. 7: The system setup (top) and the four different balls (bottom) used for real-world experiments.



| Ball | Ramp | VelTrack | E2E | DRIS (Ours) |
|---|---|---|---|---|
| Wiffle | $R = 0.13\,m$ | 0 / 5 | 0 / 5 | **5 / 5** |
| | $R = 0.20\,m$ | 0 / 5 | 2 / 5 | **4 / 5** |
| | $R = 0.32\,m$ | 0 / 5 | 0 / 5 | **3 / 5** |
| Rubber | $R = 0.13\,m$ | 0 / 5 | 0 / 5 | **2 / 5** |
| | $R = 0.20\,m$ | 0 / 5 | 0 / 5 | **5 / 5** |
| | $R = 0.32\,m$ | 0 / 5 | 1 / 5 | **2 / 5** |
| Ping-Pong | $R = 0.13\,m$ | 0 / 5 | 1 / 5 | **4 / 5** |
| | $R = 0.20\,m$ | 2 / 5 | 1 / 5 | **5 / 5** |
| | $R = 0.32\,m$ | 0 / 5 | 0 / 5 | **4 / 5** |
| Foam | $R = 0.13\,m$ | 0 / 5 | 0 / 5 | **2 / 5** |
| | $R = 0.20\,m$ | 1 / 5 | 1 / 5 | **3 / 5** |
| | $R = 0.32\,m$ | 0 / 5 | **2 / 5** | **2 / 5** |
| Total | | 3 / 60 | 8 / 60 | **41 / 60** |
| Success Rate | | 0.05 | 0.13 | **0.68** |

Fig. 8: *Top:* Snapshots of our policy catching a ping-pong ball released from the ramp. *Bottom:* Success rates for each policy across all combinations of ball types and ramp settings.

reflect the real task setting.

We evaluate robustness against the following three sources of uncertainty:

*Observation Noise.* This setting mimics scenarios where the ball's state is not accurately estimated (e.g., due to camera calibration errors or tracking inaccuracies). As shown in Fig. 4, we inject Gaussian noise into the observed ball states at different noise levels, obtained by scaling a base standard deviation $\sigma$ ($1\,cm$ for position and $5\,cm/s$ for velocity) by factors ranging from 1 to 4. We report both the final episode reward and success rate, where success is defined as the ball remaining on the plate with a velocity below $0.1\,m/s$ at the end of the episode. The E2E baseline and the DRIS policy with $N = 1$, both trained using a single ball (where the DRIS policy reduces to a standard RL setting but with a pre-trained encoder), perform comparably under perfect observations. However, as the observation noise increases, their performance degrades significantly. In contrast, DRIS policies trained with even a slightly larger number of balls (e.g., 10) exhibit substantially improved robustness, demonstrating greater tolerance to observation noise.

*Execution Error.* Then, we added noise to the desired joint positions by uniformly sampling perturbations from $[-0.05, 0.05]\ rad$ to simulate robot execution errors (e.g., controller's tracking inaccuracies). We report the final reward and success rate for each policy in Fig. 5. Similar to the observation noise evaluation, the E2E baseline is notably less robust than policies trained with DRIS. While increasing the DRIS size generally improved robustness in this evaluation, even a modest number of balls (e.g., 10) provided an obvious boost in performance under execution noise.

*Out-of-Distribution Physics.* To evaluate how the policies perform under unseen physical parameters, we tested them on balls with higher and more challenging restitution coefficients in $[0.7, 0.8]$, despite being trained only within $[0.4, 0.7]$. The reward and success rate are reported in Fig. 6. Compared with both the E2E baseline and the DRIS policy trained with a single ball, policies trained with larger DRIS generalize better to manipulate balls with unseen physical properties (restitution, in this case).

## VI. Real-Robot Deployment

To more realistically challenge our learned policies under real-world uncertainties and evaluate their sim-to-real transfer performance, we directly deployed the policies trained purely in simulation in a zero-shot manner on a real 7-DoF Franka Research 3 (FR3) robot manipulator, without any fine-tuning using real-world data.

### A. System Setup

Fig. 7 shows the setup we used for real-world evaluation (top), and the four test balls (wiffle, rubber, ping-pong, and foam) which differ in size, weight, and physical behavior (bottom). The plate is 3D-printed and covered with a neoprene foam padding to protect the robot from damage during repeated trials. Although the padding slightly reduces the impact force, its surface is smoother than the underlying plastic and causes the ball to roll and accelerate more easily, thus preserving the difficulty of the task.

To ensure relatively consistent initial ball states for fair comparative evaluation across different policies, we designed and 3D-printed a ramp with interchangeable lower sections and mounted it above the workspace. The ball is released by rolling from the top of the ramp, and the three interchangeable sections (with radii $R = 0.13$, $0.20$, and $0.32\,m$) induce different release speeds, allowing evaluation under varying initial ball states.

### B. Quantitative Evaluation

We deployed a single DRIS-trained policy (with $N = 200$) on the real robot across all evaluation trials and compared

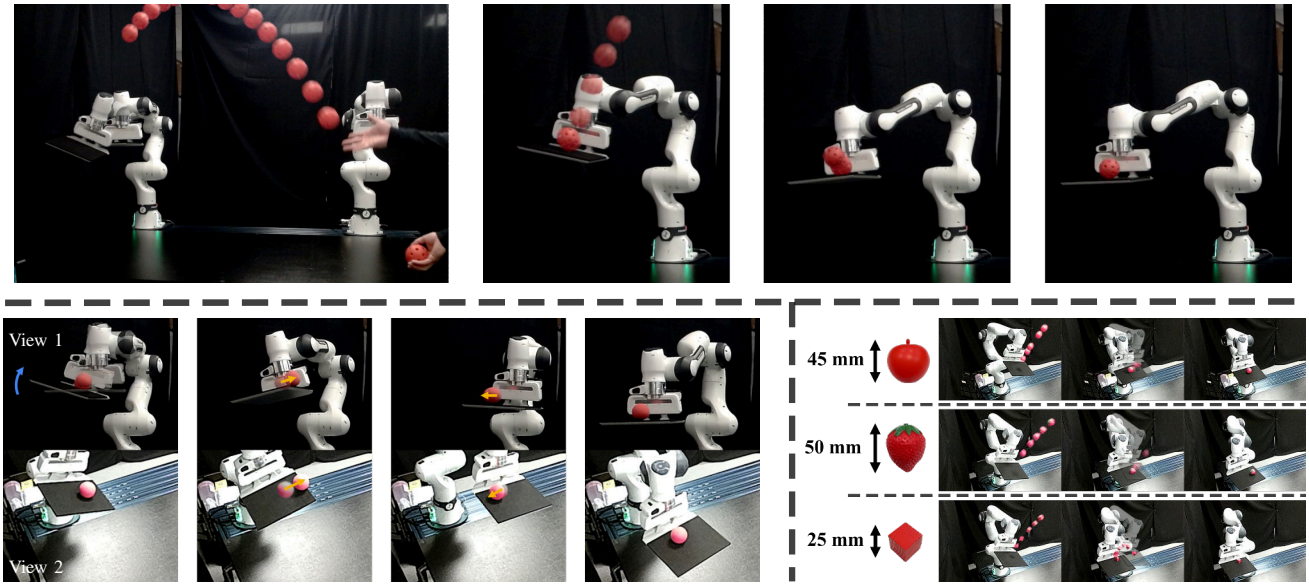



Fig. 9: Our policy successfully catches a wiffle ball thrown by a human (top), balances a foam ball (where a hard-coded motion rotates the plate by 30° (blue arrow) to initiate the ball's rolling in the first frame), and catches irregularly shaped objects.

its performance against two baselines: 1) *VelTrack*: a hand-crafted policy that moves the plate horizontally to follow the ball based on its estimated velocity after the first impact, with the plate oriented to face the ball's incoming direction at the moment of first impact; 2) *E2E*: similar to the simulation evaluation in Sec. V, this is a policy trained end-to-end with a single ball in simulation and directly deployed on the real robot. The controller gains, i.e., $\boldsymbol{K}$ and $\boldsymbol{D}$ in Eq. (13), were identified on the real robot by matching the controller's behavior to that in simulation. Aside from this gain identification, no additional adaptation or fine-tuning was performed.

For each policy, and for every combination of ball type and ramp, we conducted 5 trials and evaluated performance based on the number of successful catches, as summarized in Fig. 8. A trial is considered a successful catch if the robot keeps the ball on the plate for at least 10 seconds after it begins moving. The VelTrack baseline rarely succeeded, as its response was not sufficiently reactive, particularly under inaccurate ball velocity estimates. Its only 3 successful trials occurred when the ball was accidentally trapped in the gap between the plate and the finger. The E2E baseline, although producing reasonable motions to contact the ball, was highly sensitive to observation noise, leading to jerky robot movements that frequently knocked the ball off the plate or hit the robot's power limit. In contrast, our DRIS-trained policy achieved a 68% success rate when deployed zero-shot, and successfully caught all tested balls.

### C. Qualitative Evaluation

We conducted several qualitative evaluations to further assess the learned policy. First, as showcased in Fig. 9 (top), the policy successfully catches all test balls thrown by a human, despite the increased uncertainty and inconsistency of human throws. Second, as illustrated in Fig. 9 (bottom left), although trained for reactive catching, the policy generalizes directly to a pure balancing task. Third, as shown in Fig. 9 (bottom right), the policy was able to catch irregularly shaped objects, including a lightweight toy apple, a heavier toy strawberry, and a wooden cube. The wooden cube is caught only occasionally due to its highly unpredictable impact dynamics, which makes reliable catching difficult even for humans.

## VII. Conclusion and Limitations

In this work, we propose a DRIS-based learning strategy that enables robust policy learning in simulation and improved zero-shot sim-to-real transfer for dexterous manipulation under severe physical uncertainty. By simultaneously propagating multiple randomized instances instead of one, DRIS better approximates state evolution under diverse physical conditions. Combined with a task-specific policy design, this enables stable reactive catching without passive mechanical aids. Extensive experiments show substantially improved robustness and sim-to-real performance over conventional baselines.

*Limitations.* First, DRIS requires an encoder to map states of varying instances to a fixed-dimensional latent representation; while this is efficient for low-dimensional states, tasks involving high-dimensional visual or geometric inputs may require larger models and substantial pre-training. Second, effective DRIS training relies on efficient parallel propagation of multiple instances, which may be computationally expensive or impractical without suitable simulator support. Finally, since all instances share a single action, excessive divergence among instance states can destabilize learning, placing a practical limit on the amount of variation introduced; however, adaptive or curriculum-based instance randomization is a promising direction for future work.

## ACKNOWLEDGMENTS

This work was supported by the U.S. National Science Foundation (NSF) under grant FRR-2240040.

## REFERENCES

[1] Saminda Abeyruwan, Alex Bewley, Nicholas Matthew Boffi, Krzysztof Marcin Choromanski, David B D'Ambrosio, Deepali Jain, Pannag R Sanketi, Anish Shankar, Vikas Sindhwani, Sumeet Singh, et al. Agile catching with whole-body mpc and blackbox policy learning. In *Learning for Dynamics and Control Conference*, pages 851–863. PMLR, 2023.

[2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018.

[3] Anurag Ajay, Jiajun Wu, Nima Fazeli, Maria Bauza, Leslie P Kaelbling, Joshua B Tenenbaum, and Alberto Rodriguez. Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3066–3073. IEEE, 2018.

[4] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39 (1):3–20, 2020.

[5] Georg Bätz, Arhan Yaqub, Haiyan Wu, Kolja Kühnlenz, Dirk Wollherr, and Martin Buss. Dynamic manipulation: Nonprehensile ball catching. In *18th Mediterranean Conference on Control and Automation, MED'10*, pages 365–370. IEEE, 2010.

[6] Berthold Bäuml, Thomas Wimböck, and Gerd Hirzinger. Kinematically optimal catching a flying ball with a hand-arm-system. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2592–2599. IEEE, 2010.

[7] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 723–730. IEEE, 2011.

[8] Nikhil Chavan-Dafle and Alberto Rodriguez. Prehensile pushing: In-hand manipulation with push-primitives. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 6215–6222. IEEE, 2015.

[9] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[10] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307. PMLR, 2022.

[11] Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects. *The International Journal of Robotics Research*, 43(4):389–404, 2024.

[12] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[13] Nils Dengler, David Großklaus, and Maren Bennewitz. Learning goal-oriented non-prehensile pushing in cluttered scenes. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1116–1122. IEEE, 2022.

[14] Ke Dong, Karime Pereida, Florian Shkurti, and Angela P Schoellig. Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 6718–6725. IEEE, 2020.

[15] Ioannis Exarchos, Yifeng Jiang, Wenhao Yu, and C Karen Liu. Policy transfer via kinematic domain randomization and adaptation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 45–51. IEEE, 2021.

[16] Tobias Gold, Ralf Römer, Andreas Völz, and Knut Graichen. Catching objects with a robot arm using model predictive control. In *2022 American Control Conference (ACC)*, pages 1915–1920. IEEE, 2022.

[17] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[18] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[19] Binghao Huang, Yuanpei Chen, Tianyu Wang, Yuzhe Qin, Yaodong Yang, Nikolay Atanasov, and Xiaolong Wang. Dynamic handover: Throw and catch with bimanual hands. In *Conference on Robot Learning*, 2023.

[20] Johann Huber, François Hélénon, Hippolyte Watrelot, Faïz Ben Amar, and Stéphane Doncieux. Domain randomization for sim2real transfer of automatically generated grasping datasets. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4112–4118. IEEE, 2024.

[21] Ajinkya Jain and Scott Niekum. Efficient hierarchical robot motion planning under uncertainty and hybrid dynamics. In *Conference on Robot Learning*, pages 757–766. PMLR, 2018.

[22] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss,

Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. *arXiv preprint arXiv:2104.02646*, 2021.

[23] Yifeng Jiang, Tingnan Zhang, Daniel Ho, Yunfei Bai, C Karen Liu, Sergey Levine, and Jie Tan. Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2884–2890. IEEE, 2021.

[24] Seungsu Kim, Ashwini Shukla, and Aude Billard. Catching objects in flight. *IEEE Transactions on Robotics*, 30 (5):1049–1065, 2014.

[25] Ziang Liu, Stephen Tian, Michelle Guo, Karen Liu, and Jiajun Wu. Learning to design and use tools for robotic manipulation. In *Conference on Robot Learning*, pages 887–905. PMLR, 2023.

[26] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[27] 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.

[28] 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.

[29] Fabio Muratore, Fabio Ramos, Greg Turk, Wenhao Yu, Michael Gienger, and Jan Peters. Robot learning from randomized simulations: A review. *Frontiers in Robotics and AI*, 9:799893, 2022.

[30] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.

[31] Akio Namiki and Naoki Itoi. Ball catching in kendama game by estimating grasp conditions based on a high-speed vision system and tactile sensors. In *IEEE International Conference on Humanoid Robots (Humanoids)*, pages 634–639. IEEE, 2014.

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

[33] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[34] Aleksei Petrenko, Arthur Allshire, Gavriel State, Ankur Handa, and Viktor Makoviychuk. Dexpbt: Scaling up dexterous manipulation for hand-arm systems with population based training. *Robotics: Science and Systems*, 2023.

[35] Kai Ploeger, Michael Lutter, and Jan Peters. High acceleration reinforcement learning for real-world juggling with binary rewards. In *Conference on Robot Learning*, pages 642–653. PMLR, 2021.

[36] Josep M. Porta, Nikos Vlassis, Matthijs T.J. Spaan, and Pascal Poupart. Point-based value iteration for continuous pomdps. *Journal of Machine Learning Research*, 7 (83):2329–2367, 2006.

[37] Fabio Ramos, Rafael Possas, and Dieter Fox. Bayessim: Adaptive domain randomization via probabilistic inference for robotics simulators. *Robotics: Science and Systems*, 2019.

[38] Seyed Sina Mirrazavi Salehian, Mahdi Khoramshahi, and Aude Billard. A dynamical system approach for softly catching a flying object: Theory and experiment. *IEEE Transactions on Robotics*, 32(2):462–471, 2016.

[39] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[41] Matthew Sheckells, Gowtham Garimella, Subhransu Mishra, and Marin Kobilarov. Using data-driven domain randomization to transfer robust control policies to mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3224–3230. IEEE, 2019.

[42] Nikhil Sobanbabu, Guanqi He, Tairan He, Yuxiang Yang, and Guanya Shi. Sampling-based system identification with active exploration for legged sim2real learning. In *9th Annual Conference on Robot Learning*, 2025.

[43] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Viswesh Nagaswamy Rajesh, Yong Woo Choi, Yen-Ru Chen, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

[44] Gabriele Tiboni, Karol Arndt, and Ville Kyrki. Dropo: Sim-to-real transfer with offline domain randomization. *Robotics and Autonomous Systems*, 166:104432, 2023.

[45] 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 *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

[46] Josh Tobin, Lukas Biewald, Rocky Duan, Marcin Andrychowicz, Ankur Handa, Vikash Kumar, Bob McGrew, Alex Ray, Jonas Schneider, Peter Welinder, et al. Domain randomization and generative models for robotic grasping. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3482–3489. IEEE, 2018.

[47] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033. IEEE, 2012.

[48] Andrew Wang, Andrew C Li, Toryn Q Klassen, Rodrigo Toro Icarte, and Sheila A McIlraith. Learning belief representations for partially observable deep rl. In *International Conference on Machine Learning*, pages 35970–35988. PMLR, 2023.

[49] Gaotian Wang, Kejia Ren, Andrew S Morgan, and Kaiyu Hang. Caging in time: A framework for robust object manipulation under uncertainties and limited robot perception. *The International Journal of Robotics Research*, page 02783649251343926, 2025.

[50] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

[51] Yuanhang Zhang, Tianhai Liang, Zhenyang Chen, Yanjie Ze, and Huazhe Xu. Catch it! learning to catch in flight with mobile dexterous hands. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025.

## APPENDIX A. ALGORITHMIC PIPELINE AND IMPLEMENTATION DETAILS

### A. DRIS-based Policy Learning and Zero-Shot Inference

We summarize the DRIS-based policy learning pipeline (using an on-policy example) and its zero-shot inference in Alg. 1. At each training iteration in simulation, we randomly initialize the system state $\boldsymbol{s}_0$ and sample $N$ different domain parameters from $\mathcal{C}$ to construct the DRIS, with all instances initialized at $\boldsymbol{s}_0$. We then collect on-policy rollouts to update the policy, where the rollout stores the encoded latent state $\boldsymbol{z}_t$ rather than the full set of instance states $\left\{\boldsymbol{s}_t^{(i)}\right\}_{i=1}^{N}$. After training, the policy is deployed in the real world without fine-tuning: At each time step, the observed single real-world state $\boldsymbol{s}$ (equivalent to a DRIS of size $N = 1$) is encoded and provided as input to the trained policy until the task is finished.

**Algorithm 1** DRIS-based Policy Learning and Inference

**Input:** Domain parameter space $\mathcal{C}$, initial policy $\pi_{\boldsymbol{\theta}}$, DRIS encoder $\psi$, dynamics $\mathcal{F}$ modeled by a simulator

***Part (a): Policy Learning in Simulation***

**while** training **do**
  $\boldsymbol{s}_0 \leftarrow \text{INITSTATE}()$ ▷ Random Initialization
  $\mathcal{D}_0 \leftarrow \{\}$ ▷ Initialize DRIS
  **for** $i = 1, \cdots, N$ **do**
    $\boldsymbol{c}^{(i)} \sim \mathcal{U}(\mathcal{C})$, $\boldsymbol{s}_0^{(i)} \leftarrow \boldsymbol{s}_0$
    $\mathcal{D}_0 \leftarrow \mathcal{D}_0 \cup \left\{\left(\boldsymbol{s}_0^{(i)}, \boldsymbol{c}^{(i)}\right)\right\}$
  **end for**
  $\mathcal{T} \leftarrow \{\}$ ▷ Simulation Rollout
  **for** $t = 0, \cdots, T-1$ **do**
    $\mathcal{S}_t \leftarrow \text{proj}_{\mathcal{S}}(\mathcal{D}_t)$ ▷ DRIS State
    $\boldsymbol{z}_t \leftarrow \psi(\mathcal{S}_t)$, $\boldsymbol{a}_t \leftarrow \pi_{\boldsymbol{\theta}}(\boldsymbol{z}_t)$
    $r_t \leftarrow \frac{1}{N}\sum_{\boldsymbol{s}\in\mathcal{S}_t} r(\boldsymbol{s}, \boldsymbol{a}_t)$ ▷ Average Reward
    $\mathcal{D}_{t+1} \leftarrow \mathcal{F}(\mathcal{D}_t, \boldsymbol{a}_t)$ ▷ Propagate by Eq. (5)
    $\mathcal{T} \leftarrow \mathcal{T} \cup (\boldsymbol{z}_t, \boldsymbol{a}_t, r_t)$
  **end for**
  $\pi_{\boldsymbol{\theta}} \leftarrow \text{POLICYUPDATE}(\pi_{\boldsymbol{\theta}}, \mathcal{T})$ ▷ Policy Update
**end while**

***Part (b): Zero-Shot Inference in Real World***

**while not** finished **do**
  $\boldsymbol{s} \leftarrow \text{OBSERVESTATE}()$ ▷ Real-World State
  $\boldsymbol{z} \leftarrow \psi(\{\boldsymbol{s}\})$, $\boldsymbol{a} \leftarrow \pi_{\boldsymbol{\theta}}(\boldsymbol{z})$ ▷ Generate Action
  finished $\leftarrow \text{EXECUTE}(\boldsymbol{a})$
**end while**

### B. Implementation Details

The learning pipeline and task simulation were executed on a single NVIDIA GeForce RTX 3060 GPU with 12 GB of memory. The actions are generated at 20 Hz, corresponding to a simulation step of 1/20 second. The plate size (half-length) is set to $l_e = 12\,cm$. The domain-randomized parameters were uniformly sampled from the following ranges: ball radius $[2,\ 4]\,cm$; static and dynamic friction coefficients $[0,\ 0.1]$; and restitution coefficient $[0.4,\ 0.7]$. The latent feature dimension of encoded DRIS was set to $d_z = 64$, and the decay coefficient in the reward of Eq. (8) was $\eta = 0.25$.

When deploying the trained policy in the real world, the ball's 3D trajectory is estimated using two cameras at 80 FPS through image-based ball detection (e.g., color segmentation followed by contour-based circle fitting), triangulation, and parabolic curve fitting. The ball's velocity is estimated via ordinary least squares (OLS) by fitting a line to position measurements over a 0.1-second sliding window. The trained policies were loaded on a 3.4 GHz AMD Ryzen 9 5950X CPU for real-time inference.

## APPENDIX B. THEORETICAL ANALYSIS OF DRIS

In this appendix, we provide a theoretical analysis underlying Theorem III.1. Specifically, we contrast the conventional Domain Randomization (DR) with our proposed DRIS under a unified problem setup ( Appendix B-A); we interpret DRIS as an exact particle approximation for the system belief propagation ( Appendix B-B); then, we analyze to show that DRIS yields a more stable optimization ( Appendix B-C), more robust policies ( Appendix B-D), and improved sim-to-real generalization ( Appendix B-E).

### A. Unified Problem Setup

To ensure this appendix is self-contained, we briefly recapitulate the problem setup and notations from Sec. III in the main text. The manipulation problem is defined over an observable state space $\mathcal{S}$ and an action space $\mathcal{A}$. We denote a state by $\boldsymbol{s} \in \mathcal{S}$ and an action by $\boldsymbol{a} \in \mathcal{A}$. The system state evolves according to a deterministic transition function $\boldsymbol{s}_{t+1} = f(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{c})$, conditioned on the physical parameters $\boldsymbol{c} \in \mathcal{C}$ (e.g., friction, mass). The parameters $\boldsymbol{c}$ are unknown during execution but are assumed to be distributed according to a fixed prior $p(\boldsymbol{c})$ (which is simplified to a uniform distribution in our implementation).

*Ensemble State with Shared Actions.* In the proposed **Domain-Randomized Instance Set (DRIS)** framework, during policy learning, we sample a set of $N$ i.i.d. physical instances drawn from the prior, denoted $\hat{\mathcal{C}} = \left\{\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(N)}\right\}$ where each $\boldsymbol{c}^{(i)} \sim p(\boldsymbol{c})$, The ensemble state of these $N$ parallel instances is denoted as $\mathcal{S}_t = \left\{\boldsymbol{s}_t^{(1)}, \ldots, \boldsymbol{s}_t^{(N)}\right\} \in \mathcal{S}^N$. At each time step $t$, the policy $\pi_{\boldsymbol{\theta}}$ (parameterized by $\boldsymbol{\theta}$) generates a single consensus action $\boldsymbol{a}_t = \pi_{\boldsymbol{\theta}}(\mathcal{S}_t)$ that is applied to all $N$ parallel instances simultaneously. This introduces an explicit coupling: The state transition of instance $i$ depends on the states of all other instances via the shared action:

$$\boldsymbol{s}_{t+1}^{(i)} = f\left(\boldsymbol{s}_t^{(i)}, \pi_{\boldsymbol{\theta}}(\mathcal{S}_t), \boldsymbol{c}^{(i)}\right), \quad \forall i \in \{1, \ldots, N\} \tag{14}$$

*The Global Objective.* With all instances starting from the same initial state $\boldsymbol{s}_0 \sim p(\boldsymbol{s}_0)$, the controlled trajectory of the $i$-th instance is $\tau_{\boldsymbol{c}^{(i)}} = \left(\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1^{(i)}, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_T^{(i)}\right)$. We denote

$\tau_{\boldsymbol{c}^{(i)}}$ to emphasize that the trajectory is implicitly conditioned on the specific realization of the physical parameter $\boldsymbol{c}^{(i)}$. The *Discounted Cumulative Reward* (Return) of a single trajectory is then defined as:

$$R\left(\tau_{\boldsymbol{c}^{(i)}}\right) \coloneqq \sum_{t=0}^{T-1} \gamma^t \, r\left(\boldsymbol{s}_t^{(i)}, \, \boldsymbol{a}_t\right) \tag{15}$$

The global optimization objective with respect to the policy parameter $\boldsymbol{\theta}$ is to maximize the expected average return over all instance trajectories:

$$\mathcal{J}_N\left(\boldsymbol{\theta}\right) \coloneqq \mathbb{E}_{\boldsymbol{s}_0 \sim p(\boldsymbol{s}_0), \boldsymbol{c}^{(i)} \sim p(\boldsymbol{c})}\left[\frac{1}{N} \sum_{i=1}^{N} R\left(\tau_{\boldsymbol{c}^{(i)}}\right)\right] \tag{16}$$

**Relationship to Conventional Domain Randomization (DR).** By setting the DRIS size to $N = 1$, the ensemble state collapses to a singleton $\mathcal{S}_t = \{\boldsymbol{s}_t\}$ and the unified objective $\mathcal{J}_N$ reduces to:

$$\mathcal{J}_1\left(\boldsymbol{\theta}\right) = \mathbb{E}_{\boldsymbol{s}_0 \sim p(\boldsymbol{s}_0), \boldsymbol{c} \sim p(\boldsymbol{c})}\left[R\left(\tau_{\boldsymbol{c}}\right)\right] \tag{17}$$

This recovers the conventional DR objective, where the expectation is estimated via Monte Carlo sampling of a single physics parameter per episode.

### B. Interpretation: Exact Particle Propagation

We prove that the simulation step in DRIS corresponds to the exact propagation of a particle approximation of the system's belief state.

**Continuous Belief Dynamics.** Let $b_t\left(\boldsymbol{s}, \boldsymbol{c}\right)$ be the joint probability density (belief) of the state and physics parameters at time $t$. Given a deterministic action $\boldsymbol{a}_t$, the belief evolves according to the **Liouville Equation** (the continuity equation for probability mass):

$$\begin{aligned} b_{t+1}\left(\boldsymbol{s}', \boldsymbol{c}'\right) = \int_{\mathcal{S}} \int_{\mathcal{C}} & \delta\Big(\boldsymbol{s}' - f\left(\boldsymbol{s}, \boldsymbol{a}_t, \boldsymbol{c}\right)\Big) \\ & \cdot \delta\left(\boldsymbol{c}' - \boldsymbol{c}\right) \, b_t\left(\boldsymbol{s}, \boldsymbol{c}\right) \, d\boldsymbol{c} \, d\boldsymbol{s} \end{aligned} \tag{18}$$

Here, $\delta\left(\cdot\right)$ is the Dirac delta function. The term $\delta\left(\boldsymbol{c}' - \boldsymbol{c}\right)$ enforces that physics parameters are static during propagation.

**Proposition Appendix B.1** (Exact Particle Propagation)**.** *Let the initial belief be approximated by an empirical measure* $\hat{b}_{t,N}\left(\boldsymbol{s}, \boldsymbol{c}\right) = \frac{1}{N} \sum_{i=1}^{N} \delta(\boldsymbol{s} - \boldsymbol{s}_t^{(i)}) \delta(\boldsymbol{c} - \boldsymbol{c}^{(i)})$*. The time evolution of this measure under the Liouville Equation is exactly the empirical measure constructed from the updated particles* $\boldsymbol{s}_{t+1}^{(i)} = f\left(\boldsymbol{s}_t^{(i)}, \, \boldsymbol{a}_t, \, \boldsymbol{c}^{(i)}\right)$.

*Proof:* Substituting $\hat{b}_{t,N}$ into Eq. (18) and utilizing the sifting property of the Dirac delta $\int f(x)\delta(x - x_0)dx = f(x_0)$:

$$\begin{aligned} \hat{b}_{t+1}\left(\boldsymbol{s}', \boldsymbol{c}'\right) &= \iint \delta\Big(\boldsymbol{s}' - f\left(\boldsymbol{s}, \boldsymbol{a}_t, \boldsymbol{c}\right)\Big) \, \delta(\boldsymbol{c}' - \boldsymbol{c}) \\ &\qquad \left[\frac{1}{N} \sum_{i=1}^{N} \delta(\boldsymbol{s} - \boldsymbol{s}_t^{(i)}) \delta(\boldsymbol{c} - \boldsymbol{c}^{(i)})\right] d\boldsymbol{s} \, d\boldsymbol{c} \\ &= \frac{1}{N} \sum_{i=1}^{N} \iint \delta\Big(\boldsymbol{s}' - f\left(\boldsymbol{s}, \boldsymbol{a}_t, \boldsymbol{c}\right)\Big) \, \delta(\boldsymbol{c}' - \boldsymbol{c}) \\ &\qquad \delta(\boldsymbol{s} - \boldsymbol{s}_t^{(i)}) \, \delta(\boldsymbol{c} - \boldsymbol{c}^{(i)}) \, d\boldsymbol{s} \, d\boldsymbol{c} \\ &= \frac{1}{N} \sum_{i=1}^{N} \delta\Big(\boldsymbol{s}' - f\left(\boldsymbol{s}_t^{(i)}, \boldsymbol{a}_t, \boldsymbol{c}^{(i)}\right)\Big) \, \delta\left(\boldsymbol{c}' - \boldsymbol{c}^{(i)}\right) \end{aligned} \tag{19}$$

This resulting distribution corresponds precisely to the discrete set of updated simulation states. Thus, DRIS is mathematically equivalent to propagating the belief state through the exact system dynamics. ■

### C. Gradient Variance Reduction

We analyze the variance of the gradient estimator $\nabla_{\boldsymbol{\theta}} \mathcal{J}_N\left(\boldsymbol{\theta}\right)$. In particular, assuming the policy is permutation invariant (i.e., the ordering of instances in the DRIS does not affect the policy output), we show that DRIS yields a reduced gradient variance compared to conventional DR.

**Definition Appendix B.1** (Permutation Invariant Policy)**.** *A policy* $\pi_{\boldsymbol{\theta}}$ *is permutation invariant if for any permutation* $\phi$ *of indices* $\{1, \ldots, N\}$ *and any ensemble state* $\left\{\boldsymbol{s}^{(i)}, \ldots, \boldsymbol{s}_{(N)}\right\} \in \mathcal{S}^N$*:*

$$\pi_{\boldsymbol{\theta}}\left(\left\{\boldsymbol{s}^{(\phi(1))}, \ldots, \boldsymbol{s}^{(\phi(N))}\right\}\right) = \pi_{\boldsymbol{\theta}}\left(\left\{\boldsymbol{s}^{(1)}, \ldots, \boldsymbol{s}^{(N)}\right\}\right) \tag{20}$$

**Lemma Appendix B.1** (Gradient Exchangeability)**.** *Let* $\boldsymbol{g}_i \triangleq \nabla_{\boldsymbol{\theta}} R\left(\tau_{\boldsymbol{c}^{(i)}}\right)$ *denote the gradient obtained from the* $i$*-th instance. If* $\pi_{\boldsymbol{\theta}}$ *is permutation invariant, the gradients* $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_N$ *are exchangeable random variables. Consequently, they share a common mean* $\boldsymbol{\mu}$*, a common covariance matrix* $\boldsymbol{\Sigma}$ *and a common cross-covariance matrix* $\boldsymbol{\Sigma}_{cross}$*:*

$$Cov\left(\boldsymbol{g}_i\right) = \boldsymbol{\Sigma}, \quad Cov\left(\boldsymbol{g}_i, \boldsymbol{g}_j\right) = \boldsymbol{\Sigma}_{cross} \quad \forall i \neq j \tag{21}$$

*Proof:* Let $\boldsymbol{G} = \left(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_N\right)$ be the joint vector of gradients. Due to the i.i.d. sampling of physics parameters and the permutation invariance of $\pi_{\boldsymbol{\theta}}$, the joint distribution of $\boldsymbol{G}$ is invariant under any permutation $\phi$: $p\left(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_N\right) = p\left(\boldsymbol{g}_{\phi(1)}, \ldots, \boldsymbol{g}_{\phi(N)}\right)$. Hence, $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_N$ are exchangeable. Exchangeability implies identical marginals, so $\mathbb{E}[\boldsymbol{g}_i] = \boldsymbol{\mu}$ and $\mathrm{Cov}(\boldsymbol{g}_i) = \mathbb{E}[(\boldsymbol{g}_i - \boldsymbol{\mu})(\boldsymbol{g}_i - \boldsymbol{\mu})^T] = \boldsymbol{\Sigma}$ for all $i$. Moreover, exchangeability implies that all unordered pairs $(\boldsymbol{g}_1, \boldsymbol{g}_j)$, $i \neq j$, have identical joint distributions, and thus identical cross-covariance $\mathrm{Cov}(\boldsymbol{g}_i, \boldsymbol{g}_j) = \mathbb{E}[(\boldsymbol{g}_i - \boldsymbol{\mu})(\boldsymbol{g}_j - \boldsymbol{\mu})^T] = \boldsymbol{\Sigma}_{\text{cross}}$ for all $i \neq j$. ■

**Proposition Appendix B.2** (Variance Reduction)**.** *Let the scalar total variance be the trace of the covariance matrix:* $\sigma^2 \triangleq Tr\left(\boldsymbol{\Sigma}\right)$*, and the scalar correlation coefficient be* $\rho \triangleq$

*$Tr\left(\boldsymbol{\Sigma}_{cross}\right)/\sigma^2$. Then the total variance of the DRIS gradient estimator $\hat{\boldsymbol{g}}_{DRIS} = \frac{1}{N}\sum \boldsymbol{g}_i$ is given by:*

$$Tr\left(Cov\left(\hat{\boldsymbol{g}}_{DRIS}\right)\right) = \sigma^2\left(\rho + \frac{1-\rho}{N}\right) \tag{22}$$

*Consequently, DRIS ($N > 1$) achieves strict variance reduction compared to conventional DR ($N = 1$) if and only if $\rho < 1$ (i.e., when the aggregate cross-covariance is strictly smaller than the marginal covariance).*

*Proof:* Using the bilinearity of covariance and the exchangeability of the gradients:

$$\begin{aligned}\mathrm{Cov}\left(\hat{\boldsymbol{g}}_{DRIS}\right) &= \mathrm{Cov}\left(\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{g}_i\right) \\ &= \frac{1}{N^2}\left[\sum_{i=1}^{N}\mathrm{Cov}\left(\boldsymbol{g}_i\right) + \sum_{i\neq j}\mathrm{Cov}\left(\boldsymbol{g}_i, \boldsymbol{g}_j\right)\right]\end{aligned} \tag{23}$$

Substituting the covariance and cross-covariance matrices defined in Lemma. Appendix B.1:

$$\begin{aligned}\mathrm{Cov}\left(\hat{\boldsymbol{g}}_{DRIS}\right) &= \frac{1}{N^2}\left[N\boldsymbol{\Sigma} + N(N-1)\boldsymbol{\Sigma}_{\text{cross}}\right] \\ &= \frac{1}{N}\boldsymbol{\Sigma} + \frac{N-1}{N}\boldsymbol{\Sigma}_{\text{cross}}\end{aligned} \tag{24}$$

Taking the trace (a linear operator) to obtain the scalar total variance:

$$\begin{aligned}\mathrm{Tr}\left(\mathrm{Cov}\left(\hat{\boldsymbol{g}}_{DRIS}\right)\right) &= \frac{1}{N}\mathrm{Tr}\left(\boldsymbol{\Sigma}\right) + \left(1-\frac{1}{N}\right)\mathrm{Tr}\left(\boldsymbol{\Sigma}_{\text{cross}}\right) \\ &= \frac{\sigma^2}{N} + \left(1-\frac{1}{N}\right)\rho\sigma^2 \\ &= \sigma^2\left(\rho + \frac{1-\rho}{N}\right)\end{aligned} \tag{25}$$

In practice, independently sampled physics parameters introduce instance-specific perturbations in rollout trajectories, so gradients across instances are typically not perfectly aligned and $\rho < 1$ is expected, yielding strict variance reduction for DRIS ($N > 1$) compared to conventional DR ($N = 1$). ■

**Implication:** The result above implies that as the DRIS size increases ($N \to \infty$), the estimator's total variance decreases from the conventional DR baseline $\sigma^2$ to a non-zero floor $\rho\sigma^2$. Intuitively, increasing $N$ averages out the instance-specific (uncorrelated) component of gradient variability; this stabilizes the optimization by suppressing the high-variance noise from single-instance rollouts.

### *D. Robustness Analysis*

We analyze how the DRIS objective implicitly promotes robustness against physical uncertainty. While conventional DR and DRIS optimize the same theoretical objective in expectation, their empirical optimization landscapes differ fundamentally.

In our problem setup, we can define the empirical objective as the average return over the $N$ rollout trajectories of DRIS. Since physics parameters $\boldsymbol{c}^{(i)}$ are sampled i.i.d. from $p\left(\boldsymbol{c}\right)$, the rollouts are conditionally i.i.d. across instances given the shared action sequence $\{\boldsymbol{a}_t\}_{t=0}^{T-1}$:

$$\hat{\mathcal{J}}_N\left(\boldsymbol{\theta}\right) \coloneqq \frac{1}{N}\sum_{i=1}^{N} R\left(\tau_{\boldsymbol{c}^{(i)}}\right) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\gamma^t\, r\left(\boldsymbol{s}_t^{(i)},\, \boldsymbol{a}_t\right) \tag{26}$$

We analyze the optimization landscape by decomposing this return into its local contributions at a single time step $t$, and define empirical local cost (negative reward) as $C_t(\boldsymbol{s}) \triangleq -r\left(\boldsymbol{s}, \boldsymbol{a}_t\right)$. The global maximization is equivalent to minimizing the discounted sum of empirical local costs:

$$\hat{\mathcal{J}}_N(\boldsymbol{\theta}) = -\sum_{t=0}^{T-1}\gamma^t \underbrace{\left(\frac{1}{N}\sum_{i=1}^{N} C_t(\boldsymbol{s}_t^{(i)})\right)}_{\hat{J}_{N,t}} \tag{27}$$

This decomposition simplifies the analysis: by examining the local cost $\hat{J}_{N,t}$ conditioned on the shared action $\boldsymbol{a}_t$, we can determine how $N$ affects the signal-to-noise ratio of the gradient at each step.

**Assumption Appendix B.1** (Local Geometry)**.** *We assume $C_t(\boldsymbol{s})$ is twice differentiable and locally convex w.r.t. the state $\boldsymbol{s}$ near the mean $\boldsymbol{\mu}_t \triangleq \mathbb{E}[\boldsymbol{s}_t \mid \boldsymbol{a}_t]$. We approximate $C_t(\boldsymbol{s}_t)$ using a second-order Taylor expansion:*

$$C_t\left(\boldsymbol{s}\right) \approx C_t(\boldsymbol{\mu}_t) + \boldsymbol{g}_t^\top(\boldsymbol{s}-\boldsymbol{\mu}_t) + \frac{1}{2}(\boldsymbol{s}-\boldsymbol{\mu}_t)^\top\mathbf{H}_t(\boldsymbol{s}-\boldsymbol{\mu}_t) \tag{28}$$

*where $\boldsymbol{g}_t = \nabla_{\boldsymbol{s}} C_t\left(\boldsymbol{\mu}_t\right)$ and $\mathbf{H}_t = \nabla_{\boldsymbol{s}}^2 C_t(\boldsymbol{\mu}_t) \succeq 0$ are the Jacobian and Hessian, respectively.*

Substituting this expansion into $\hat{J}_{N,t}$ and using the cyclic property of the trace operator $\mathrm{Tr}\left(\boldsymbol{x}^\top\mathbf{A}\boldsymbol{x}\right) = \mathrm{Tr}\left(\mathbf{A}\boldsymbol{x}\boldsymbol{x}^\top\right)$, we derive the decomposition:

$$\begin{aligned}\hat{J}_{N,t} &= \frac{1}{N}\sum_{i=1}^{N} C_t\left(\boldsymbol{s}_t^{(i)}\right) \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\Big[C_t(\boldsymbol{\mu}_t) + \boldsymbol{g}_t^\top(\boldsymbol{s}_t^{(i)}-\boldsymbol{\mu}_t) \\ &\quad + \frac{1}{2}\mathrm{Tr}\left(\mathbf{H}_t(\boldsymbol{s}_t^{(i)}-\boldsymbol{\mu}_t)(\boldsymbol{s}_t^{(i)}-\boldsymbol{\mu}_t)^\top\right)\Big] \\ &= C_t(\boldsymbol{\mu}_t) + \boldsymbol{g}_t^\top\left(\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{s}_t^{(i)}-\boldsymbol{\mu}_t)\right) \\ &\quad + \frac{1}{2}\mathrm{Tr}\left(\mathbf{H}_t\left(\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{s}_t^{(i)}-\boldsymbol{\mu}_t)(\boldsymbol{s}_t^{(i)}-\boldsymbol{\mu}_t)^\top\right)\right) \\ &= \underbrace{C_t(\boldsymbol{\mu}_t)}_{\text{Nominal}} + \underbrace{\boldsymbol{g}_t^\top\bar{\boldsymbol{\delta}}_{N,t}}_{\text{Linear Noise}} + \underbrace{\frac{1}{2}\mathrm{Tr}\left(\mathbf{H}_t\hat{\boldsymbol{\Sigma}}_{N,t}\right)}_{\text{Robustness Signal}}\end{aligned} \tag{29}$$

where $\bar{\boldsymbol{\delta}}_{N,t} = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{s}_t^{(i)} - \boldsymbol{\mu}_t\right)$ is the sample mean error; $\hat{\boldsymbol{\Sigma}}_{N,t} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{s}_t^{(i)}-\boldsymbol{\mu}_t)(\boldsymbol{s}_t^{(i)}-\boldsymbol{\mu}_t)^\top$ is the empirical second moment about the true mean, satisfying $\mathbb{E}\left[\hat{\boldsymbol{\Sigma}}_{N,t}\right] = \boldsymbol{\Sigma}_t$.

**Theorem Appendix B.1** (Asymptotic Unmasking)**.** *As the DRIS size $N$ increases, the linear noise is suppressed while*

*the robustness signal concentrates. The empirical local cost converges in probability to a regularized objective:*

$$\hat{J}_{N,t} \xrightarrow{P} C_t(\boldsymbol{\mu}_t) + \frac{1}{2}\mathrm{Tr}(\mathbf{H}_t \boldsymbol{\Sigma}_t) \quad \text{as } N \to \infty. \tag{30}$$

*Proof:* By the Weak Law of Large Numbers, the error $\bar{\boldsymbol{\delta}}_{N,t} \xrightarrow{P} \mathbf{0}$, causing the linear noise term $\boldsymbol{g}_t^\top \bar{\boldsymbol{\delta}}_{N,t}$ to vanish. Similarly, $\hat{\boldsymbol{\Sigma}}_{N,t} \xrightarrow{P} \boldsymbol{\Sigma}_t$. By the Continuous Mapping Theorem, $\frac{1}{2}\mathrm{Tr}\left(\mathbf{H}_t \hat{\boldsymbol{\Sigma}}_{N,t}\right) \xrightarrow{P} \frac{1}{2}\mathrm{Tr}(\mathbf{H}_t \boldsymbol{\Sigma}_t)$. Due to local convexity ($\mathbf{H}_t \succeq 0$), this limit is strictly non-negative, acting as a penalty on aleatoric (physics-induced) variance. ∎

**Implication.** Substituting this local result back into Eq. (27) proves that DRIS improves the Signal-to-Noise Ratio (SNR) of the global optimization.

- **At $N = 1$ (Conventional DR):** The objective is dominated by linear noise $\sum \gamma^t \boldsymbol{g}_t^\top \bar{\boldsymbol{\delta}}_{1,t}$. The optimizer "chases" random first-order fluctuations, which can obscure the variance penalty.
- **At $N \gg 1$ (DRIS):** The linear noise is suppressed by $\mathcal{O}\left(1/\sqrt{N}\right)$. The global objective behaves as:

$$\hat{\mathcal{J}}_N(\boldsymbol{\theta}) \approx -\sum_{t=0}^{T-1} \gamma^t \left[ C_t(\boldsymbol{\mu}_t) + \frac{1}{2}\mathrm{Tr}(\mathbf{H}_t \boldsymbol{\Sigma}_t) \right] \tag{31}$$

  This steers the policy to update in directions that reduce the physics-induced variance $\boldsymbol{\Sigma}_t$.

As a result, the learned policy via DRIS favors state-action regions that are invariant (i.e., robust) to physical uncertainty.

### *E. Sim-to-Real Generalization*

We provide a theoretical analysis of the sim-to-real transfer capability of DRIS. We explicitly account for the distribution shift between the physics in simulation and the true physics of the real world.

Recall that given a fixed initial state $\boldsymbol{s}_0$, for physics parameter $\boldsymbol{c}$, we let $\tau_{\boldsymbol{c}}$ denote the trajectory induced by policy $\pi_{\boldsymbol{\theta}}$. As defined earlier in Eq. (15), the discounted return of the trajectory is $R(\tau_{\boldsymbol{c}})$. We equivalently define the trajectory cost as the negative return:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c}) \coloneqq -R(\tau_{\boldsymbol{c}}) = -\sum_{t=0}^{T-1} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \tag{32}$$

Maximizing return is therefore equivalent to minimizing trajectory cost. If rewards are bounded as $|r(\boldsymbol{s}, \boldsymbol{a})| \le r_{\max}$, then

$$|\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c})| \le \frac{r_{\max}}{1-\gamma} =: B \tag{33}$$

**Source vs. Target Distributions.** We distinguish between two distributions over the physics parameters $\mathcal{C}$:

- **Source Distribution** with density $p_S(\cdot)$: The distribution used in the simulator for learning (e.g., uniform).
- **Target Distribution** with density $p_T(\cdot)$: The unknown true distribution of physical parameters in the real world.

We seek to minimize the expected cost in the real world:

$$\mathcal{J}_T(\boldsymbol{\theta}) \coloneqq \mathbb{E}_{\boldsymbol{c} \sim p_T(\boldsymbol{c})}[\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c})] \tag{34}$$

The DRIS algorithm minimizes the empirical cost drawn from $p_S(\cdot)$, averaged over the $N$ instances inside DRIS:

$$\hat{\mathcal{J}}_N(\boldsymbol{\theta}) \coloneqq \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\theta, \boldsymbol{c}^{(i)}), \quad \text{where } \boldsymbol{c}^{(i)} \overset{\text{i.i.d.}}{\sim} p_S(\boldsymbol{c}) \tag{35}$$

Let $\mathcal{J}_S(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{c} \sim p_S(\boldsymbol{c})}[\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c})]$ be the expected cost in simulation.

**Definition Appendix B.2** (Rademacher Complexity)**.** *To quantify the capacity of the policy parameter class $\boldsymbol{\Theta}$ to fit random fluctuations in the simulation, we introduce the **empirical Rademacher Complexity** of the induced (trajectory) cost function class $\mathcal{L}_{\boldsymbol{\Theta}} = \{\mathcal{L}(\boldsymbol{\theta}, \cdot) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$. Given a sample $\hat{\mathcal{C}} = \{\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(N)}\}$, it is defined as:*

$$\hat{\mathfrak{R}}_{\hat{\mathcal{C}}}(\mathcal{L}_{\boldsymbol{\Theta}}) \coloneqq \mathbb{E}_{\boldsymbol{\sigma}}\left[ \sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \frac{1}{N}\sum_{i=1}^{N} \sigma_i \mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{c}^{(i)}\right) \right], \tag{36}$$

*where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)$ are independent Rademacher variables with $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = \frac{1}{2}$. The **expected Rademacher Complexity** is then defined by averaging over the sampling of $\hat{\mathcal{C}}$:*

$$\mathfrak{R}_N(\mathcal{L}_{\boldsymbol{\Theta}}) \coloneqq \mathbb{E}_{\hat{\mathcal{C}} \sim p_S(\boldsymbol{c})^N}\left[ \hat{\mathfrak{R}}_{\hat{\mathcal{C}}}(\mathcal{L}_{\boldsymbol{\Theta}}) \right]. \tag{37}$$

**Theorem Appendix B.2** (Sim-to-Real Transfer Bound)**.** *For any $\delta \in (0,1)$, with probability at least $1-\delta$ over the draw of DRIS $\hat{\mathcal{C}} \sim p_S(\boldsymbol{c})^N$, the following bound holds for all policy parameter $\boldsymbol{\theta} \in \boldsymbol{\Theta}$:*

$$\mathcal{J}_T(\boldsymbol{\theta}) \le \hat{\mathcal{J}}_N(\boldsymbol{\theta}) + \underbrace{2\mathfrak{R}_N(\mathcal{L}_{\boldsymbol{\Theta}}) + 2B\sqrt{\frac{\ln(1/\delta)}{2N}}}_{\textit{Generalization Gap (Reducible via } N)} + \underbrace{d_{\mathcal{L}_{\boldsymbol{\Theta}}}(p_S, p_T)}_{\textit{Physics Mismatch (Irreducible)}} \tag{38}$$

*where $d_{\mathcal{L}_{\boldsymbol{\Theta}}}(p_S, p_T)$ is the Integral Probability Metric (IPM) measuring the discrepancy between source and target distributions over physics parameter (i.e., $\mathcal{C}$) with respect to the cost class induced by the policy family.*

$$d_{\mathcal{L}_{\boldsymbol{\Theta}}}(p_S, p_T) \coloneqq \sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left| \int_{\mathcal{C}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c}) \, (p_T(\boldsymbol{c}) - p_S(\boldsymbol{c})) \, d\boldsymbol{c} \right| \tag{39}$$

*Proof:* Applying the triangle inequality, we decompose the sim-to-real error into two distinct components: an *estimation error* (finite sampling) and a *transfer error* (distribution shift):

$$\begin{aligned} \mathcal{J}_T(\boldsymbol{\theta}) &= \hat{\mathcal{J}}_N(\boldsymbol{\theta}) + \left(\mathcal{J}_S(\boldsymbol{\theta}) - \hat{\mathcal{J}}_N(\boldsymbol{\theta})\right) + \left(\mathcal{J}_T(\boldsymbol{\theta}) - \mathcal{J}_S(\boldsymbol{\theta})\right) \\ &\le \hat{\mathcal{J}}_N(\boldsymbol{\theta}) + \underbrace{\left|\mathcal{J}_S(\boldsymbol{\theta}) - \hat{\mathcal{J}}_N(\boldsymbol{\theta})\right|}_{\text{Estimation Error (Step 1)}} + \underbrace{\left|\mathcal{J}_T(\boldsymbol{\theta}) - \mathcal{J}_S(\boldsymbol{\theta})\right|}_{\text{Transfer Error (Step 2)}} \end{aligned} \tag{40}$$

**Step 1: Bounding the Estimation Error.** We seek to bound $\sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left|\mathcal{J}_S(\boldsymbol{\theta}) - \hat{\mathcal{J}}_N(\boldsymbol{\theta})\right|$. Let us define a deviation

$$\Phi(\hat{\mathcal{C}}) = \sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left(\mathcal{J}_S(\boldsymbol{\theta}) - \hat{\mathcal{J}}_N(\boldsymbol{\theta})\right) \tag{41}$$

Consider two sample sets $\hat{\mathcal{C}} = \{\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(i)}, \ldots, \boldsymbol{c}^{(N)}\}$ and $\hat{\mathcal{C}}' = \{\boldsymbol{c}^{(1)}, \ldots, \tilde{\boldsymbol{c}}^{(i)}, \ldots, \boldsymbol{c}^{(N)}\}$ that differ by exactly one sample at index $i$. The difference in the empirical cost is at most $2B/N$:

$$\begin{aligned}
&\left|\hat{\mathcal{J}}_N(\boldsymbol{\theta}) - \hat{\mathcal{J}}_{N'}(\boldsymbol{\theta})\right| \\
=&\left|\frac{1}{N}\sum_{j=1}^{N}\mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{c}^{(j)}\right) - \frac{1}{N}\left(\mathcal{L}\left(\boldsymbol{\theta}, \tilde{\boldsymbol{c}}^{(i)}\right) + \sum_{j\neq i}\mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{c}^{(j)}\right)\right)\right| \\
=&\frac{1}{N}\left|\mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{c}^{(i)}\right) - \mathcal{L}\left(\boldsymbol{\theta}, \tilde{\boldsymbol{c}}^{(i)}\right)\right| \\
\leq&\frac{1}{N}\left(\left|\mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{c}^{(i)}\right)\right| + \left|\mathcal{L}\left(\boldsymbol{\theta}, \tilde{\boldsymbol{c}}^{(i)}\right)\right|\right) = \frac{2B}{N}
\end{aligned} \tag{42}$$

Because the supremum is a contraction, the function $\Phi(\hat{\mathcal{C}})$ satisfies the bounded difference condition (i.e., substituting the value of $\boldsymbol{c}^{(i)}$ changes the value of $\Phi(\hat{\mathcal{C}})$ by at most $2B/N$):

$$\left|\Phi(\hat{\mathcal{C}}) - \Phi(\hat{\mathcal{C}}')\right| \leq \sup_{\boldsymbol{\theta}}\left|\hat{\mathcal{J}}_N(\boldsymbol{\theta}) - \hat{\mathcal{J}}_{N'}(\boldsymbol{\theta})\right| \leq \frac{2B}{N}. \tag{43}$$

We now apply McDiarmid's Inequality:

$$\begin{aligned}
\mathbb{P}\left(\Phi(\hat{\mathcal{C}}) - \mathbb{E}[\Phi(\hat{\mathcal{C}})] \geq \epsilon\right) &\leq \exp\left(\frac{-2\epsilon^2}{N(2B/N)^2}\right) \\
&= \exp\left(\frac{-N\epsilon^2}{2B^2}\right)
\end{aligned} \tag{44}$$

Setting the right-hand side to $\delta$ and solving for $\epsilon$, we obtain the concentration bound with:

$$\epsilon = 2B\sqrt{\frac{\ln(1/\delta)}{2N}}. \tag{45}$$

Finally, we bound the expectation $\mathbb{E}[\Phi(\hat{\mathcal{C}})]$ via standard symmetrization. Let $\hat{\mathcal{C}}_g$ be an independent ghost sample and $\sigma_i$ i.i.d. Rademacher variables. Then:

$$\begin{aligned}
\mathbb{E}_{\hat{\mathcal{C}}}[\Phi(\hat{\mathcal{C}})] &= \mathbb{E}_{\hat{\mathcal{C}}}\left[\sup_{\boldsymbol{\theta}}\mathbb{E}_{\hat{\mathcal{C}}_g}\left[\hat{\mathcal{J}}_{N_g}(\boldsymbol{\theta}) - \hat{\mathcal{J}}_N(\boldsymbol{\theta})\right]\right] \\
&\leq \mathbb{E}_{\hat{\mathcal{C}},\hat{\mathcal{C}}_g}\left[\sup_{\boldsymbol{\theta}}\frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c}_g^{(i)}) - \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c}^{(i)})\right)\right] \\
&\leq 2\mathbb{E}_{\hat{\mathcal{C}},\sigma}\left[\sup_{\boldsymbol{\theta}}\frac{1}{N}\sum_{i=1}^{N}\sigma_i\,\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c}^{(i)})\right] = 2\mathfrak{R}_N\left(\mathcal{L}_{\boldsymbol{\Theta}}\right)
\end{aligned} \tag{46}$$

Combining the concentration and expectation bounds yields the result for Step 1:

$$\begin{aligned}
\left|\mathcal{J}_S(\boldsymbol{\theta}) - \hat{\mathcal{J}}_N(\boldsymbol{\theta})\right| &\leq \sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\left|\mathcal{J}_S(\boldsymbol{\theta}) - \hat{\mathcal{J}}_N(\boldsymbol{\theta})\right| \\
&\leq 2\mathfrak{R}_N\left(\mathcal{L}_{\boldsymbol{\Theta}}\right) + 2B\sqrt{\frac{\ln(1/\delta)}{2N}}
\end{aligned} \tag{47}$$

**Step 2: Bounding the Transfer Error (Physics Mismatch).** We must bound the discrepancy $|\mathcal{J}_T(\boldsymbol{\theta}) - \mathcal{J}_S(\boldsymbol{\theta})|$ caused by the shift from $p_S(\cdot)$ to $p_T(\cdot)$ by:

$$\begin{aligned}
&|\mathcal{J}_T(\boldsymbol{\theta}) - \mathcal{J}_S(\boldsymbol{\theta})| \\
\leq&\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}|\mathcal{J}_T(\boldsymbol{\theta}) - \mathcal{J}_S(\boldsymbol{\theta})| \\
=&\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\left|\mathbb{E}_{\boldsymbol{c}\sim p_T(\boldsymbol{c})}[\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c})] - \mathbb{E}_{\boldsymbol{c}\sim p_S(\boldsymbol{c})}[\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c})]\right| \\
=&\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\left|\int_{\mathcal{C}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c})\,p_T(\boldsymbol{c})\,d\boldsymbol{c} - \int_{\mathcal{C}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{c})\,p_S(\boldsymbol{c})\,d\boldsymbol{c}\right| \\
=&d_{\mathcal{L}_{\boldsymbol{\Theta}}}(p_S, p_T)
\end{aligned} \tag{48}$$

where $d_{\mathcal{L}_{\boldsymbol{\Theta}}}(p_S, p_T)$ is the IPM defined in Eq. (39), measuring the worst-case performance gap over the hypothesis (policy parameter) space $\boldsymbol{\Theta}$. Unlike the estimation error, this does not vanish with $N \to \infty$.

**Conclusion.** Substituting the bounds from Step 1 and Step 2 back into the decomposition, we obtain:

$$\mathcal{J}_T(\boldsymbol{\theta}) \leq \hat{\mathcal{J}}_N(\boldsymbol{\theta}) + \left(2\mathfrak{R}_N + 2B\sqrt{\frac{\ln(1/\delta)}{2N}}\right) + d_{\mathcal{L}_{\boldsymbol{\Theta}}}(p_S, p_T). \tag{49}$$

■

**Remark:** Note that if the simulator cannot model the real world (e.g., $p_T$ is disjoint from $p_S$), the discrepancy term $d_{\mathcal{L}_{\boldsymbol{\Theta}}}$ is large and irreducible with $N$. However, compared to conventional DR ($N$=1), DRIS ($N \gg 1$) minimizes the first term (Generalization Gap), ensuring that the policy is at least robust to the *modeled* uncertainty.