Dynamics as Prompts: In-Context Learning for Sim-to-Real System Identifications

Shiqi Liu[1*], Xilun Zhang[1*], Peide Huang[1], William Jongwon Han[1], Yiqi Lyu[1], Mengdi Xu[1], Ding Zhao[1]

[1]Carnegie Mellon University
{shiqiliu, xilunz, peideh, wjhan, yiqilyu, mengdixu, dingzhao}@andrew.cmu.edu

Abstract

Sim-to-real transfer remains a significant challenge in robotics due to the discrepancies between simulated and real-world dynamics. Traditional methods like Domain Randomization often fail to capture fine-grained dynamics, limiting their effectiveness for precise control tasks. In this work, we propose a novel approach that dynamically adjusts simulation environment parameters online using in-context learning. Using past interaction histories as context, our method adapts the simulation environment dynamics to real-world dynamics without requiring gradient updates, resulting in faster and more accurate alignment between simulated and real-world performance. We validate our approach across two tasks: object scooping and table air hockey. In the sim-to-sim evaluations, our method significantly outperforms the baselines on environment parameter estimation by 80% and 42% in the object scooping and table air hockey setups, respectively. Furthermore, our method achieves at least 70% success rate in sim-to-real transfer on object scooping across three different objects. By incorporating historical interaction data, our approach delivers efficient and smooth system identification, advancing the deployment of robots in dynamic real-world scenarios.

Project Website — https://sim2real-capture.github.io/

1 Introduction

Learning-based methods like deep Reinforcement Learning (RL) allow robots to tackle complex tasks in areas such as object manipulation (Peng et al. 2018; Lin, Corcodel, and Zhao 2024) and locomotion for quadrupedal robots (Li et al. 2024; Kumar et al. 2021) and humanoids (Chen et al. 2024; Zhang et al. 2024). However, RL's high sample complexity and risks of unsafe exploration (Xu et al. 2022a; Wang et al. 2023b; Yao et al. 2024) make it necessary to train policies in simulations and then deploy in the real world. A key challenge is the sim-to-real gap, caused by discrepancies between simulated and real-world dynamics (Hu et al. 2024; Torne et al. 2024; Huang et al. 2022), which can lead to catastrophic failures during deployment.

Traditional sim-to-real approaches aim to develop robust policies by randomizing environment parameters during training, known as Domain Randomization (DR) (Peng et al. 2018; Mehta et al. 2020). While effective in some cases (Mehta et al. 2020; Li et al. 2024), DR captures only average dynamics, limiting precision in fine-grained control tasks. In contrast, System Identification (SysID) methods aim to align the simulation and real-world performance through actively adjusting the simulation environment parameters, which often requiring iterative SysID model updates to test new parameters (Ramos, Possas, and Fox 2019; Huang et al. 2023). For instance, in a kitchen environment, when a robot tries to scoop grilled celery from a pan (Figure 1), traditional offline SysID methods would involve learning a new SysID model that predict the center of mass of the celery at each iteration, making the process time-consuming and inefficient. Humans, on the other hand, can quickly adapt online. A more intuitive solution is to develop a model with online SysID, allowing for more efficient parameter estimation across different environment dynamics.

In-context learning has gained traction as a method for adjusting model behavior without gradient updates, widely used in Natual Language Processing (NLP) (Dong et al. 2022) and recently applied in robotics to improve generalization (Laskin et al. 2022; Grigsby, Fan, and Zhu 2024; Xu et al. 2022b). For example, Xu et al. enhanced the Decision Transformer (DT) (Chen et al. 2021) by using new task demonstrations as prompts for online adaptation. Most current in-context learning approaches focus on adapting policies when rewards or expert demonstrations changes, assuming fixed environment dynamics (Grigsby, Fan, and Zhu 2024; Fu et al. 2024). While different dynamics could be framed as diverse tasks in a multi-task RL setting, more than it's a counter-intuitive setting, it also becomes impractical with a high-dimensional continuous environment parameter space, requiring many tasks to capture the full range of behaviors. In this paper, we explore a novel question: "*Can we adapt simulation environment parameters using the in-context learning paradigm?*" Our goal is to eliminate the optimization loop in SysID, in order to accelerate the parameter estimation process by incorporating the in-context learning ability of transformer models.

We introduce in-**C**ontext **A**da**PT**ation mod**U**le for sim-to-**RE**al system identification, or CAPTURE, to bridge the

[*]These authors contributed equally.

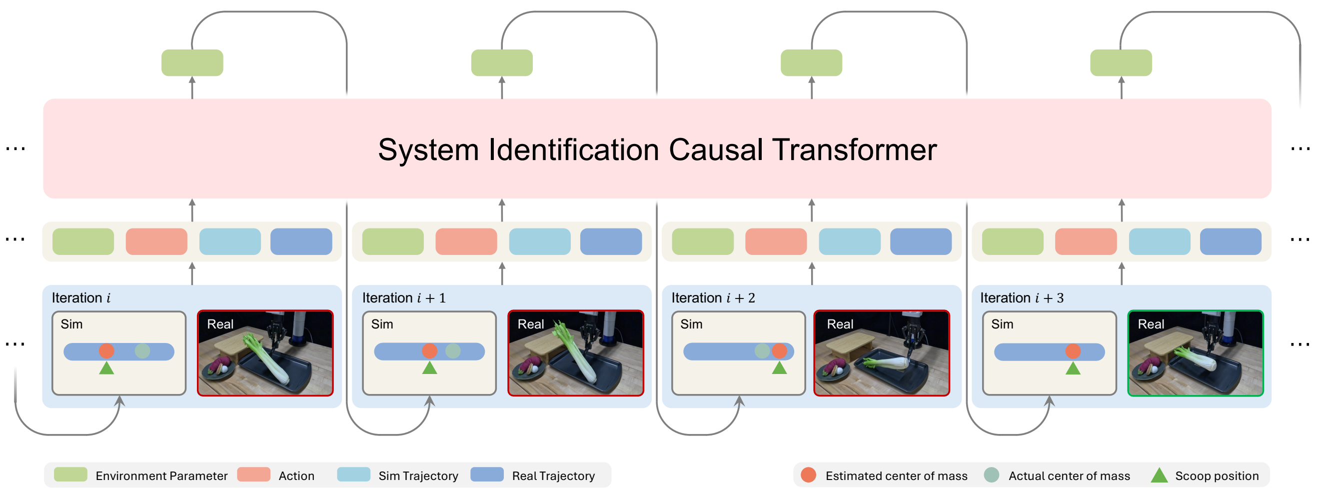


Figure 1: CAPTURE aims to take the history information to predict the next step environment parameters. SysID causal transformer adapts the simulation environment parameters to match the real-world performance on the fly via next-token prediction. CAPTURE takes three iterations to identify the correct center of mass of celery.

sim-to-real gap. CAPTURE aims to dynamically adjust the environment parameters online to align simulated and real-world trajectories using next-token prediction based on past interaction data, which includes simulated trajectories, actions, environment parameters, and real-world trajectories as shown in Figure 1. CAPTURE frames the SysID problem as an in-context learning formulation, treating interaction histories as "*context*." Unlike existing techniques (Kumar et al. 2021, 2022; Ren et al. 2023) that rely on short state-action history, CAPTURE aim to learn the complex SysID search process itself through rich and multi-episodic interaction history data. Beyond learning single-step expert parameter matching behaviors, longer interaction histories enables the learned SysID causal transformer to capture a better dynamics representation of the environments. By incorporating in-context learning, CAPTURE provides a smoother and more accurate prediction of subsequent environmental parameters and dynamic behaviors.

In summary, this study makes the following contributions:

1. We propose a novel method that can identify real-world environment parameters without any network parameter updates using in-context learning.
2. CAPTURE distills the SysID parameter update process using multi-episode history, rather than relying on a single-step behavior-to-parameter mapping. This approach allows the SysID causal transformer to learn more comprehensive dynamics properties through interactions, which baseline methods struggle to capture.
3. We evaluate CAPTURE in two experiments, object scooping and table air hockey, where we report substantial performance increases in both sim-to-sim transfer and sim-to-real transfer.

2 Related Work

Sim-to-real transfer is a pivotal area of robotics research, focusing on the application of simulation-trained models to real-world tasks. DR involves injecting variability into the parameters of the simulation environment regarding dynamical or visual attributes (Peng et al. 2018; Mehta et al. 2020), but struggle with over-conservative or average task behaviours. In the following subsections, prior works on SysID for domain adaptation and in-context learning will be discussed in more detail.

2.1 Sim-to-Real SysID for Domain Adaptation

There are two primary approaches on SysID for sim-to-real transfer: offline and online. Offline SysID typically requires iterative refinement of the identification module through repeated training cycles(Huang et al. 2023; Murooka et al. 2021; Chebotar et al. 2019; Ramos, Possas, and Fox 2019; Muratore et al. 2022; Lim et al. 2022). In contrast, online SysID focuses on the determination of environment parameters or latent variables without the need for model updates. This approach has proven effective in highly dynamic systems, employing strategies such as RMA (Kumar et al. 2021), which leverages short-term historical state-action pairs to infer environment dynamics (Yu et al. 2017; Evans, Thankaraj, and Pinto 2022; Kumar et al. 2022; Allevato et al. 2020). Memmel et al. describes exploring the object dynamics through curiosity-driven exploration first and then deploying on the task environment.Ren et al. propose a meta-learning framework, prioritizing task-specific adaptation over simple trajectory alignment. In addition to aligning environment parameters, Jiang et al. introduced a human-in-the-loop correction method to mitigate the sim-to-real gap. More recently, Dai et al. proposed reconstructing real-world environmental variations in simulation to enhance the generalizability on real-world policy deployment. Most relevant to our work, IIDA (Evans, Thankaraj, and Pinto 2022) uses long-term historical state action pairs to infer latent real-world dynamic models. In contrast, our method focuses on distilling the sim-to-real parameter update process to create

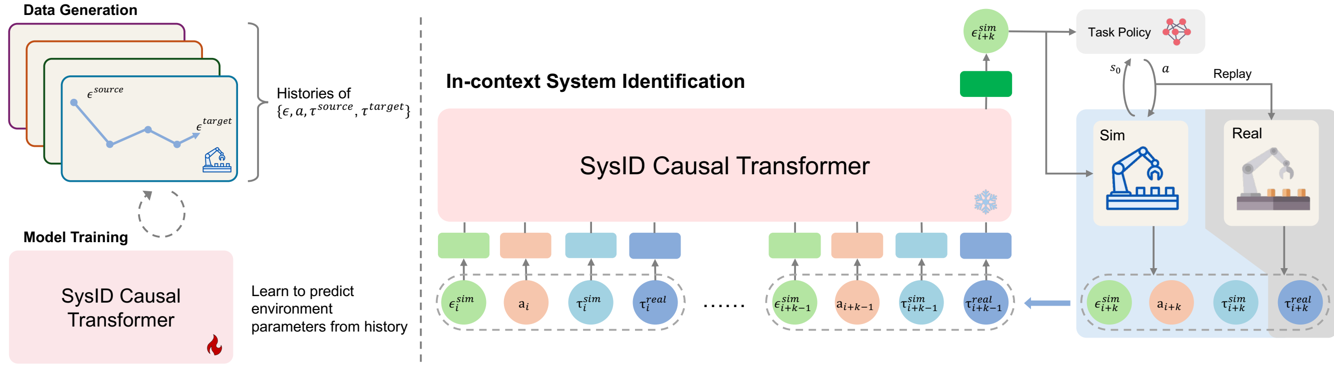


Figure 2: System overview: training and inference pipeline. The SysID causal transformer is trained with multi-episodic parameter update histories. During the in-context SysID, it will take the interaction history as context, and iteratively update the environment parameters online through a task policy rollout in both simulation and the real world. The SysID causal transformer will maintain a fixed-length context window, where in our setting, the length is 4.

more accurate simulation environments, effectively closing the sim-to-real gap.

2.2 In-context Learning in Robotics

In-context learning has garnered significant attention in NLP (Dong et al. 2022; Krishnamurthy et al. 2024) and computer vision (Wang et al. 2023a) due to its remarkable ability to infer tasks from context. This ability to infer tasks through contextual information, such as expert demonstrations, allows for adaptation to new tasks without updating the model's weights (Min et al. 2022), which has been shown to be beneficial in robotics settings (Xu et al. 2022b, 2023; Zhu et al. 2024; Di Palo and Johns 2024; Yu et al. 2024; Jiang, Ke, and van Hasselt 2023). The potential of in-context learning for generalizing to unseen tasks has been further explored in recent studies. Laskin et al. employed transformer models to distill the RL learning history, showing RL algorithms can be distilled into transformer models and successfully in-context adapt to new goal settings (Grigsby, Fan, and Zhu 2024). Previous work on in-context adaptation has either focused on RL algorithm distillation or policy generalization abilities, where CAPTURE focuses on learning environment parameters through interaction histories.

3 Methodology

Rather than directly adapting the task policy, we prioritize leveraging historical data—including past environment parameters, task state trajectories, and task actions—to estimate next-iteration environment parameters. This approach aims to align simulation dynamics with real-world performance. We assume that as the discrepancy between simulation and real-world environment parameters decreases, the sim-to-real performance gap will naturally narrow. This process is guided by the underlying monotonic properties of the environment parameter adjustments. We start with a description of the problem formulation in Section 3.1. Following with three key modules in our pipeline: Section 3.2 describes the task policy training, Section 3.3 describes how we generate efficient source-to-target adaptation iterations, and Section 3.4 defines different components in the SysID causal transformer structure. The main components of CAPTURE pipeline is demonstrated in Figure 2, where it consists the data generation, model training, and inference pipeline.

3.1 Problem Formulation

In this section, we define the task of predicting accurate simulation environment parameters to align simulated dynamics with real-world environments. We outline how to model the interaction between dynamics behaviours and environment parameters as a sequence, forming the training data for SysID causal transformer models. We begin by introducing the simulation parameters, followed by the task policy and data generation notations, and conclude with the SysID causal transformer notations for domain adaptation.

Environment Parameter Space. We define the task-related environment parameter space $\epsilon \in \mathcal{E}$ that parameterized quantities such as the center of mass and sliding frictions. We also assume that the environment parameter space $\mathcal{E}$ is finite and bounded, encompassing properties of different objects. We modify the environment parameters with Robosuite (Zhu et al. 2020), which provides API for modifying the environment parameters through Python code.

SysID Causal Transformer and Interaction Histories. During the SysID causal transformer and data collection setting, we use previous SysID iterations as context, including simulated state trajectories $\tau^{sim} = \{s_0^{sim}, s_1^{sim}, \ldots, s_T^{sim}\}$, real state trajectories $\tau^{real} = \{s_0^{real}, s_1^{real}, \ldots, s_T^{real}\}$, rollout action $a \sim \pi(a|s_0, \epsilon)$, and the past environment parameters ϵ. A robust SysID process explores complex parameter behaviours. We aim to capture this behavior using a causal transformer that infers parameters from past interactions. Following (Laskin et al. 2022), we treat these sequential interactions as *history*, where current environment parameters depend on previous SysID iterations. Formally, we define the *history* as:

$$h_i := \left(\epsilon_{i-k}^{sim}, a_{i-k}, \tau_{i-k}^{sim}, \tau_{i-k}^{real}, \ldots, \epsilon_{i-1}^{sim}, a_{i-1}, \tau_{i-1}^{sim}, \tau_{i-1}^{real}\right) \tag{1}$$

where h_i is the history containing the past k iterations at i-th iteration. Our goal is to learn a causal transformer such that it can replicate the SysID process given history. We define the SysID causal transformer, P_θ, with the objective of modeling the distribution of simulation parameters conditioned on the history. This encourages the simulated trajectories τ^{sim} eventually behave close to real-world trajectories, τ^{real}, bridge the sim-to-real gap.

Given an ideal search strategy that successfully adapts to the target environment parameters, our goal is to learn the underlying search capabilities from this process by predicting the next iteration in the history. The optimization objective can be formalized as:

$$\theta^* = \arg\min_\theta \left[\mathcal{L}\left(P_\theta(h_i), \epsilon_i^{sim}\right)\right] \tag{2}$$

where $P_\theta(h_i)$ represents the predicted next environment parameters from the model, $\mathcal{L}(\cdot)$ is the Mean-Square-Error (MSE) loss function that measures the discrepancy between the predicted and the ground-truth next-iteration environment parameters.

3.2 Environment-Conditioned RL Training

The environment-conditioned RL task policy $\pi(a \mid s_0, \epsilon)$ is trained to adapt to varying environment parameters $\epsilon \in \mathcal{E}$. For each episode, ϵ is sampled uniformly from the parameter space $\mathcal{E}$. Within the episode, the agent selects an action a from $\pi(a \mid s_0, \epsilon)$, considering the initial state s_0 and environment parameter ϵ. This action is executed, producing a state trajectories $\{s_1, s_2, \ldots, s_T\}$ and a reward r. Each episode $\{a, r, s_0, \epsilon\}$, is stored in the replay buffer. After certain episodes, the policy is updated using Soft Actor-Critic (SAC) (Haarnoja et al. 2018), refining actions for smoother domain adaptation with predicted parameters.

3.3 Source-to-Target SysID Iteration Generation

In the data generation process, we develop source-to-target adaptation transitions that mimic sim-to-real adaptation. Each iteration includes four elements: the current simulation parameter ϵ_i, the policy action a_i, the simulated trajectories τ_i^{source}, and the collected target environment trajectories τ_i^{target} under the same action a_i. The trajectories and actions are obtained through simulation rollouts using an environment-conditioned task policy.

In simulation, both source and target values are known, allowing for direct single-step mapping from source to target. However, this approach often performs poorly in real-world deployment when the target's dynamics representation (state trajectories) lacks sufficient detail. Rather than learning a single-step mapping, we focus on learning a search algorithm that finds the target environment parameter with dynamic representations. The duration of the parameter iteration history L indicates the number of iterations that we pre-defined to generate a complete transition sequence from ϵ^{source} to ϵ^{target}. We pick a transition number $L = 7$ during data generation.

In the sim-to-real SysID setting, a search algorithm must balance exploration and precision, as it lacks the ground-truth target value and relies only on performance labels (higher or lower). Linear interpolation is suboptimal here because it limits exploration during adaptation. To overcome this, we propose emulating a randomized binary search process (Martínez and Roura 1998), which optimally navigates a constrained space by dynamically adjusting the upper and lower search bounds at each iteration. To further promote exploration, we use a beta distribution when selecting the environment parameters for the next iteration. An ablation study is discussed in Section 4.2 on how different search algorithms impact parameter estimation. The transition iteration generation process is illustrated in Figure 3, and the formal pseudocode is described in Algorithm 1.

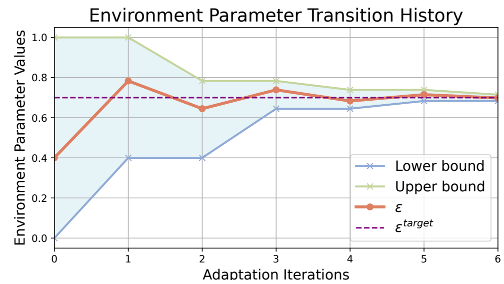


Figure 3: A environment parameter transition history from ϵ^{source} to ϵ^{target}, with gradually shrank upper and lower bounds of the search space.

Algorithm 1: Source-to-Target SysID Iteration Generation

Initialize data buffer $\mathcal{D}$
Choose parameter transition iteration length L
Choose symmetric beta distribution parameter α
for $n = 1$ to N **do** ▷ This loop can be run in parallel
  Sample $\epsilon^{\text{source}}, \epsilon^{\text{target}}$ from space $\mathcal{E}$
  Let l be the dynamic lower bound of space $\mathcal{E}$
  Let u be the dynamic upper bound of space $\mathcal{E}$
  Set $\epsilon_0 = \epsilon^{\text{source}}$
  for $i = 0$ to L **do**
    Sample action $a_i \sim \pi(a_i \mid s_0, \epsilon_i)$
    $\tau_i^{\text{source}} \leftarrow$ rollout in $sim(\epsilon_i)$ with a_i
    $\tau_i^{\text{target}} \leftarrow$ rollout in $sim(\epsilon^{\text{target}})$ with a_i
    for $j = 1$ to $\dim(\mathcal{E})$ **do**
      if $\epsilon_i[j] < \epsilon^{\text{target}}[j]$ **then**
        Update lower bound: $l[j] = \epsilon_i[j]$
      else
        Update upper bound: $u[j] = \epsilon_i[j]$
      Sample r from $\mathrm{B}(\alpha, \alpha)$
      Set $\epsilon_{i+1}[j] = r\,(u[j] - l[j]) + l[j]$
    Store trajectory $h[i] = \{\epsilon_i, a_i, \tau_i^{\text{source}}, \tau_i^{\text{target}}\}$
  Update data buffer: $\mathcal{D} \leftarrow \mathcal{D} \cup h$

3.4 SysID Causal Transformer

Given the collected SysID parameter transition histories, $\mathcal{D}$, our goal is to distill the binary search process through parameter transition sequences with length L, where each iteration represents an adaptation iteration. The model pre-

Algorithm 2: SysID Causal Transformer Training and Evaluation

Environment-conditioned task policy π
Collected SysID transition history buffer $\mathcal{D}$
Initialize SysID causal transformer P_θ
Initialize SysID causal transformer window size k
// SysID causal transformer training
while P_θ not converged **do**
    Sample multi-episodic k subsequence from $\mathcal{D}$:
$$h_i = \left(\epsilon_{i-k}^{sim}, a_{i-k}, \tau_{i-k}^{sim}, \tau_{i-k}^{real}, \ldots, \epsilon_i, a_i, \tau_i^{sim}, \tau_i^{real}\right)$$
    Calculate shifted input loss $||P_\theta(h_{i-1}) - \epsilon_i||_2$
    Backpropagate to update P_θ
// In-context SysID with env-conditioned policy
for $i = 0, \ldots, MaxIters$ **do**
    $\tau_i^{sim} \leftarrow$ rollout $a_i \sim \pi(a_i \mid s_0, \epsilon_i)$ in $sim(\epsilon_i)$
    $\tau_i^{real} \leftarrow$ rollout a_i in unknown real environment
    Predict $\epsilon_{i+1} = P_\theta(\{\epsilon_x, a_x, \tau_x^{sim}, \tau_x^{real}\}_{x=i}^{i-k})$

dicts the next environment parameter $\hat{\epsilon}_{i+1}$ at iteration i using a next-token prediction framework with a shifted input setup (Radford et al. 2019). We sample a multi-episode window of size k from $\mathcal{D}$, where k is a subsequence of the full L iterations. The SysID causal transformer processes this history to predict the next parameter. Each iteration block contains $2 + 2T$ tokens: one action, one parameter, and T state trajectory tokens for both simulated and real rollouts.

During rollout, the model attends to preceding tokens to predict ϵ_{i+1} using relative timestep embedding (Al-Rfou et al. 2019) to focus on subsequence order. Starting with initial tokens $\{\epsilon_0, a_0, \tau_0^{sim}, \tau_0^{real}\}$, we update actions with an environment-conditioned policy π in the new simulation ϵ_{i+1} and initial state, obtaining updated trajectories τ_{i+1}^{sim} and τ_{i+1}^{real}. The process is detailed in Algorithm 2.

4 Experiments

We conducted two sets of experiments to evaluate the performance of CAPTURE: object scooping and table air hockey. In both tasks, we demonstrated that CAPTURE significantly outperforms the baselines in both sim-to-sim and sim-to-real transfer scenarios. The experiment setups will be explained in Section 4.1, followed by descriptions of our baseline and ablation methods in Section 4.2. The sim-to-sim evaluation results compared with baselines and ablations results are detailed in Section 4.3, and the sim-to-real experiment results compared with baselines are described in Section 4.4.

4.1 Experimental Setups

We evaluate our algorithm using two tasks: object scooping and table air hockey. For object scooping, inspired by (Memmel et al. 2024; Shi et al. 2023), the goal is to identify the object's center of mass in kitchen scenarios, which often involve complex items like celery, carrots, and eggplants with varying centers of mass. We aim to determine the balance point for successful scooping through online interactions using CAPTURE.

Environment	Notion	Description	Range
Object Scooping	X_{com}	Object Center of Mass	[-1.0, 1.0]
Table Air Hockey	μ_{left}	Table Sliding Friction	[0.03, 0.07]
	μ_{right}	Table Sliding Friction	[0.03, 0.07]
	ζ_{mallet}	Mallet Damping	[-15, -3]
	ζ_{left}	Wall Damping	[-40, -3]
	ζ_{right}	Wall Damping	[-40, -3]

Table 1: Tunable Environment Parameters in Simulation

In table air hockey, we test the scalability of CAPTURE with a higher-dimensional parameter and action space (Huang et al. 2023; Chuck et al. 2024). This task requires precise control and adaptability to match simulated and real-world dynamics. Tunable environment parameters are listed in Table 1, with setups shown in Figure 4.

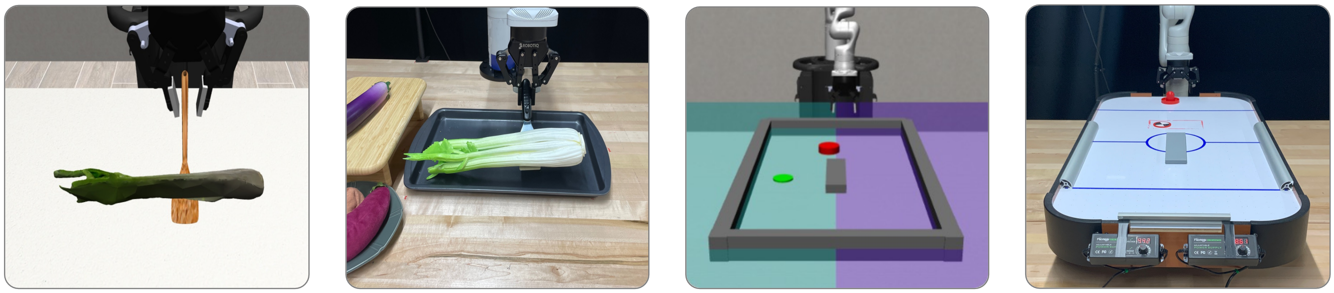
Object Scooping. In this task, our objective is to identify the optimal scooping points during food transfer from one toasting pan to another using a spatula. In this setting, CAPTURE needs to identify the center of mass noted as X_{com}, and then scoop at the corresponding placement such that the object can be balanced on the spatula. The range of X_{com} is defined based on the relative position of the objects, where -1.0 means the center of mass located at the most left of the object, and vice versa. To handle pose estimation uncertainties, a classifier labels the object as tilted left (-1), right (1), or balanced (0) and uses them as state trajectories.

Table Air-Hockey. The setup involves a robot-controlled mallet hitting a puck on an air-hockey table. The table is divided into left and right sections with different friction levels, causing varied puck behavior. We expect CAPTURE to learn surface friction and damping differences from both sides via incorporating context information. The five parameters considered are left-surface friction μ_{left}, right-surface friction μ_{right}, left-wall damping ζ_{left}, right-wall damping ζ_{right}, and puck damping ζ_{puck}. Lower absolute damping values make objects more responsive, and trajectory evaluation is based on the sum of point-wise L2 distances.

4.2 Baselines and Ablations

To discover how different module of CAPTURE affects the performances, the baselines aim to demonstrate the effectiveness of context history during rollout. The ablations are meant to demonstrate how different data generation methods affect the performance. We have compared CAPTURE with two ablations in sim-to-sim evaluation and three baselines methods in both sim-to-sim and sim-to-real evaluation.

Baselines. CAPTURE distills the sim-to-real adaptation process to learn an efficient transition from source to target. We compare with the following baselines for online adaptation tasks: Expert Distillation (ED) (Laskin et al. 2022), TuneNet (Allevato et al. 2020), and DR (Peng et al. 2018). ED is similar to CAPTURE but with expert SysID training data consists of one-iteration source-to-target parameter adaptation, rather than learning histories. To make a fair comparison, we have also implemented the TuneNet (Allevato et al. 2020) algorithm with a transformer backbone,



(a) Simulated Object Scooping (b) Real-world Object Scooping (c) Simulated Table Air Hockey (d) Real-world Table Air Hockey

Figure 4: Experiment setups for both object scooping and table air hockey.

where the model follows the ED setting but with residual parameter updates.

Ablations on Different Data Generation Approaches. We modify the data generation module to demonstrate the effectiveness of our distilled searching algorithm over others, including linear interpolation (linterp) and the standard binary search method without randomness (binary), while selecting the next iteration parameters. Linear interpolation randomly selects L points between source and target environment parameters and orderly constructs the SysID transition. The standard binary search method (Sikorski 1982) follows a similar setting as ours. However, it does not consider the random beta distribution, it only selects the middle point between the upper and lower bound.

4.3 Sim-to-Sim SysID Evaluation

In the sim-to-sim transfer, we evaluate whether CAPTURE can align trajectories by adjusting the environment parameters in-context without updating the model's parameters. We simulated 100 pairs of random environment parameters to mimic unknown real dynamics and test the performance across three seeds. For each pair, one simulation environment is designated as the "real" (target) environment, where only the dynamics performance is provided to the model, not the parameters. To improve parameter estimation independent of actions, we roll out the model with an environment-conditioned policy for online evaluation, as described in Section 3.2. In the results, baseline methods are shown with solid lines, while dashed lines indicate different ablation settings for data collection.

Object Scooping Sim-to-Sim Evaluation. In the sim-to-sim transfer, we evaluated the normalized context differences, which are one-dimensional in this setting, as shown in Figure 5. Since we use an angle classifier for smoother real-world deployment, reporting trajectory differences would be meaningless, as the trajectory here is represented by a label. Instead, we measure the task's success rate, defined as lift the object with label (0). Figure 5 shows that CAPTURE achieves a success rate 50% higher than other SysID methods and 70% higher than the DR approach. This is expected, as the baselines lack historical interaction data, making identification only dependent on current scooping points. In contrast, CAPTURE uses a rich previous interaction history, allowing it to gradually narrow down the center of mass search space.

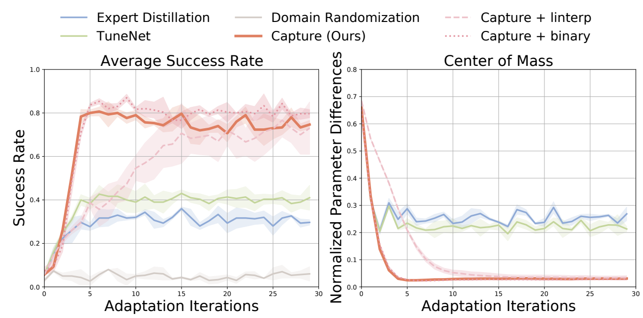


Figure 5: Object scooping sim-to-sim transfer parameter estimation and success rate performance. CAPTURE identifies objects' center of mass after around 4 iterations.

Table Air Hockey Sim-to-Sim Evaluation. In Figure 6, CAPTURE offers better parameter estimation with smoother and more accurate adaptation curves. In scenarios where environment parameters require rollout histories, baselines struggle due to their inability to account for historical interactions. For instance, while the ED method might successfully detect the left wall after hitting it, it tends to forget earlier right wall interactions. This short-term memory leads to faster adaptation in simple environments but falls short in more complex ones. In dynamic settings, where SysID needs to identify parameters on both sides for sustained task performance, maintaining a history of parameter updates becomes critical, as it informs subsequent iterations.

In Table 2, we show that with lower context differences between the source and target, the point-wise L2 trajectory distance also becomes smaller accordingly. CAPTURE are able to improve trajectory differences by about 40% compared to identification baselines, and 50% compared to DR. Given the parameter estimation error shown in Figure 6, the significant trajectory difference is expected from baseline methods.

Ablation results. In object scooping experiments, we observe that the linear interpolation approach converges more slowly in terms of adaptation iterations, as shown in Figure 5. Due to limited exploration, it hinders performance. Figure 6 shows that CAPTURE + linear interpolation fol-

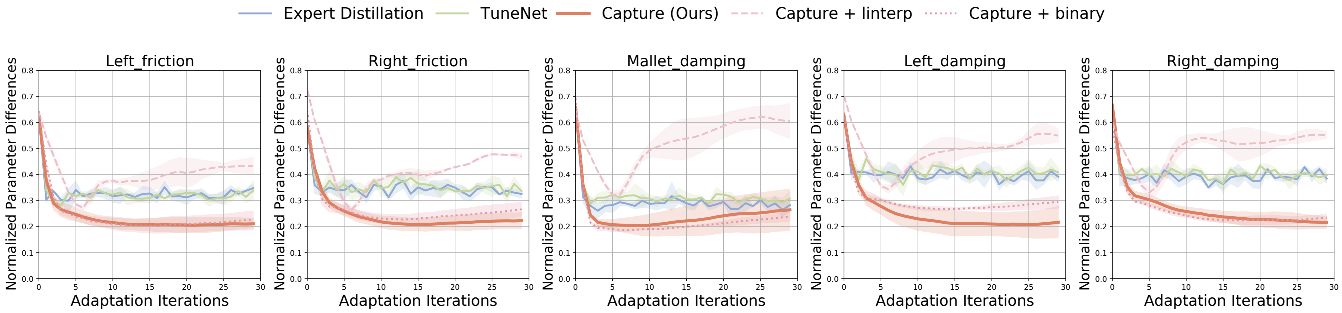


Figure 6: Table air hockey sim-to-sim transfer parameter estimation performances. The red lines represent our proposed method CAPTURE, which outperforms the baseline methods in all five parameters. Our approach reaches around 0.2 differences after 7 adaptation iterations, where the baselines converge at 0.35 for most parameters.

Method	Adaptation iterations				
	5 iterations	10 iterations	15 iterations	20 iterations	30 iterations
ED	0.25±0.01	0.26±0.03	0.27±0.02	0.27±0.01	0.26±0.01
DR	0.34±0.03	0.31±0.02	0.34±0.04	0.31±0.03	0.33±0.00
TuneNet	0.29±0.02	0.27±0.01	0.27±0.01	0.26±0.01	0.25±0.02
CAPTURE + linterp	0.22±0.02	0.23±0.01	0.26±0.03	0.24±0.02	0.22±0.01
CAPTURE + binary	0.20±0.01	0.17±0.01	0.16±0.02	0.17±0.01	0.18±0.01
CAPTURE	**0.20±0.01**	**0.16±0.01**	**0.14±0.01**	**0.14±0.01**	**0.15±0.01**

Table 2: Sim-to-Sim Table Air Hockey Trajectory Differences in Meters over 3 Seeds.

lows a near-linear sim-to-real transition until iteration 7, closely matching the dataset's transition history. However, it struggles to establish a robust search process due to overfitting to linear interpolated transition histories. Except for left damping parameter, no significant performance differences are seen between randomized binary search (ours) and standard binary search. With added randomized, it did not hinder the estimation performance, whereas it learned a more robust adaptation process. Figure 6 also illustrates that CAPTURE can smooth the adaptation process using history, regardless of SysID accuracy.

4.4 Sim-to-Real SysID Evaluation

We evaluate the task performance during sim-to-real SysID in real-world setups of object scooping and table air hockey. Our method has shown significant performance improvement on trajectory alignment and success rate compared to baseline methods. We evaluated all of our baselines in the sim-to-real transfer.

Object Scooping Sim-to-Real Evaluation. In this experiment, we verify that CAPTURE can accurately identify the center of mass across various objects during scooping. To verify the effectiveness of our algorithm, we selected three different objects (i.e.,celery, carrot, and eggplant) with asymmetrical properties to ensure the difficulty of identifying the center of mass. We evaluated each object ten times starting at the absolute center point. Similarly to the sim-to-sim transfer setting, we use task success rate to reflect the task performance instead of trajectory matching. To obtain the real-world object 3D pose, we utilize a point cloud to localize the object and provide the tilting direction labels.

Inspecting Table 3, we find that DR excels when scooping objects with centralized centers of mass, such as the eggplant, achieving a success rate of 90% or higher from just 1 iteration. However, for objects with more complex mass distributions (i.e.,celery and carrot), DR's performance drops significantly. CAPTURE is able to adapt to different objects and achieve at least 70% at 7th iterations. After successfully lifting the object, one-step adaptation methods randomly sample other parameter values due to the absence of history and lack of target-to-target parameter transition during training, while CAPTURE consistently lifts the object in subsequent iterations. This performance demonstrates CAPTURE 's ability to generalize to unseen scenarios (target-to-target adaptation) by leveraging context history. Its ability to maintain high success rates, especially with objects that have complex mass properties, underscores its effectiveness in real-world scooping tasks.

Air Hockey Sim-to-Real Evaluation. We set up the real-world table air hockey as shown in Figure 4. To create varying friction across the two surfaces, we installed separate fans under each side of the table, with adjustable fan voltages controlling the sliding friction. We evaluated the sim-to-real transfer performance over 15 trials using 3 different seeds, with each trial having randomized fan voltages on both sides. The results from the sim-to-real air hockey experiment, presented in Table 4, show the performance of different methods in trajectory matching over multiple adaptation iterations. For one-iteration adaptation, ED performs best with a trajectory difference of 0.40, as it tries to adapts to the

Scooping Objects	Method	Adaptation iterations				
		1 iter	3 iters	5 iters	7 iters	9 iters
Eggplant	ED	0.8	0.5	0.3	0.2	0.1
	DR	**0.9**	**0.9**	**1.0**	0.9	0.9
	TuneNet	0.9	0.9	0.7	0.6	0.4
	CAPTURE	0.3	0.6	0.9	**0.9**	**0.9**
Celery	ED	0.0	0.1	0.3	0.1	0.2
	DR	0.0	0.2	0.0	0.1	0.0
	TuneNet	0.0	0.1	0.3	0.1	0.1
	CAPTURE	0.0	**0.4**	**0.6**	**0.7**	**0.7**
Carrot	ED	0.0	0.8	0.5	0.5	0.6
	DR	0.0	0.1	0.0	0.1	0.0
	TuneNet	0.0	**0.9**	0.8	0.5	0.4
	CAPTURE	**0.3**	0.7	**0.9**	**0.9**	**0.9**

Table 3: Sim-to-Real Object Scooping Success Rate.

Method	Adaptation iterations				
	1 iteration	3 iterations	5 iterations	7 iterations	9 iterations
ED	**0.40 $\pm$ 0.19**	**0.34 $\pm$ 0.14**	0.51 $\pm$ 0.44	0.34 $\pm$ 0.15	0.34 $\pm$ 0.14
DR	0.41 $\pm$ 0.27	0.40 $\pm$ 0.1	0.37 $\pm$ 0.33	0.42 $\pm$ 0.42	0.43 $\pm$ 0.40
TuneNet	0.47 $\pm$ 0.22	0.40 $\pm$ 0.11	**0.32 $\pm$ 0.21**	0.38 $\pm$ 0.16	0.34 $\pm$ 0.15
CAPTURE	0.47 $\pm$ 0.18	0.35 $\pm$ 0.14	0.35 $\pm$ 0.12	**0.29 $\pm$ 0.10**	**0.27 $\pm$ 0.10**

Table 4: Sim-to-Real Table Air Hockey Trajectory Differences in Meters over 15 Runs.

target parameter within one iteration. However, as iterations increase, CAPTURE steadily improves, outperforming the baselines. By the 7th and 9th iterations, CAPTURE achieves the lowest trajectory differences of 0.29 and 0.27, respectively. In the final iterations, CAPTURE delivers about 20% better performance than the top baseline methods.

5 Conclusion

This paper introduces a novel in-context learning approach to bridge the sim-to-real gap in robotic tasks by adjusting environment parameters online. By leveraging interaction histories as context, we enable dynamics adaptation to real-world environments without requiring model updates. Evaluated in scooping and table air-hockey tasks, our method outperforms traditional approaches such as domain randomization and TuneNet, reducing the sim-to-real gap and improving both sim-to-sim and sim-to-real performance. The approach leverages historical multi-episode data to infer system parameters and provide a better real-world dynamics prediction. While our method demonstrates strong performance, it still requires to train a new SysID model for new task environments. Nonetheless, the framework offers a more efficient and accurate solution for real-world deployment of simulation-based robotic systems.

Acknowledgments

The authors want to acknowledge the support from the National Science Foundation under grants CNS-2047454. We also thank Haohong Lin and Changyi Lin for discussion.

References

Al-Rfou, R.; Choe, D.; Constant, N.; Guo, M.; and Jones, L. 2019. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 3159–3166.

Allevato, A.; Short, E. S.; Pryor, M.; and Thomaz, A. 2020. Tunenet: One-shot residual tuning for system identification and sim-to-real robot task transfer. In *Conference on Robot Learning*, 445–455. PMLR.

Chebotar, Y.; Handa, A.; Makoviychuk, V.; Macklin, M.; Issac, J.; Ratliff, N.; and Fox, D. 2019. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, 8973–8979. IEEE.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34: 15084–15097.

Chen, Z.; He, X.; Wang, Y.-J.; Liao, Q.; Ze, Y.; Li, Z.; Sastry, S. S.; Wu, J.; Sreenath, K.; Gupta, S.; et al. 2024. Learning Smooth Humanoid Locomotion through Lipschitz-Constrained Policies. *arXiv preprint arXiv:2410.11825*.

Chuck, C.; Qi, C.; Munje, M. J.; Li, S.; Rudolph, M.; Shi, C.; Agarwal, S.; Sikchi, H.; Peri, A.; Dayal, S.; et al. 2024. Robot Air Hockey: A Manipulation Testbed for Robot Learning with Reinforcement Learning. *arXiv preprint arXiv:2405.03113*.

Dai, T.; Wong, J.; Jiang, Y.; Wang, C.; Gokmen, C.; Zhang, R.; Wu, J.; and Fei-Fei, L. 2024. ACDC: Automated Creation of Digital Cousins for Robust Policy Learning. *arXiv preprint arXiv:2410.07408*.

Di Palo, N.; and Johns, E. 2024. Keypoint Action tokens enable in-context imitation learning in robotics. *arXiv preprint arXiv:2403.19578*.

Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Wu, Z.; Chang, B.; Sun, X.; Xu, J.; and Sui, Z. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Evans, B.; Thankaraj, A.; and Pinto, L. 2022. Context is everything: Implicit identification for dynamics adaptation. In *2022 International Conference on Robotics and Automation (ICRA)*, 2642–2648. IEEE.

Fu, L.; Huang, H.; Datta, G.; Chen, L. Y.; Panitch, W. C.-H.; Liu, F.; Li, H.; and Goldberg, K. 2024. In-Context Imitation Learning via Next-Token Prediction. *arXiv preprint arXiv:2408.15980*.

Grigsby, J.; Fan, L.; and Zhu, Y. 2024. AMAGO: Scalable In-Context Reinforcement Learning for Adaptive Agents. In *The Twelfth International Conference on Learning Representations*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. PMLR.

Hu, H.; Zhang, X.; Lyu, X.; and Chen, M. 2024. Learning Robust Policies via Interpretable Hamilton-Jacobi Reachability-Guided Disturbances. *arXiv preprint arXiv:2409.19746*.

Huang, P.; Xu, M.; Fang, F.; and Zhao, D. 2022. Robust Reinforcement Learning as a Stackelberg Game via Adaptively-Regularized Adversarial Training. In *the 31st International Joint Conference on Artificial Intelligence (IJCAI)*. Proceedings of the Thirty-First International Joint Conference on Artificial ....

Huang, P.; Zhang, X.; Cao, Z.; Liu, S.; Xu, M.; Ding, W.; Francis, J.; Chen, B.; and Zhao, D. 2023. What Went Wrong? Closing the Sim-to-Real Gap via Differentiable Causal Discovery. In Tan, J.; Toussaint, M.; and Darvish, K., eds., *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, 734–760. PMLR.

Jiang, C.; Ke, N. R.; and van Hasselt, H. 2023. Learning how to infer partial mdps for in-context adaptation and exploration. *arXiv preprint arXiv:2302.04250*.

Jiang, Y.; Wang, C.; Zhang, R.; Wu, J.; and Fei-Fei, L. 2024. TRANSIC: Sim-to-Real Policy Transfer by Learning from Online Correction. *arXiv preprint arXiv:2405.10315*.

Krishnamurthy, A.; Harris, K.; Foster, D. J.; Zhang, C.; and Slivkins, A. 2024. Can large language models explore in-context? *arXiv preprint arXiv:2403.15371*.

Kumar, A.; Fu, Z.; Pathak, D.; and Malik, J. 2021. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*.

Kumar, A.; Li, Z.; Zeng, J.; Pathak, D.; Sreenath, K.; and Malik, J. 2022. Adapting rapid motor adaptation for bipedal robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1161–1168. IEEE.

Laskin, M.; Wang, L.; Oh, J.; Parisotto, E.; Spencer, S.; Steigerwald, R.; Strouse, D.; Hansen, S.; Filos, A.; Brooks, E.; et al. 2022. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*.

Li, Y.; Li, J.; Fu, W.; and Wu, Y. 2024. Learning Agile Bipedal Motions on a Quadrupedal Robot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 9735–9742.

Lim, V.; Huang, H.; Chen, L. Y.; Wang, J.; Ichnowski, J.; Seita, D.; Laskey, M.; and Goldberg, K. 2022. Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting. In *2022 International Conference on Robotics and Automation (ICRA)*, 8282–8289. IEEE.

Lin, H.; Corcodel, R.; and Zhao, D. 2024. Generalize by Touching: Tactile Ensemble Skill Transfer for Robotic Furniture Assembly. *arXiv preprint arXiv:2404.17684*.

Martínez, C.; and Roura, S. 1998. Randomized binary search trees. *Journal of the ACM (JACM)*, 45(2): 288–323.

Mehta, B.; Diaz, M.; Golemo, F.; Pal, C. J.; and Paull, L. 2020. Active Domain Randomization. In Kaelbling, L. P.; Kragic, D.; and Sugiura, K., eds., *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, 1162–1176. PMLR.

Memmel, M.; Wagenmaker, A.; Zhu, C.; Fox, D.; and Gupta, A. 2024. ASID: Active Exploration for System Identification in Robotic Manipulation. In *The Twelfth International Conference on Learning Representations*.

Min, S.; Lyu, X.; Holtzman, A.; Artetxe, M.; Lewis, M.; Hajishirzi, H.; and Zettlemoyer, L. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Muratore, F.; Gruner, T.; Wiese, F.; Belousov, B.; Gienger, M.; and Peters, J. 2022. Neural posterior domain randomization. In *Conference on Robot Learning*, 1532–1542. PMLR.

Murooka, T.; Hamaya, M.; von Drigalski, F.; Tanaka, K.; and Ijiri, Y. 2021. Exi-net: Explicitly/implicitly conditioned network for multiple environment sim-to-real transfer. In *Conference on Robot Learning*, 1221–1230. PMLR.

Peng, X. B.; Andrychowicz, M.; Zaremba, W.; and Abbeel, P. 2018. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 3803–3810.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Ramos, F.; Possas, R. C.; and Fox, D. 2019. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*.

Ren, A. Z.; Dai, H.; Burchfiel, B.; and Majumdar, A. 2023. AdaptSim: Task-Driven Simulation Adaptation for Sim-to-Real Transfer. In Tan, J.; Toussaint, M.; and Darvish, K., eds., *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, 3434–3452. PMLR.

Shi, H.; Xu, H.; Clarke, S.; Li, Y.; and Wu, J. 2023. RoboCook: Long-Horizon Elasto-Plastic Object Manipulation with Diverse Tools. In *Conference on Robot Learning*, 642–660. PMLR.

Sikorski, K. 1982. Bisection is optimal. *Numerische Mathematik*, 40: 111–117.

Torne, M.; Simeonov, A.; Li, Z.; Chan, A.; Chen, T.; Gupta, A.; and Agrawal, P. 2024. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. *arXiv preprint arXiv:2403.03949*.

Wang, X.; Wang, W.; Cao, Y.; Shen, C.; and Huang, T. 2023a. Images speak in images: A generalist painter for in-context visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6830–6839.

Wang, Y.; Xu, M.; Shi, G.; and Zhao, D. 2023b. Guardians as you fall: Active mode transition for safe falling. *arXiv preprint arXiv:2310.04828*.

Xu, M.; Liu, Z.; Huang, P.; Ding, W.; Cen, Z.; Li, B.; and Zhao, D. 2022a. Trustworthy reinforcement learning against intrinsic vulnerabilities: Robustness, safety, and generalizability. *arXiv preprint arXiv:2209.08025*.

Xu, M.; Lu, Y.; Shen, Y.; Zhang, S.; Zhao, D.; and Gan, C. 2023. Hyper-decision transformer for efficient online policy adaptation. *arXiv preprint arXiv:2304.08487*.

Xu, M.; Shen, Y.; Zhang, S.; Lu, Y.; Zhao, D.; Tenenbaum, J.; and Gan, C. 2022b. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, 24631–24645. PMLR.

Yao, Y.; Liu, Z.; Cen, Z.; Zhu, J.; Yu, W.; Zhang, T.; and Zhao, D. 2024. Constraint-conditioned policy optimization for versatile safe reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Yu, C.; Lu, H.; Gao, J.; Tan, Q.; Yang, X.; Wang, Y.; Wu, Y.; and Vinitsky, E. 2024. Few-shot In-Context Preference Learning Using Large Language Models. arXiv:2410.17233.

Yu, W.; Tan, J.; Liu, C. K.; and Turk, G. 2017. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*.

Zhang, C.; Xiao, W.; He, T.; and Shi, G. 2024. WoCoCo: Learning Whole-Body Humanoid Control with Sequential Contacts. *arXiv preprint arXiv:2406.06005*.

Zhu, J. Y.; Cano, C. G.; Bermudez, D. V.; and Drozdzal, M. 2024. InCoRo: In-Context Learning for Robotics Control with Feedback Loops. *arXiv preprint arXiv:2402.05188*.

Zhu, Y.; Wong, J.; Mandlekar, A.; Martín-Martín, R.; Joshi, A.; Nasiriany, S.; and Zhu, Y. 2020. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*.

Appendix

5.1 Analysis on Different Transition Sequence Length L

We conducted ablation studies on the transition sequence length L to validate the chosen hyperparameters in our experiments. The sequence length L was varied from 5 to 13, as shown in Figure 7. We collected 100 pairs of simulation and simulated real environment parameters to comprehensively evaluate performance under different initial conditions. The results demonstrate minimal performance variation across different values of L, with some degradation observed at $L = 5$ and $L = 13$. Transition sequence lengths between 7 and 9 consistently yielded stable performance.

This range aligns well with other settings, as the proposed randomized binary search algorithm is independent of the dimensionality of state trajectories, action spaces, and environment parameter spaces.

A similar pattern was observed in the object scooping task, shown in Figure 11a. Parameter estimation performance followed similar trends, further validating the choice of transition sequence length.

5.2 Analysis on Different Transformer Window Sizes k

We evaluated the impact of varying the transformer window size k on performance, with the transition sequence length L fixed at 7. The window size k was varied from 2 to 6, with results shown in Figures 8 and 11b.

Shorter window sizes ($k = 2$, $k = 3$) generally outperformed larger ones, likely due to reduced overfitting and improved generalization. Larger window sizes exhibited diminished performance, especially in later iterations, due to overfitting to training data sequences and reduced generalization in out-of-distribution (OOD) settings. Smaller window sizes focus on local features, enhancing their OOD robustness.

5.3 Comparison with State-of-the-Art Offline SysID Baselines

We included COMPASS as a baseline for comparison. Figures 9 and 11c compare the performance of CAPTURE and COMPASS, highlighting CAPTURE's faster convergence and superior performance in most cases.

In the table air hockey setting, COMPASS struggled to perform well, despite favorable initialization for some environment parameters. For object scooping tasks, COMPASS aligned the center of mass but exhibited slower convergence compared to CAPTURE.

5.4 Limitations of Monotonicity Assumptions Between Environment Parameters and State Trajectories

The binary search method assumes monotonic relationships between parameters and trajectories. However, CAPTURE demonstrated strong performance even in non-monotonic environments. We modified the scoop environment to introduce non-monotonic relationships, and CAPTURE outperformed baselines in these settings, as shown in Figure 11d.

In non-monotonic settings, CAPTURE experienced slower convergence in the initial iterations but significantly reduced errors after the 10th iteration. CAPTURE consistently maintained superior performance compared to baselines in non-monotonic environments.

5.5 Additional Evaluation Under Noisy Observations

We further evaluated CAPTURE's robustness under noisy observations, as shown in Figures 10 and 11e, for both table air hockey and object scooping tasks.

In the object scooping task, random angular perturbations were introduced to simulate pose disturbances. Despite these challenges, CAPTURE maintained robust performance and outperformed baseline methods.

For table air hockey, uniform noise was added to state trajectory components, resulting in observed trajectory values ranging between 90% and 110% of ground-truth values. CAPTURE demonstrated resilience to noisy observations, outperforming baselines in later iterations.

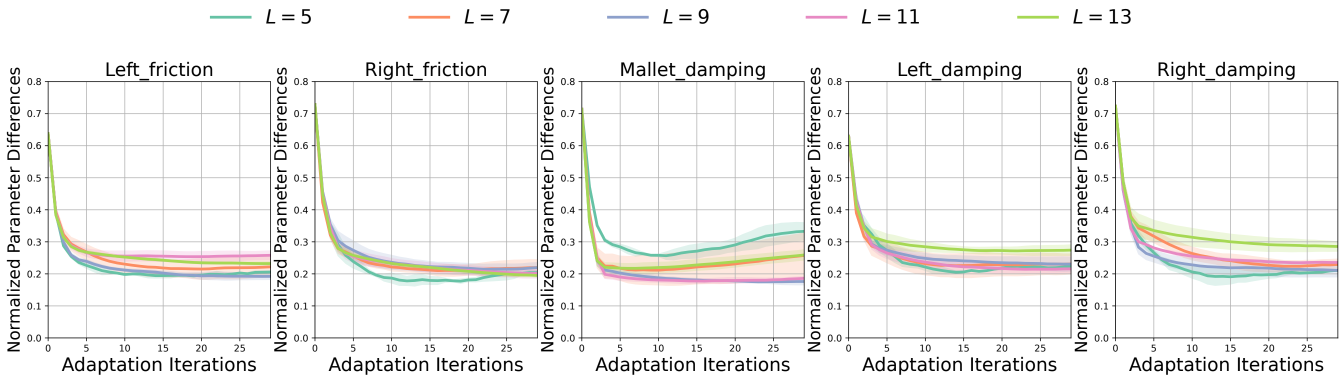


Figure 7: Table air hockey sim-to-sim transfer SysID performance across different parameter transition sequence lengths L.

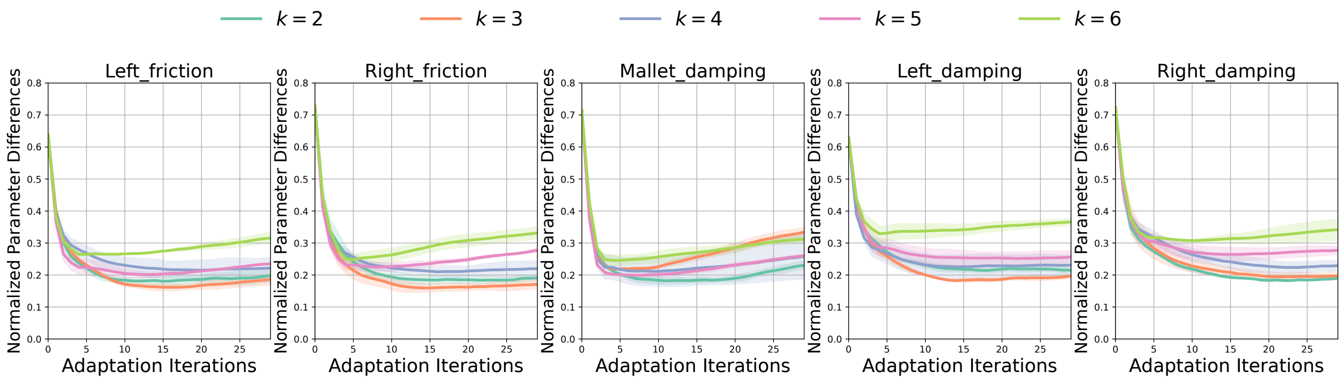


Figure 8: Table air hockey sim-to-sim transfer SysID performance across different window sizes.

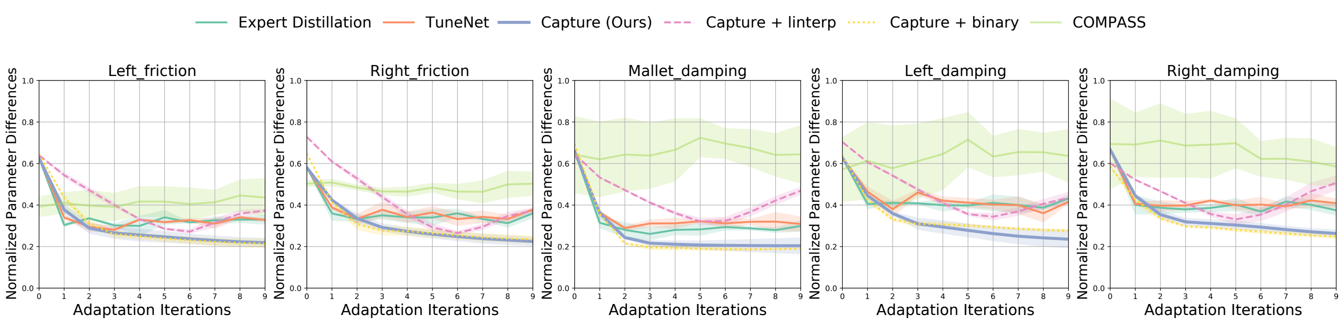


Figure 9: Table air hockey sim-to-sim transfer with added baseline.

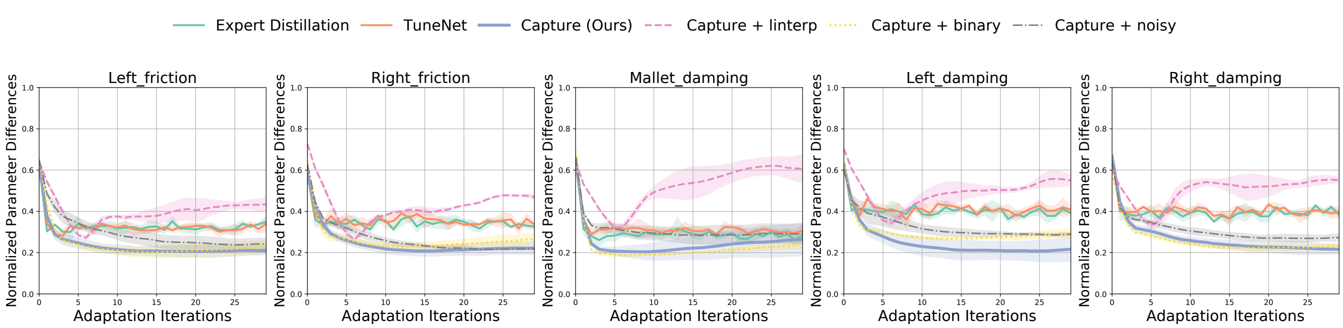
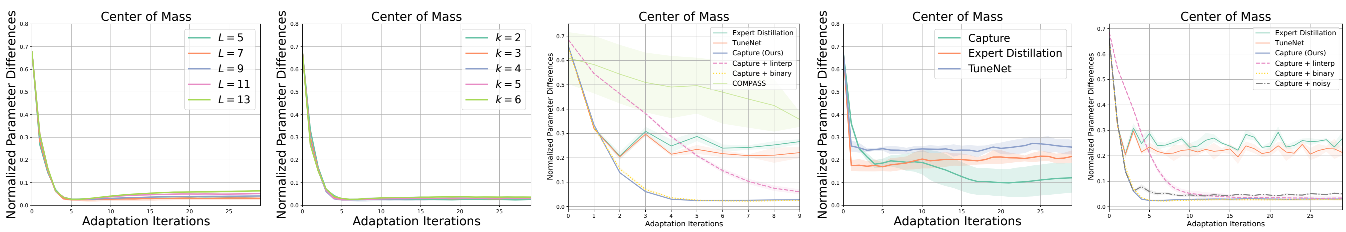


Figure 10: Table air hockey sim-to-sim transfer under noisy observations.



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

Figure 11: Object scooping sim-to-sim transfer SysID performance under different variants: (a) Performance across varying parameter transition sequence lengths L. (b) Performance across different window sizes k. (c) Performance with an added baseline. (d) Comparison of CAPTURE, Expert Distillation, and TuneNet in a non-monotonic scooping environment. (e) Performance under noisy observation conditions.