arXiv:2302.04903v2 [cs.RO] 1 Oct 2023

AdaptSim：面向仿真到现实迁移的任务驱动仿真适应

Allen Z. Ren[1], **Hongkai Dai**[2], **Benjamin Burchfiel**[2], **Anirudha Majumdar**[11]普林斯顿大学， [2]丰田研究院

摘要：接触模型和物体几何近似等仿真参数设置，对于训练能够从仿真迁移到现实世界部署的稳健操作策略至关重要。仿真与现实之间往往存在不可消除的差距：试图匹配仿真与现实之间的动力学可能不可行，也未必能让策略在现实环境中针对特定任务表现良好。我们提出 AdaptSim，一种面向仿真到现实迁移的全新任务驱动适应框架，旨在优化策略在目标（现实）环境中的任务性能。首先，我们在仿真中使用强化学习进行元学习，训练一种适应策略，根据策略在目标环境中的当前表现调整仿真参数分布。随后，我们通过推断新的仿真参数分布来进行策略训练，从而实现迭代式现实世界适应。大量仿真与硬件实验表明，与基于系统辨识（Sys-ID）的方法以及直接在目标环境中训练任务策略相比，AdaptSim 在适应不同环境时能够达到 1–3 倍的渐近性能和 ∼2 倍的真实数据效率。[1]

1 引言

学习能够适应现实世界操作任务且具有泛化能力的策略，通常需要大量训练数据。由于仅使用真实数据的成本可能非常高，甚至难以实现，我们往往转而主要在仿真中进行训练。这就引出了一个问题：我们应如何设定仿真参数，才能在最大限度提升现实世界任务性能的同时，尽可能减少所需的真实世界数据?

一种常见的方法是进行域随机化[1, 2, 3, 4]：使用范围广泛的不同仿真参数训练策略，希望策略能够应对现实世界中动力学或观测可能出现的变化。然而，训练出的策略可能取得良好的平均性能，但在特定真实环境中的表现却不佳。已有研究通过系统辨识（Sys-ID），根据真实世界数据估计出最符合机器人或环境动力学的参数点值或分布。该估计可以通过单次迭代 [5] 或多次迭代[6]来完成。这些自适应域随机化技术能够训练出适用于特定目标环境的策略。

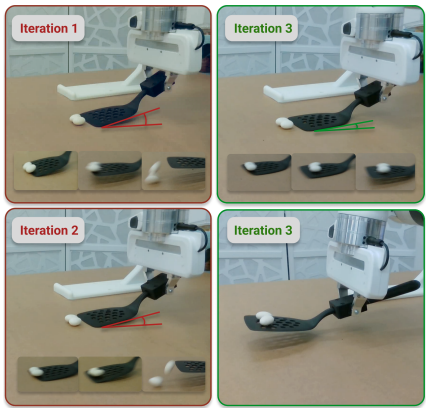


图 1：AdaptSim 在存在“不可消除”的仿真到现实差距的情况下，迭代提升动态舀取任务的性能。

虽然长方体等简单物体及其惯性等属性可以得到良好建模，但仍有大量“不可约的”仿真到现实差距存在于许多场景中，例如接触密集型操作任务。以使用烹饪锅铲从桌面上动态铲起小块食物这一任务为例（图1）。碎片和锅铲的精确几何形状难以指定；诸如锅铲抵住桌面时发生弯曲等形变，在仿真器中的实现尚不成熟；而点接触等接触模型已被证实无法很好地近似真实世界中复杂的接触行为 [7]在这些场景中。在这种

[1]网页：https://irom-lab.github.io/AdaptSim/

第七届机器人学习大会（CoRL 2023），亚特兰大，美国。

情况下，真实环境相对于仿真属于分布外（OOD），并且由于这种固有的不可约差距，对仿真参数进行系统辨识可能无法训练出适用于真实世界的有效策略。

贡献。 本文采用任务驱动 的方法：我们不试图使仿真与现实世界动力学相匹配，而是专注于寻找能够使所得策略优化任务性能的仿真参数。即使仿真到现实差距无法消除，这种方法也能使策略在现实世界中获得高奖励。我们考虑这样一种设置：机器人在每轮现实世界交互之间都可以访问模拟器，从而能够观察现实世界动力学，并相应地调整模拟器，以提升现实中的任务性能。我们提出了*AdaptSim* ——一种两阶段框架：(i) 首先在仿真中使用强化学习对用于更新仿真参数的适应策略进行元训练；(ii) 然后将其部署到真实环境中进行迭代。训练适应策略以最大化任务奖励，可通过只识别与任务相关的仿真参数来提高真实数据使用效率，并帮助训练好的策略更好地泛化到分布外（真实）环境。我们在三项机器人任务中展示了该方法的效果，其中两项涉及高接触操作。与基于系统辨识以及直接在目标环境中训练任务策略的方法相比，该方法在分布外环境中实现了 1–3 倍的渐近性能和 ∼2 倍的真实数据效率。

2 相关工作

机器人领域的仿真到现实迁移主要通过域随机化（DR）技术实现[1, 8, 9, 10, 11, 12, 13]，这些技术会在视觉、动力学和驱动相关的仿真参数中注入噪声。下面我们总结一些能更好地适应真实环境的技术。

系统辨识领域自适应。受系统辨识经典工作的启发 [14, 15]，，一种广受关注的研究方向是识别仿真参数，使其与真实环境中的机器人和环境动力学相匹配。BayesSim [6]及后续工作 [16, 17]应用贝叶斯推断，根据仿真和真实世界的轨迹，迭代搜索仿真参数的后验分布。然而，这些方法考虑的是建模相对完善的环境参数化，例如平面接触时的物体质量或摩擦系数；当仿真无法很好地近似真实世界时，系统辨识方法可能会十分脆弱 [13, 18]。

任务驱动的领域自适应。AdaptSim 更适合归入另一类研究，这类研究旨在寻找能使目标环境中的任务奖励最大化的仿真参数。Muratore 等人 [19] 将贝叶斯优化（BO）用于优化真实摆锤摆起任务中的摆杆质量和关节阻尼系数等参数。其他研究则只关注适应仿真领域 [20, 21, 22]。这类方法的一大缺点是需要在目标环境中进行大量滚动执行（例如 [19] 中的 150 次），对于许多需要人工重置的任务而言，这非常耗时。AdaptSim 在仿真中元学习适应策略，推断时仅需少量真实滚动执行（例如，在我们的推送实验中为 20 次）。

3 问题表述

环境。在仿真中，我们考虑一个空间 Ω，用于参数化摩擦系数和几何基元的维度等量。令 $\mathcal{E}$ 表示一个支撑集为 Ω 的仿真参数分布。用 $E \in$表示一个仿真环境，用 Ω 和 E^r表示一个真实环境。

任务策略与轨迹。 我们将任务策略 $\pi \in \Pi: \mathcal{O} \to \mathcal{A}$ 记为一个将机器人的观测 o_t 映射为动作 a_t的函数。在环境中运行该策略会产生一条状态-动作轨迹 $\tau(\pi; E): [0, T] \times \Pi \times \mathcal{E} \to \mathcal{S} \times \mathcal{A}$，其时间范围为 T。该轨迹还受初始状态分布影响。我们使用奖励函数来指定机器人的任务（例如，将某个物体推送到桌面的特定位置），并令$R(\tau) \in [0,1]$ 表示一条轨迹所获得的归一化累积奖励。我们将任务策略 π 在环境 E中运行时的奖励记为 $R(\pi; E)$，该奖励是对初始状态分布取期望所得。

目标。 我们的最终目标是在真实环境E_r中找到能够最大化任务性能的任务策略。我们不直接搜索策略，而是搜索用于训练 π 的最佳仿真参数分布 $\mathcal{E}$，以求解以下双层优化目标：

$$\sup_{\mathcal{E}} R(\pi^*_{\mathcal{E}}; E^r), \text{ where } \pi^*_{\mathcal{E}} := \sup_{\pi} \mathop{\mathbb{E}}_{E \sim \mathcal{E}} [R(\pi; E)], \tag{1}$$

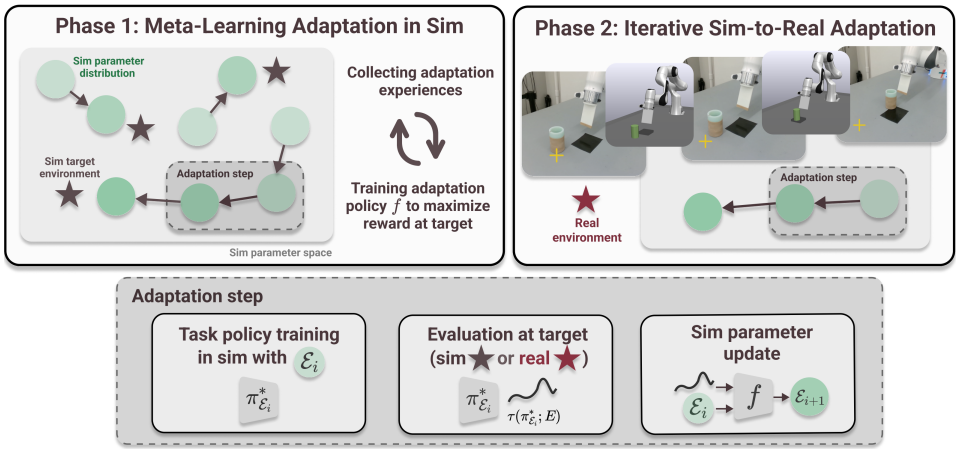


图 2：AdaptSim 包含两个阶段：（1）在仿真中进行元训练，针对随机采样的模拟目标环境最大化任务奖励，从而训练适应策略；（2）根据真实轨迹迭代调整仿真参数分布。右上方的示意图表明，仅使用少量真实轨迹，就能调整任务策略，使其将瓶子推得更靠近目标位置（黄色十字）。

适用于 $\mathcal{E}$ 的最优任务策略。执行（1）的外层优化需要与 E^r （真实世界）进行交互；我们允许少量此类交互预算。我们强调，上述目标旨在确定能够最大化任务性能的仿真参数最优分布，这不同于试图匹配仿真与现实之间动力学的目标。

4 方法

解决（1）的一种方法是对 $\mathcal{E}$ 进行黑箱优化，方法是评估 $R(\pi^*_{\mathcal{E}}; E^r)$ [19], ，这需要大量真实轨迹预算（结果见第 6节）。AdaptSim 则将代价高昂的外循环转移到仿真中：它针对许多仿真环境求解（1），学习从环境到解的映射，然后推断出 E_r的解。该方法分为两个阶段（图2）：

1) 在仿真中对适应策略进行元学习：在仿真中随机采样目标环境 $E^s \in \Omega$，然后使用强化学习训练“适应”策略 $f:(\mathcal{E},\ \tau)\mapsto\Delta_{\mathcal{E}}$，使其在 E^s 中最大化任务奖励；训练过程中通过迭代更新仿真参数分布（及相应的任务策略）。**2) 利用真实数据迭代调整仿真参数**：给定真实环境 E^r，利用训练好的 f 和少量真实轨迹，迭代推断出更好的仿真参数分布；随后在仿真中迭代微调任务策略，以便在更新后的参数分布下提升任务奖励。

4.1 第一阶段：在仿真中元学习适应策略

为了在测试时正确推断未见过的真实环境的仿真参数，我们首先训练适应策略，使其能够为多个模拟目标环境推断出更优的参数。此阶段完全在仿真中进行。形式上，我们将该问题建模为一个部分可观测的上下文老虎机 [23]。

定义 1 仿真适应上下文老虎机（*SA-CB*）由一个元组 $(\Omega,\mathcal{T},\mathcal{P},R)$指定：

- Ω 是上下文空间。每个上下文对应一个模拟目标环境 E^s；该上下文无法直接观测。
- $\mathcal{T}$是上下文部分观测的空间。每个观测对应于在给定上下文中运行任务策略所观察到的一条轨迹。
- $\mathcal{P}$ 是动作空间。一个动作对应于选择一个仿真参数分布 $\mathcal{E}$。
- R 是在特定情境下选择某个动作所获得的奖励（即，任务策略 $\pi^*_{\mathcal{E}}$ 使用 $\mathcal{E}$ 训练后部署到目标环境 E^s 时的奖励 $R(\pi^*_{\mathcal{E}};E^s)$ ）。

仅通过与目标环境进行一次交互，可能很难推断出最优的 $\mathcal{E}\in\mathcal{P}$ ——如果当前任务策略在目标环境中表现很差，这次交互可能无法提供多少信息。因此，我们迭代地对 $\mathcal{E}$进行增量变化，并将参数分布初始化为 $\mathcal{E}_{i=0}$。通过求解 SA-CB（使用我们将在下文详细介绍的技术），我们元学习得到一个适应

策略 $f(\mathcal{E},\ \tau)$，以最大化：

$$\mathop{\mathbb{E}}_{E^s\sim\mathcal{U}_\Omega}\ \mathop{\mathbb{E}}_{\mathcal{E}_0\sim\mathcal{U}_\mathcal{P}}\ \sum_{i=0}^{I}\gamma^i R(\pi^*_{\mathcal{E}_i};E^s), \tag{2}$$

$$\text{where } \mathcal{E}_{i+1}=\mathcal{E}_i+\Delta_{\mathcal{E}_i},\Delta_{\mathcal{E}_i}=f\left(\mathcal{E}_i,\tau(\pi^*_{\mathcal{E}_i};E^s)\right),$$

且 $\mathcal{U}_\Omega$ 和 $\mathcal{U}_\mathcal{P}$ 分别是在 Ω 和 $\mathcal{P}$ 上的均匀分布，$\gamma<1$ 是折扣因子。这是在随机采样模拟目标环境和初始仿真参数分布的情况下，从 $i=0$ 到适应时域 I进行多次交互所获得的任务奖励折扣和的期望值。

仿真参数分布空间。我们将 $\mathcal{P}$ 可能的仿真参数分布空间设为高斯分布，其均值限制在 Ω 内，方差固定。我们还使用固定的步长 δ 来调整每个仿真参数，其范围根据 Ω 的维度而定，为参数范围的 10% 到 15%；因此，沿每个维度的可能 $\Delta_\mathcal{E}$ 集合为 $\{\delta,-\delta,0\}$。

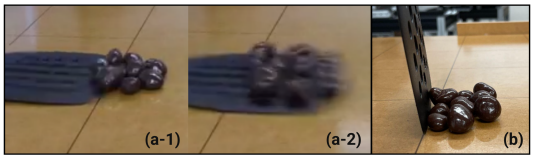


图 3：任务策略轨迹能够更好地揭示任务相关信息，例如快速接触下的舀取动力学。

算法 1 在仿真中进行元学习以适应策略

要求： $(\Omega\ \mathcal{T},\ \mathcal{P},\ R)$，SA-CB
需要： $S_f=\varnothing$，回放缓冲区
需要： $S_\mathcal{E}=\varnothing$，所有仿真参数的集合
分布（及其任务策略）
1: 初始化 $\epsilon\leftarrow 1$
2: **对于** $k\leftarrow 0$ 到 K **执行**
3: 采样目标 $E^s\sim\mathcal{U}_\Omega$ 和 $\mathcal{E}_{i=0}\sim\mathcal{U}_\mathcal{P}$
4: **为** $i\leftarrow 0$ 到 I **执行**
5: 训练任务策略 $\pi^*_{\mathcal{E}_i}$（第A2.2节）
6: 收集 $\tau(\pi^*_{\mathcal{E}_i};\ E^s)$ 并 $R(\pi^*_{\mathcal{E}_i};\ E^s)$
7: 采样随机的 $\Delta_{\mathcal{E}_i}$ 或推断 $\Delta_{\mathcal{E}_i}=f(\mathcal{E}_i,\ \tau(\cdot;\ \cdot))$
8: 更新 $\mathcal{E}_{i+1}\leftarrow\mathcal{E}_i+\Delta_{\mathcal{E}_i}$
9: 将 $(\mathcal{E}_i,\ \Delta_{\mathcal{E}_i},\ \tau(\cdot;\cdot)\ R(\cdot;\cdot))$添加至 S_f
10: 将 $\mathcal{E}_i$（以及 $\pi^*_{\mathcal{E}_i}$）添加至 $S_\mathcal{E}$
11: **结束循环**
12: 使用双重Q学习和 S_f训练 f
13: 将 ϵ 退火至0
14: **结束循环**
15: **返回** f，$S_\mathcal{E}$

将任务策略轨迹作为观测。我们选择任务策略 $\pi^*_\mathcal{E}$ 来生成适应策略所用的轨迹观测。我们的直觉是，与任意策略或那些在动力学方面生成最具“信息量”轨迹的策略相比[24]，$\pi^*_\mathcal{E}$ 此策略能更好地揭示目标环境中与任务相关的信息。在舀取任务中，机器人需要尝试舀起碎片，以便了解环境对任务的影响（例如，平底碎片通常更难舀起）。单纯推送碎片并不能展现碎片在快速接触下的行为（图3）。

使用强化学习训练适应策略。适应策略 f 采用分支式 Dueling Q 网络 [25], 进行参数化，该网络输出沿每个 $\{\delta,-\delta,0\}$ 动作维度选择任一动作的状态-动作价值。其输入包括：（1）当前仿真参数分布均值向量；（2）轨迹观测。我们应用强化学习（RL）来训练 f 以最大化公式（2）。在仿真中，我们收集 K “适应轨迹”；每条轨迹是一组$\{(\mathcal{E}_i,\Delta_{\mathcal{E}_i},R(\pi^*_{\mathcal{E}_i};E^s)\ \tau(\pi^*_{\mathcal{E}_i};E^s))\}_{i=0}^{I}$，并保存到回放缓冲区 S_f中。由于每一步都需要训练相应的任务策略 $\pi^*_{\mathcal{E}_i}$，这可能开销很大，因此我们采用离策略 Double Q-Learning [26] 来提高样本效率。使用这种方法，适应策略输出参数化 Q 函数的贪心动作，$f(\mathcal{E},\tau)=\text{argmax}_{\Delta_\mathcal{E}}Q(\mathcal{E},\tau;\Delta_\mathcal{E})$。我们采用 ϵ-贪心探索，初始值设为 1，并逐渐衰减至 0。

这构成了 AdaptSim 的第一阶段。算法1 详细说明了在内循环中收集适应轨迹（第 4–15 行）以及元学习适应策略的步骤。我们将所有分布（以及对应的任务策略，为方便起见在符号中省略）保存在集合 $S_\mathcal{E}$中，并在第二阶段再次使用。在 Sec. $\mathcal{E}$ 中，为每个 1训练任务策略是算法 A2.2 中计算开销最大的部分；我们将在 $\mathcal{E}$ 中说明所采用的启发式方法，以便重复使用任务策略，从而提高计算效率。

4.2 阶段 2：使用真实数据迭代调整仿真参数

在仿真中对适应策略进行元训练，使其能够为各种目标环境找到良好的任务策略后，我们可以在推断时应用该策略，并针对真实环境进行适应 E^r。算法2详细说明了这一迭代过程。我们采用与算法1内循环相同的适应流程，迭代 I^r次：在仿真中训练任务策略，在真实环境中评估该策略，并根据真实轨迹推断仿真参数的变化。我们始终应用来自 $f(\mathcal{E},\ \tau)$ $(\epsilon=0)$的贪心动作。

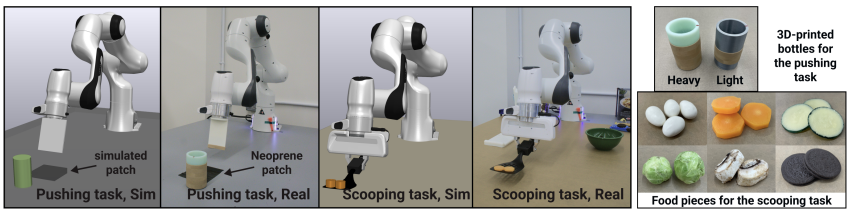


图 4：仿真与现实中的动态推推动作和动态舀取任务设置。

算法 2 利用真实数据迭代调整仿真参数

要求： E^r，真实环境**要求：** $(\Omega\ \mathcal{T},\ \mathcal{P},\ R)$，SA-CB**要求：** f，在第一阶段训练的适应策略**要求：** S_f，第一阶段得到的仿真参数分布集合（及相应的任务策略）1 S'_f S_f：2 $i \leftarrow 0$ I^r从中采样：**对于**至**执行**3：**对于** $\mathcal{E}_i \in S'_f$ **执行**4：在仿真中训练或微调任务策略 $\pi^*_{\mathcal{E}_i}$ 5：在真实环境中收集 $\tau(\pi^*_{\mathcal{E}_i}; E^r)$和 $R(\pi^*_{\mathcal{E}_i}; E^r)$ 6 $\mathcal{E}_{i+1} \leftarrow \mathcal{E}_i + f\big(\mathcal{E}_i\ \tau\ \cdots\big)$：更新，7 $(\ ;\ )$：**结束循环**8：**结束循环**9：**返回** $\pi^*_{\mathcal{E}_i}$，其 R $(\cdot; E^r)$最高

由于我们在第一阶段已采样了大量 $S_\mathcal{E}$ 参数分布，并针对这些分布训练了任务策略，因此可以在此处重复使用它们。在阶段 2 开始时，我们采样 $S'_\mathcal{E}$，即一组 N 保存在 $S_\mathcal{E}$中的分布，作为需要独立调整的初始分布。通常，我们选择 $N=2$，以权衡所需真实轨迹的数量与任务性能的收敛情况（分析见附录A5）。

5 任务

接下来，我们将详细介绍用于评估 AdaptSim 和基线方法的三项机器人任务。我们选择这些任务并设计相应环境，以突出训练域与测试域之间无法消除的差异。

5.1 线性化双摆的摆起

这是一个经典的控制任务，目标是将一个简单的双摆摆起。双摆的两根连杆一端各有一个驱动关节。我们考虑摆锤处于顶端时状态附近线性化后的动力学，因此可以使用线性二次型调节器（LQR）精确求解最优策略 [27]，针对一组特定的仿真参数（即狄拉克δ分布）。任务代价（奖励）函数采用标准的二次状态误差和驱动惩罚进行定义。轨迹观测是沿两个关节的轨迹均匀采样的点。

仿真设置。环境由四个参数确定：m_1 和 $m_2 \in [1,2]$，分别是两个关节的点质量；以及 b_1 和 $b_2 \in [1,2]$，分别是两个关节的阻尼系数。动力学通过数值积分进行仿真，未使用专用的物理仿真器。

5.2 瓶子的动态桌面顶推送

机器人需要将一个瓶子动态推送到桌面上的特定目标位置（图4）。由于目标可能位于机器人工作空间之外，机器人必须以较高速度推送物体，使物体在短暂接触后继续滑动。任务策略由一个神经网络参数化，该网络将期望目标位置映射为动作，包括：(1) 平面推送角度和 (2) 机器人末端执行器速度（可视化见附录A4），以及预测的奖励。随后，该网络充当状态值（Q）函数，并通过离策略方式进行训练，同时将仿真轨迹保存到回放缓冲区中。任务代价（奖励）定义为目标位置与瓶子最终位置之间的距离。轨迹观测可以是：(1) 仅瓶子的最终二维位置，或 (2) 沿二维轨迹均匀采样的点——我们在实验中考察了这两种表示。

仿真设置。我们使用 Drake 物理仿真器 [28]，因为它能够准确模拟接触力学。在这个仿真环境中，桌面的一小块区域会被赋予不同的物理属性，以模拟工作表面上的潮湿或黏性区域。该任务的参数设置如表 1所示。水弹性模量是水弹性接触模型的一个参数 [7]

符号	描述	范围
μ	桌面摩擦系数	$[0.05, 0.2]$
e	水弹性模量 [7]	$[1e4, 1e6]$
μ_p	补丁摩擦系数	$[0.20, 0.80]$
y_p	斑块横向位置	$[-0.10, 0.10]$

表 1：推动任务的仿真设置。

在 Drake 中实现——它大致模拟物体之间接触的“柔软”程度，数值越小表示接触越柔软。

真实设置。 使用两个尺寸相同但材料和质量不同的 3D打印瓶子（图 4，重型和轻型）。在理想化模型中，滑动距离应只取决于接触面，而与质量无关——仿真中的情况确实如此——但在真实实验中，我们发现两个瓶子始终移动不同的距离。此外，重型瓶子即使被直线推动，也往往会因底面略有不平而发生轻微旋转。这类未建模效应体现了不可约的仿真到现实差距。我们还在桌面上粘贴了一小块高摩擦力的氯丁橡胶，它会使瓶子减速，并进一步增加任务动力学的复杂性。

5.3 使用锅铲动态舀取食物块

The robot scoops up small food pieces with a spatula on the table (Fig. 4). 这是一项具有挑战性的任务，需要对舀取轨迹进行精细规划——我们发现，人类若不经过几次练习，便无法稳定地完成这项任务。任务策略由一个神经网络参数化，该网络将食物块的初始位置映射为舀取轨迹的参数：(1) 锅铲与碎片之间的初始距离，(2) 锅铲相对于桌面的初始俯仰角，以及 (3) 抬起锅铲的时间步（详情见附录 A4 ），并预测奖励。任务奖励定义为动作结束时锅铲上的食物块比例。轨迹观测是沿食物块二维轨迹均匀间隔的点。

仿真设置。 我们再次使用Drake仿真器。参数设置见表 2。

真实环境设置。使用了六种不同的食物块（图4）：(1) 巧克力葡萄干，(2)（仿制的、橡胶质地的）胡萝卜片，(3)（仿制的、硬质的）切片黄瓜，(4) 生抱子甘蓝，(5) 生蘑菇片，以及 (6) 奥利奥饼干。它们涵盖了从圆形、椭球形到近似圆柱形的不同形状，同时具有不同程度的形变和摩擦特性。

符号	描述	范围
μ	摩擦系数	[0.25,0.4]
e	水弹性模量	[1e4,5e5]
g	食物块几何形状	{椭球体，圆柱体}
h	食物块高度	[1.5厘米 2。5厘米]

表2：舀取任务的仿真设置。

6 实验

通过下面的大量实验，我们证明，与系统辨识及其他基线方法相比，AdaptSim 在适应现实环境和分布外仿真环境时，能够提升渐近任务性能，同时提高数据效率。对于基线方法，我们首先考虑直接优化任务策略的方法：(1) **均匀域随机化（UDR）：** 训练任务策略，使其在来自 $\mathcal{U}_\Omega$的环境中优化平均任务奖励；(2) **UDR+目标：** 使用真实数据对 UDR 的任务策略进行微调；(3) **LearnInTarget：** 仅使用目标环境中的数据直接训练任务策略，具体做法是拟合一个小型神经网络，将动作映射为最终奖励。随后，策略输出预测奖励最高的动作。只要有足够的真实数据，这种基线方法就应当能够充当任务性能的预言机或上界，但其数据效率可能较低。接下来，我们考虑两种像 AdaptSim 一样执行系统辨识并迭代训练任务策略的方法：(4) **SysID-Bayes [6, 29]:** 根据真实轨迹迭代推断仿真参数分布，以使仿真中的动力学与现实相匹配；这种方法称为 BayesSim；(5) **SysID-Point：** 推断仿真参数的点估计，而非参数分布（我们推测，在某些情况下，用分布随机化仿真参数可能会对任务策略训练产生负面影响）。

6.1 AdaptSim 通过适应实现更好的任务性能

仿真到仿真适应。我们针对所有基线方法进行了实验，考察它们适应不同分布内（WD）和分布外仿真环境的表现。分布内环境通过在每项任务的 Ω 范围内采样所有仿真参数生成，而分布外环境则通过在 Ω 范围之外采样部分参数生成（详情见附录A5 ）。桌面3 展示了三项任务在目标环境中的适应结果。尽管系统辨识基线在分布内环境中取得了较高奖励，AdaptSim 在几乎所有分布外环境中都优于系统辨识基线。

仿真到现实适应。 接下来，我们开展适应真实环境的实验。图 5 展示了推动任务和舀取任务中每次适应迭代所获得的平均奖励。总体而言，AdaptSim 与系统辨识基线之间的性能差距在现实环境中更大，而 AdaptSim 实现了更好的性能。例如，在舀取任务中，AdaptSim 能够为黄瓜片训练出性能良好的任务策略（成功率为 60%）；由于这些碎片非常薄，难以舀取（见图 8）。其他基线方法均无法舀起这些碎片。

方法	双摆摆起					推瓶子					舀取食物				
	分布内	OOD-1	OOD-2	OOD-3	OOD-4	分布内	OOD-1	OOD-2	OOD-3	OOD-4	分布内	OOD-1	OOD-2	OOD-3	OOD-4
AdaptSim	**0.98**	**0.96**	**0.95**	**0.95**	**0.98**	0.95	0.87	**0.73**	0.86	0.77	**1.00**	0.64	**1.00**	**1.00**	**0.55**
SysID-Bayes [6]	0.85	0.76	0.79	0.23	0.96	**0.98**	0.80	0.65	0.81	**0.79**	0.90	**0.66**	0.81	**1.00**	0.36
SysID-Point	0.95	0.60	0.73	0.39	0.76	0.94	0.84	0.68	0.85	0.78	0.94	0.63	0.90	**1.00**	0.42
UDR	-	-	-	-	-	0.68	0.65	0.61	0.67	0.58	0.65	0.22	0.43	0.55	0.12
UDR+目标	-	-	-	-	-	0.78	0.73	0.66	0.71	0.70	0.61	0.31	0.49	0.60	0.21
LearnInTarget	-	-	-	-	-	0.91	0.75	0.66	0.74	0.71	0.03	0.00	0.25	0.26	0.03

表3：**仿真到仿真适应**。在三项任务中，针对不同分布内和分布外模拟目标环境，在各适应时域内取得的最佳平均奖励。对于摆锤任务，数值以 UDR 所取得的奖励（下界）和在 Ω 内使用最佳可能参数所取得的奖励（上界，通过穷举采样估算）为基准进行归一化。对于推动任务，数值以 20cm 作为最大误差进行归一化；该误差对应前向方向上可能的目标位置范围。

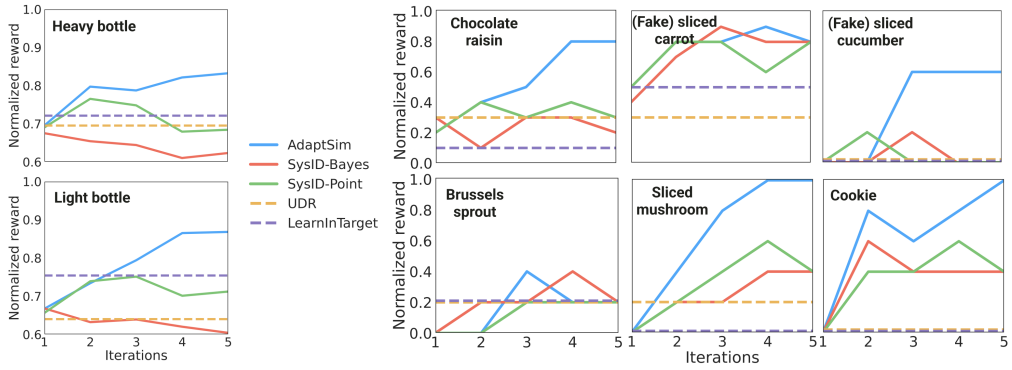


图5：**仿真到现实适应**。所有方法在推动任务（左）和舀取不同真实物体任务（右）中，经过适应迭代所取得的奖励（图片见图4）。结果为推动任务 10 次试验和舀取任务 5 次试验的平均值。

6.2 AdaptSim 提高真实数据效率

推动任务。我们比较了 AdaptSim 与 LearnInTarget 和 UDR+目标在不同真实数据预算下的表现。数据充足时，LearnInTar- get 和 UDR+目标应在目标环境中获得较高的奖励。这里不与系统辨识基线进行比较，因为第 6.1 节表明，它们通常无法在真实环境中达到相同水平的任务性能。在推动重型瓶子的任务中，桌面 4 表明 AdaptSim 仅使用 16 次试验就达到了相近水平的任务性能（~0.83），而 LearnInTarget 和 UDR+目标使用了 40 次试验。在 UDR+目标中，使用真实数据进行微调在真实数据预算足够之前并不奏效，并且可能会降低低数据量情形下的性能（例如，4 次和 8 次）。这也体现了利用仿真来摊销策略训练的数据需求。我们还在此引入了一个新的基线方法 **BayesOpt**，其基于 [19]，通过贝叶斯优化直接优化公式（1）。然而，使用 24 次滚动执行次数（初始化优化所需的最少次数）时，它只能达到 0.65。

方法	真实数据预算							
	0	4	8	16	24	32	40	48
AdaptSim	0.30	0.69	0.80	0.83	0.84	0.82	0.83	
LearnInTarget	0.05	0.04	0.63	0.69	0.76	0.80	0.84	0.83
UDR+目标	0.63	0.56	0.62	0.66	0.68	0.74	0.82	0.82
BayesOpt	-	-	-	-	0.65	0.72	0.79	0.80

表 4：**适应数据效率**。在使用不同数量真实数据的推动任务中，使用重型瓶子时获得的归一化奖励。

舀取任务中的提升幅度更大。虽然 LearnInTarget 和 UDR+Target 在推动任务中表现尚可，但 LearnInTarget 在舀取任务的所有食物块上都获得了较低的奖励，而 UDR+Target 的表现并未优于 UDR策略。舀取任务中的动作空间更加复杂，需要显著更多的数据来搜索或改进任务策略。AdaptSim 在仿真中的适应预训练大幅降低了对真实数据的需求。

6.3 AdaptSim 找到的仿真参数与系统辨识所得参数不同

我们预期 AdaptSim 能找到任务性能更佳的仿真设置，而不一定会最小化仿真与现实之间的全部动力学差异。图6 显示，SysID-Bayes 找到的参数在参数空间中更接近目标，但对于摆锤任务，这些参数带来的任务奖励低于 AdaptSim 找到的参数。此外，我们计算了动力学

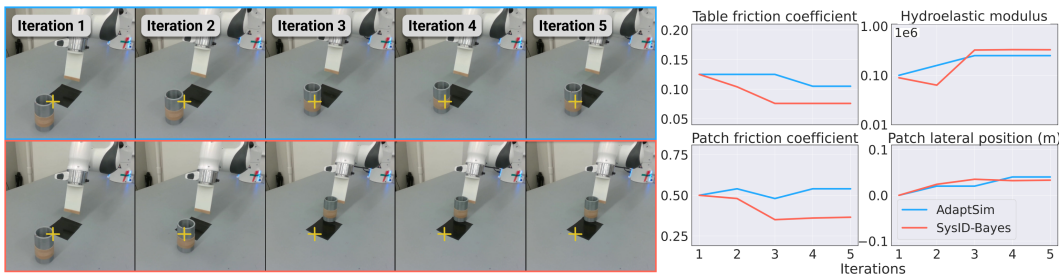


图7：**推送任务中的适应**。 AdaptSim正确学会了快速将瓶子推向目标附近。AdaptSim学得的任务相关仿真参数与SysID-Bayes学得的参数明显不同；后者往往低估桌面和垫片的摩擦，导致推瓶时用力不足，任务性能也更差。

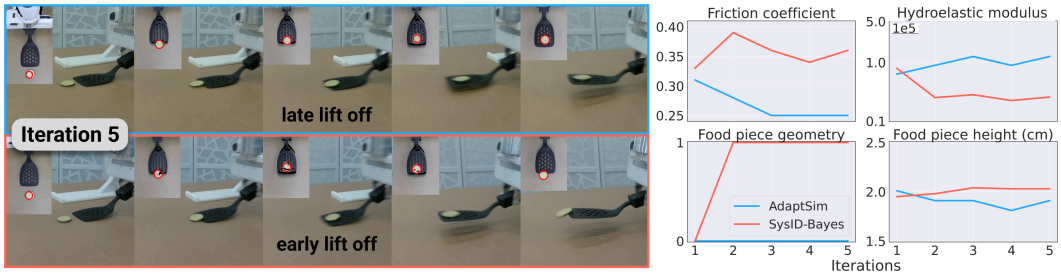


图8：**舀取任务中的适应**。 借助AdaptSim，黄瓜被成功舀起：锅铲较晚才从桌面抬起；否则，食物块就会从锅铲上滑落。AdaptSim推断食物块呈椭球形状（$g=1$，食物块几何形状），而SysID-Bayes推断其呈圆柱形状。

差异，计算方式为目标环境中的轨迹与采用适配参数的环境中的轨迹之间的总变差。AdaptSim 和 SysID-Bayes 在 OOD-1环境中的结果分别为 17.6 和 12.1，在 OOD-2 中为 21.7 和 11.1，在 OOD-3 中为 39.9 和 16.4，在 OOD-4 中为 75.8 和 56.4。因此，对于全部四个分布外目标环境，SysID-Bayes 找到的仿真参数所产生的动力学更接近目标环境（差异更小），但表3 显示，任务性能反而更差。

图 7 和图 8 进一步展示了SysID-Bayes表现不如AdaptSim的案例，同时也能明显看出两种方法得到的仿真参数分布存在差异。在推动任务中，SysID-Bayes推断的桌面和垫片摩擦系数过低，因此训练出的任务策略只能缓慢地推动瓶子。有趣的是，在舀取任务中，尽管切片黄瓜看起来像一个非常扁的圆柱体，AdaptSim仍推断其呈椭球形状，任务策略的成功率达到60%。系统辨识推断其呈圆柱形状，但任务策略完全失败。

7 讨论

总结。 我们发现，黑盒系统辨识通常并非最优，因为它试图拟合环境动力学中与任务无关的部分，却牺牲了下游性能。为了解决这一问题，我们提出了 *AdaptSim*，一种高效地将仿真训练任务策略适应到真实世界的框架。AdaptSim 通过元学习，掌握如何调整仿真参数分布，以便在多样化的模拟目标环境中获得更好的性能；随后，它利用少量真实数据，为训练真实世界中的任务策略推断出更优的分布。

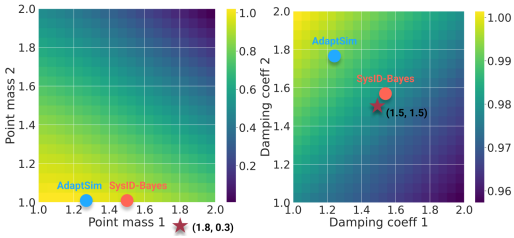


图 6：在双摆任务的 OOD-1 设置中，AdaptSim 与 SysID-Bayes 找到的仿真参数。颜色表示每个参数对应的最大可能奖励。SysID-Bayes 找到的参数在参数空间中更接近目标（深红色星形），但任务性能更差。

局限性与未来工作。 在某些情况下，AdaptSim 的表现不如基线方法（例如，推动任务中的 OOD-4，以及使用硬件舀起抱子甘蓝）。首先，AdaptSim 的任务驱动适应训练要求训练好的任务策略在相应的仿真参数分布上达到（近乎）最优——虽然双摆任务可以精确求解，但两个操作任务中的任务策略训练可能会受到噪声影响。其次，如果目标环境与仿真域存在极大的分布外差异，且适应策略未使用相似轨迹进行训练，AdaptSim 可能

无法取得良好效果。舀起抱子甘蓝就是一个例子（图 A8）。我们认为，可以通过增加用于任务策略训练的仿真预算，并改进任务策略的复用设计，来缓解第一个问题。第二个问题则可以通过设计仿真参数空间 Ω 来更好地涵盖可能出现的现实世界行为。我们还希望研究仿真的欠参数化和过参数化（Ω）对可达到的最佳任务性能的影响。最后，AdaptSim 未考虑多任务设置——一种可能的方向是先用多个任务预训练任务策略，再通过 AdaptSim 针对特定任务和环境对其进行微调。

致谢

作者部分得到了丰田研究院（TRI）、NSF CAREER奖[#2044149], 以及海军研究办公室（ONR） [N00014-23-1-2148、N00014-21-1-2803]的支持。本文仅反映作者及NSF、ONR、TRI或丰田其他实体的观点和结论。作者感谢TRI Robotics的动力学与仿真团队和灵巧操作团队提供技术支持与指导。我们还感谢林一民的宝贵讨论。

参考文献

[1] I. Akkaya、M. Andrychowicz、M. Chociej、M. Litwin、B. McGrew、A. Petron、A. Paino、M. Plappert、G. Powell、R. Ribas等。使用机器人手解决魔方问题。arXiv预印本 arXiv:1910.07113，2019。[2] A. Handa、A. Allshire、V. Makoviychuk、A. Petrenko、R. Singh、J. Liu、D. Makoviichuk、K. Van Wyk、A. Zhurkevich、B. Sundaralingam等。DeXtreme：将灵巧手内操作从仿真迁移到现实。arXiv预印本 arXiv:2210.13702，2022。[3] J. Lee、J. Hwangbo、L. Wellhausen、V. Koltun和M. Hutter。在复杂地形上学习四足运动。《Science Robotics》，5(47):eabc5986，2020。[4] A. Loquercio、E. Kaufmann、R. Ranftl、A. Dosovitskiy、V. Koltun和D. Scaramuzza。深度无人机竞速：通过域随机化从仿真走向现实。《IEEE机器人学汇刊》，36(1):1–14，2019。[5] V. Lim、H. Huang、L. Y. Chen、J. Wang、J. Ichnowski、D. Seita、M. Laskey和K. Goldberg。利用真实到仿真再到真实的自监督学习进行平面机器人抛掷。载于《IEEE国际机器人与自动化会议（ICRA）论文集》，2022。[6] F. Ramos、R. C. Possas和D. Fox。BayesSim：通过概率推断为机器人仿真器进行自适应域随机化。载于《机器人学：科学与系统》论文集，2019。[7] J. Masterjohn、D. Guoy、J. Shepherd和A. Castro。压力场接触斑块的速度级近似。《IEEE机器人与自动化快报》，7(4):11593–11600，2022。[8] K.-C. Hsu、A. Z. Ren、D. P. Nguyen、A. Majumdar和J. F. Fisac。从仿真到实验室再到现实：具有屏蔽机制和泛化保证的安全强化学习。《人工智能》，314:103811，2023。[9] J. Tobin、R. Fong、A. Ray、J. Schneider、W. Zaremba和P. Abbeel。用于将深度神经网络从仿真迁移到真实世界的域随机化。载于《IEEE/RSJ智能机器人与系统国际会议（IROS）论文集》，2017。

[10] F. Sadeghi 和 S. Levine。Cad2rl：无需任何真实图像的真实单图像飞行。载于机器人学：科学与系统论文集，2017 年。

[11] X. B. Peng、M. Andrychowicz、W. Zaremba 和 P. Abbeel。通过动力学随机化实现机器人控制的仿真到现实迁移。载于 *IEEE*国际机器人与自动化会议（*ICRA*）论文集，2018。

[12] G. B. Margolis 和 P. Agrawal。Walk These Ways：通过行为多样性调节机器人控制以实现泛化。载于机器人学习会议（CoRL）论文集，2022。[13] F. Muratore、F. Ramos、G. Turk、W. Yu、M. Gienger 和 J. Peters。来自随机化仿真的机器人学习：综述。Frontiers in Robotics and AI，9，2022。[14] M. Gautier 和 W. Khalil。关于机器人惯性参数的辨识。载于 IEEE决策与控制会议（CDC）论文集，1988。[15] P. K. Khosla 和 T. Kanade。机器人动力学的参数辨识。载于 IEEE决策与控制会议（CDC）论文集，1985。[16] E. Heiden、C. E. Denniston、D. Millard、F. Ramos 和 G. S. Sukhatme。通过并行可微仿真进行仿真参数的概率推断。载于 IEEE国际机器人与自动化会议（ICRA）论文集，2022。[17] R. Antonova、J. Yang、P. Sundaresan、D. Fox、F. Ramos 和 J. Bohg。用于可变形物体操作的真实到仿真的贝叶斯方法。IEEE机器人与自动化快报，7(3)：5819–5826，2022。[18] C. Chi、B. Burchfiel、E. Cousineau、S. Feng 和 S. Song。迭代残差策略：面向目标条件的可变形物体动态操作。载于机器人学：科学与系统论文集，2022。[19] F. Muratore、C. Eilers、M. Gienger 和 J. Peters。利用贝叶斯优化实现数据高效域随机化。IEEE机器人与自动化快报，6(2)：911–918，2021。[20] Q. Vuong、S. Vikram、H. Su、S. Gao 和 H. I. Christensen。如何为强化学习策略的仿真到现实迁移选择域随机化参数？arXiv预印本 arXiv:1903.11774，2019。[21] W. Yu、C. K. Liu 和 G. Turk。通过策略优化实现策略迁移。载于国际学习表征会议（ICLR）论文集，2018。[22] N. Ruiz、S. Schulter 和 M. Chandraker。学习仿真。载于国际学习表征会议（ICLR）论文集，2019。[23] A. Bensoussan。部分可观测系统的随机控制。剑桥大学出版社，1992。[24] J. Swevers、C. Ganseman、D. B. Tukel、J. De Schutter 和 H. Van Brussel。最优机器人激励与辨识。IEEE机器人与自动化汇刊，13(5)：730–740，1997。[25] A. Tavakoli、F. Pardo 和 P. Kormushev。深度强化学习的动作分支架构。载于 AAAI人工智能会议论文集，2018。[26] H. Van Hasselt、A. Guez 和 D. Silver。采用双重Q学习的深度强化学习。载于 AAAI人工智能会议论文集，2016。[27] R. Tedrake。欠驱动机器人学。2023。网址 https://underactuated.csail.mit.edu。[28] R. Tedrake 和 Drake开发团队。Drake：面向机器人的基于模型的设计与验证，2019。网址 https://drake.mit.edu。[29] R. Antonova、F. Ramos、R. Possas 和 D. Fox。BayesSimIG：利用 Isaac Gym 实现面向自适应域随机化的可扩展参数推断。arXiv预印本 arXiv:2107.04527，2021。[30] Y. Chebotar 、 A. Handa 、 V. Makoviychuk 、 M. Macklin、J. Issac、N. Ratliff 和 D. Fox。闭合仿真到现实闭环：利用真实世界经验调整仿真随机化。载于 IEEE国际机器人与自动化会议（ICRA）论文集，2019。

[31] A. Allevato、E. S. Short、M. Pryor 和 A. Thomaz。《Tunenet：用于系统辨识和机器人任务从仿真到现实迁移的单次残差调优》。载于《机器人学习会议（CoRL）论文集》，2020 年。[32] A. Ajay、M. Bauza、J. Wu、N. Fazeli、J. B. Tenenbaum、A. Rodriguez 和 L. P. Kaelbling。《结合物理模拟器与基于物体的网络进行控制》。载于《IEEE国际机器人与自动化会议（ICRA）论文集》，2019 年。[33] A. Zeng、S. Song、J. Lee、A. Rodriguez 和 T. Funkhouser。《Tossingbot：利用残差物理学习投掷任意物体》。发表于《IEEE机器人学汇刊》，36(4):1307–1319，2020 年。[34] A. Kumar、Z. Fu、D. Pathak 和 J. Malik。《RMA：足式机器人的快速运动适应》。载于《机器人学：科学与系统》论文集，2021 年。[35] W. Yu、J. Tan、C. K. Liu 和 G. Turk。《为未知做好准备：通过在线系统辨识学习通用策略》。载于《机器人学：科学与系统》论文集，2017 年。[36] B. Evans、A. Thankaraj 和 L. Pinto。《情境决定一切：用于动力学适应的隐式辨识》。载于《IEEE国际机器人与自动化会议（ICRA）论文集》，2022 年。[37] J. Liang、S. Saxena 和 O. Kroemer。《学习主动任务导向探索策略以弥合仿真到现实差距》。载于《机器人学：科学与系统》论文集，2020 年。[38] W. Jin 和 M. Posa。《用于灵巧操作的任务驱动的混合模型降阶》。arXiv 预印本 arXiv:2211.16657，2022 年。[39] A. Z. Ren 和 A. Majumdar。《通过对抗性环境生成进行分布鲁棒策略学习》。发表于《IEEE机器人与自动化快报》，7(2):1379–1386，2022 年。[40] H. S. Gomes、B. L´eger 和 C. Gagn´e。《元学习黑盒群体优化器》。载于《国际学习表征会议（ICLR）论文集》，2023 年。[41] M. Wistuba、N. Schilling 和 L. Schmidt-Thieme。《用于超参数优化的可扩展基于高斯过程的迁移代理模型》。发表于《机器学习》，107(1):43–78，2018 年。[42] Y. Chen、M. W. Hoffman、S. G. Colmenarejo、M. Denil、T. P. Lillicrap、M. Botvinick 和 N. Freitas。《通过梯度下降学习如何学习，无需梯度下降》。载于《国际机器学习会议（ICML）论文集》，2017 年。[43] M. Volpp、L. P. Fr¨ohlich、K. Fischer、A. Doerr、S. Falkner、F. Hutter 和 C. Daniel。《贝叶斯优化中用于迁移学习的元学习采集函数》。载于《国际学习表征会议（ICLR）论文集》，2020 年。[44] A. Z. Ren、B. Govil、T.-Y. Yang、K. Narasimhan 和 A. Majumdar。《利用语言加速工具操作学习》。载于《机器人学习会议（CoRL）论文集》，2022 年。[45] L. Johannsmeier、M. Gerchow 和 S. Haddadin。《机器人操作框架：技能形式化、元学习与自适应控制》。载于《IEEE国际机器人与自动化会议（ICRA）论文集》，2019 年。[46] A. Nagabandi、G. Kahn、R. S. Fearing 和 S. Levine。《用于基于模型的深度强化学习并结合无模型微调的神经网络动力学》。载于《IEEE国际机器人与自动化会议（ICRA）论文集》，2018 年。[47] J. Fu、K. Luo 和 S. Levine。《通过对抗性逆强化学习学习鲁棒奖励》。载于《国际学习表征会议（ICLR）论文集》，2018 年。[48] S. Gu、T. Lillicrap、I. Sutskever 和 S. Levine。《结合基于模型加速的连续深度Q学习》。载于《国际机器学习会议（ICML）论文集》，2016 年。

附录

A1 扩展相关工作

系统辨识领域自适应。受系统辨识领域经典研究的启发， [14, 15], 一种广受关注的研究路线是在任务策略训练前，辨识能够匹配真实环境中机器人及环境动力学的仿真参数。BayesSim [6] 及其后续工作 [16, 17] 运用贝叶斯推断，根据仿真和真实世界中的轨迹，迭代搜索仿真参数的后验分布。也有研究将这一推断问题表述为强化学习，以最小化轨迹差异 [30]。另一种方法 [31, 32, 33] 则学习一个动力学残差模型（通常以神经网络为参数化形式），使仿真或理想物理模型与现实相匹配。然而，所有这些方法都假设环境参数化模型相对准确，例如平面接触时的物体质量或摩擦系数；研究表明，当仿真无法准确逼近真实世界时，系统辨识方法会失效 [13, 18]。还有一些工作不试图推断完整的动力学，而是在线适应一个低维潜在表示 [34, 35, 36], , 但仍通过回归训练该表示，以匹配动力学或仿真参数。值得注意的是，上述系统辨识方法都与任务无关；当训练好的任务策略对仿真与现实之间的动力学差异较为敏感时，这可能导致性能不佳。Chi 等人[18] 通过利用仿真，根据动作变化预测轨迹变化，并将其作为隐式策略，来解决这一问题；但这种方法要求环境能够重置，而 AdaptSim 可处理随机初始化的物体状态。

任务驱动的领域自适应。 AdaptSim 更适合归入另一类研究：这类研究旨在寻找能使目标环境中的任务奖励最大化的仿真参数。Muratore 等人 [19] 将贝叶斯优化（BO）应用于真实摆锤摆起任务，优化摆杆质量和关节阻尼系数等参数。其他研究则只关注适应仿真领域[20, 21, 22]。这类方法的一个主要缺点是，需要在目标环境中进行大量滚动执行（例如， [19] 中为 700 次），对于许多需要人工重置的任务而言，这非常耗时。AdaptSim 在仿真中元学习适应策略，推断时只需少量真实滚动执行（例如，我们的推送实验中为 20 次）。Liang 等人 [37] 将相同的任务驱动目标应用于操作任务中的探索策略学习，但任务策略是通过系统辨识（Sys-ID）估计仿真参数后合成的。Jin 等人 [38] 将任务驱动的降阶模型应用于灵巧操作任务，但同样通过系统辨识来确定模型，且不涉及基于视觉的控制。Ren 等人[39] 根据当前任务性能搜索对抗环境（例如，物体），以增强策略的鲁棒性；但与 AdaptSim 不同，其对抗性指标仅在仿真领域中测量，不使用真实数据。

学习搜索/优化。我们的工作通过在相关问题的分布上进行元学习，学习优化策略，从而能够针对具体设置进行定制，并提高样本效率 [40, 41]。Chen 等人 [42] 通过元学习为黑盒优化学习一个 RNN 优化器。Volpp 等人 [43] 结合强化学习，在贝叶斯优化中对采集函数进行元学习；该方法能够为黑盒优化以及仿真到现实迁移中的控制器增益调节学习新的探索策略。元强化学习直接训练任务策略，以优化其在新环境中的性能 [44, 45, 46]——而 AdaptSim 则应用元强化学习来优化仿真参数。

A2 方法补充细节

A2.1 稀疏适应奖励

实际上，我们只关注奖励是否达到某个最低阈值——较差的任务策略并无实际用处。因此，我们使用公式（2）的稀疏奖励版本，

$$\mathop{\mathbb{E}}_{E^s\sim\mathcal{U}_\Omega}\mathop{\mathbb{E}}_{\mathcal{E}_0\sim\mathcal{U}_\mathcal{P}}\sum_{i=0}^{I}\gamma^i\mathbb{1}\big(R(\pi^*_{\mathcal{E}_i};E^s)\geq\overline{R}\big)R(\pi^*_{\mathcal{E}_i};E^s), \tag{A1}$$

其中 1() 是指示函数， R 是稀疏奖励阈值。使用稀疏奖励还有助于避免适应策略目光短浅并陷入次优解，

尤其是因为我们使用相对较小的 I（例如 5-10）来尽量减少真实数据的用量，并使用较小的折扣因子 γ（$=0.9$）。

A2.2 跨参数分布复用任务策略

算法1 要求针对每个 $\mathcal{E}$训练任务策略，而对于两个操作任务来说，这可能代价高昂。我们的直觉是，可以在距离较近的参数分布之间共享任务策略，具体启发式规则如下：

• 记录总预算（即轨迹数量），以及 j任务策略已针对其完成训练的仿真参数分布数量。

• 定义两个参数分布 $D(\cdot,\ \cdot)$之间的距离，例如均值之间的 L2距离。如果 $\mathcal{E}_i$与之前见过的某个分布之间的距离小于阈值 D，则复用该任务策略。如果该策略已接受总计 $M_{\text{最大}}$预算的训练，则不再训练；否则，使用 $\max(M_{\text{最小}}\ \alpha^{j-1}M)$ 预算进行训练，其中 $\alpha<1$ 和 M 是首次训练该策略的预算。

• 如果附近的参数分布复用了某个任务策略，就不要再次复用同一个策略。这样可以避免同一个任务策略被用于过多 $\mathcal{E}$。

备注 1 在不同参数分布之间重复使用任务策略，会使奖励 R 取决于适应历史，因为 $\pi^*_{\mathcal{E}}$ 取决于用于训练的先前 $\mathcal{E}$。我们选择不在 f中对这种历史依赖性建模，因为奖励应主要由当前的 $\mathcal{E}$决定。

A3 适应策略的更多细节

超参数。 桌面 A1 展示了第一阶段适应策略训练所用的超参数，包括用于定义在不同仿真参数分布之间重复使用任务策略的启发式规则的超参数。对于维度较小的 δ，我们通常采用更小的适应步长 Ω。

参数	Task		
	摆锤	推送	舀取
总适应步数，K	1e4	1e4	1e4
适应时域，I	10	8	8
适应步长，δ	0.10	0.15	0.15
适应折扣因子，γ	0.9	0.9	0.9
稀疏奖励阈值，R	0.95	0.8	0.5
任务策略复用阈值，D	-	0.16	0.16
任务策略最大预算，$M_{\max}$	-	3e4	4e3
任务策略预算折扣，α	-	0.9	0.9
任务策略初始预算，M	-	1e4	1.2e3

表 A1：三个任务的适应策略训练所用的超参数。

轨迹观测。我们详细说明了三个任务中使用的轨迹观测（作为适应策略的输入）。

• 摆锤任务：每次试验持续 2.5 秒，我们沿两个关节的轨迹均匀选取 12 个点，因此每条轨迹的维度为 24。对于 AdaptSim-State 、 SysID-Bayes-State 和 SysID-Bayes-Point，同样使用 12 个点，但仅从最后 0.5 秒内采样。每次适应迭代使用一条轨迹——输入适应策略的轨迹维度为 24。

• 推送任务：每次试验时长为 1.3 秒，我们沿瓶子的 X-Y 轨迹均匀选取 6 个点，因此每条轨迹也是 12 维。对于 AdaptSim-State、SysID-Bayes-State 和 SysID-Bayes-Point，仅使用瓶子的最终 X-Y 位置。每次适应迭代使用两条轨迹——输入适应策略的轨迹为 24 维。

• 舀取任务：每次试验时长为 1 秒，我们使用食物块在时间步[0,0.2,0.3,0.4,0.5,0.6,0.8,1.0]s 时的 X-Y 位置（在锅铲与食物块初次接触附近进行更密集的采样），因此每条轨迹为 16 维。每次适应迭代使用两条轨迹——输入适应策略的轨迹为 32 维。

在真实实验中，我们使用 Azure Kinect RGB-D相机的三维点云信息跟踪推动任务中的瓶子位置，发现这种方法十分准确。在舀取任务中，食物块太小、太薄，无法通过点云可靠地跟踪，因此我们改为从 RGB图像中提取轮廓，再在深度图像中查找相同像素处对应的深度值。在快速接触过程中，食物块周围可能会出现运动模糊，因此我们在仿真中的真实轨迹点上添加高斯噪声：X 位置的均值为 0.2 cm，Y 位置的均值为零，二者的协方差均为 0.2 cm。我们在 X 方向使用正均值，因为运动模糊往往出现在前向方向。

A4 任务设置和任务策略的补充细节

轨迹观测 首先，我们从任务策略轨迹中移除动作序列，仅保留状态序列。由于真实环境中的动力学可能处于分布外状态，为了达到与仿真环境中类似的高奖励状态，机器人需要使用训练期间未见过的动作（或针对特定状态未见过的动作）；如果任务策略轨迹中包含动作序列，就会妨碍适应策略的泛化能力。我们假设，只要任务策略在真实环境中表现良好，任务相关的 状态 序列就会得到覆盖$\mathcal{T}$。这一选择也见于处理训练测试动力学不匹配的仅状态逆强化学习文献 [47]。参见图A4 以及第 6.3 节的相关讨论。

A4.1 瓶子的动态推推动作

轨迹参数化。这里详细介绍为该任务设计的末端执行器推动器轨迹（图A1）。该轨迹由两个参数进行参数化：（1）平面推送角度，即推动器相对于前向方向的偏航方向，它控制瓶子被推送的方向；（2）沿推送角度所指方向的前进速度（末端执行器的速度）。推送角度在 -0.3弧度和0.3弧度之间变化，前进速度在0.4米/秒和0.8米/秒之间变化。我们发现，0.8米/秒大致是所用 Franka Panda机械臂的速度上限。推动器在运动过程中也会向上俯仰，速度固定为0.8弧度/秒。我们设计此类轨迹，以在硬件上限下最大限度地增加推送距离。

初始状态和目标状态。试验开始前，将瓶子放置在桌面上的固定位置（x=0.56米 y=0，相对于机械臂基座）。目标位置从一个区域中采样，其中 X 位置介于 0.7 至 1.0 米之间，Y 位置与中心线的偏差不超过 10 度（图A1 右上）。垫片是一个边长为10厘米的10厘米正方形，放置在 $x=0.75$米处，其中心位置（横向位置作为一个仿真参数进行变化）。

任务策略参数化。 任务策略采用归一化优势函数（NAF）进行参数化 [48] 通过将 Q 值限制为动作的二次函数，该方法能够以连续动作输出进行高效的 Q 学习，因此无需采样即可精确求出使 Q 值最大的动作。在此任务中，它将瓶子的期望二维目标位置映射为两个动作参数：平面推送角度和前进速度。该策略为开环策略——动作在试验开始前确定，且不会利用相机观测提供反馈。

硬件设置。在仿真与现实中，我们都将一个3D打印的板状推杆安装在末端执行器上，以替代平行夹爪。我们还在推杆底部和瓶子的接触区域缠绕弹性橡皮筋，以产生更强的弹性碰撞；我们发现，这会增加瓶子的滑动距离。

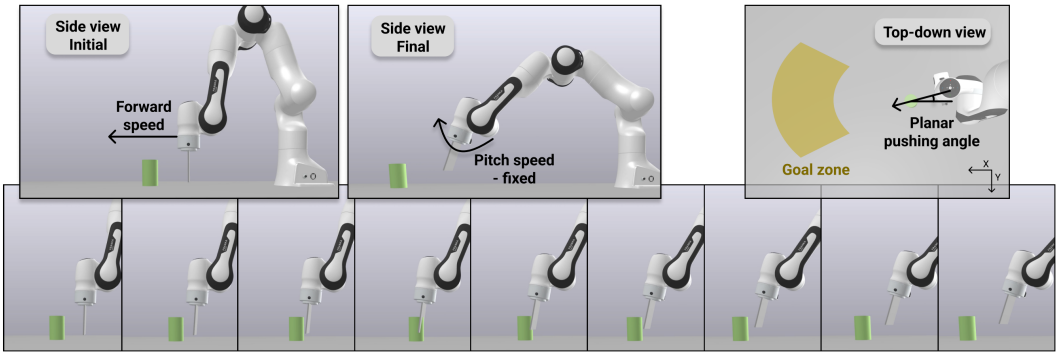


图 A1：Drake 仿真器中推动轨迹和目标位置的可视化。共有两个动作参数：（1）末端执行器的前进速度；（2）平面推送角度（即 末端执行器的偏航方向）。图中未显示垫片。

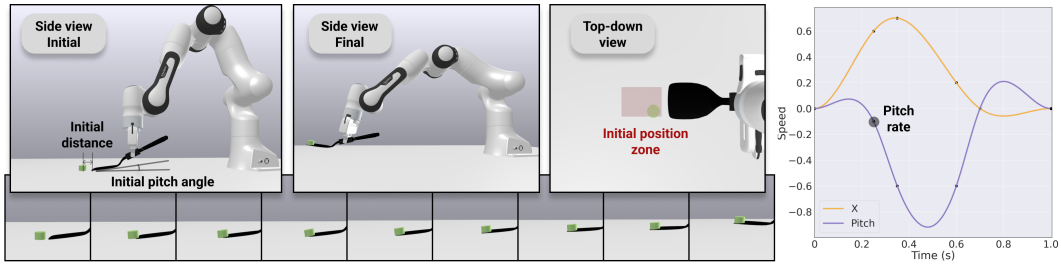


图 A2：Drake 仿真器中舀取轨迹和食物块初始位置的可视化。共有三个动作参数：（1）初始距离（锅铲与食物块之间的距离）；（2）初始俯仰角（锅铲相对于桌面的俯仰角）；（3）时间步 $t=0.25$时末端执行器的俯仰角速度。

A4.2 食物块的动态舀取

轨迹参数化。这里详细介绍使用为该任务设计的锅铲时末端执行器的轨迹（图 A2）。末端执行器的速度轨迹采用三次样条生成，并在五个时间步处对数值进行限制。轨迹仅在世界坐标系中的 X 方向和俯仰方向发生变化，其他方向则保持为零。任务策略需要学习的、用于定义轨迹的唯一数值是俯仰角速度，即时间 t=0.25s 时的俯仰速度，其范围为 -0.2rad/s 至0。2rad/s。正俯仰角速度表示锅铲较晚抬离桌面，而负俯仰角速度表示较早抬离桌面（效果见图 8）。任务策略输出的另外两个数值分别是锅铲相对于桌面的初始俯仰角（范围为 2 至 10 度），以及锅铲与食物块之间的初始距离（范围为 0。5cm 至 2cm）。通常，较大的初始俯仰角有助于将锅铲铲入平底食物块下方，而较小的角度则有助于铲入椭球形食物块下方。我们在仿真与现实中对具有各种几何形状和物理属性的食物块进行了大量测试后，设计了这类轨迹。

初始状态。 食物块被随机放置在锅铲前方一个 8x6cm 的盒状区域内；初始距离是相对于食物块的初始位置而言的。

任务策略参数化。 该任务策略同样使用 NAF 进行参数化。在此任务中，它将食物块的初始二维位置映射为三个动作参数：俯仰角速度、初始俯仰角和初始距离。

硬件设置。 我们使用市售的 OXO尼龙方形锅铲[2] 作为舀取用的锅铲。它的边缘相对较薄（约 1.2 毫米），有助于铲入薄片下方。我们在手柄上安装了一个带有高摩擦胶带的长方体状 3D打印适配器，以便平行夹爪牢固地夹住锅铲。我们为 Drake仿真器设计并使用了带适配器的锅铲的精确三维模型；但仿真中未对锅铲抵住桌面弯曲时产生的变形效应进行建模。

[2]链接：https://www.amazon.com/OXO-11107900LOW-Grips-Square-Turner/dp/B003L0OOSU

A5 实验的更多细节

A5.1 模拟适应

TableA2 展示了三种模拟目标环境中使用的仿真参数 e
tasks（结果见桌面3）。

Task	参数	设置					
		分布内	OOD-1	OOD-2	OOD-3	OOD-4	范围
摆锤	m_1	1.8	1.8	**0.5**	1.2	**0.4**	[1,2]
	m_2	1.2	**0.3**	1.8	1.8	**2.6**	[1,2]
	b_1	1.5	1.5	1.5	**10.0**	1.0	[1,2]
	b_2	1.5	1.5	1.5	**10.0**	2.0	[1,2]
推送	μ	0.1	**0.25**	0.05	0.15	**0.30**	[0.05,0.2]
	e	1e5	5e4	1e5	**5e6**	1e5	[1e4,1e6]
	μ_p	0.6	**0.1**	**0.9**	**0.1**	**0.15**	[0.2,0.8]
	y_p	0.05	-0.1	0.05	**-0.15**	0.1	$[-0.1,0.1]$
舀取	μ	0.30	**0.45**	**0.20**	0.30	0.40	[0.25,0.4]
	e	5e4	1e4	5e4	**1e6**	1e5	[1e4,5e5]
	g	1	0	1	0	**2**	{0,1}
	h	2.0	**1.4**	2.2	**2.8**	1.9	[1.5,2.5]

表 A2：三个任务在不同模拟目标环境中使用的仿真参数。分布外参数（超出适应策略训练所用范围）以粗体标出。对于 g 舀取任务，0 表示椭球体，1 表示圆柱体，2 表示长方体。

A5.2 真实适应

在图 A7 和图 A9 中，我们展示了 AdaptSim 在推送和舀取任务中的更多可视化结果。

A5.3 其他研究

仿真参数空间的选择。为回答问题 3，我们通过固定目标环境（双摆任务中的 OOD-1）并改变仿真参数空间，进行敏感性分析。在 OOD-1 中，分布外参数为 $m_2 = 0.3$，而 Ω 中的范围为 [1,2]。图 A3 展示了 AdaptSim 和两个系统辨识基线方法在适应后获得的奖励结果，其中范围逐渐远离$m_2 = 0.3$ 并移至 [1.1,2.1]、[1.2,2.2]和 [1.3,2.3]。系统辨识的性能迅速下降，而 AdaptSim 则更加稳健。

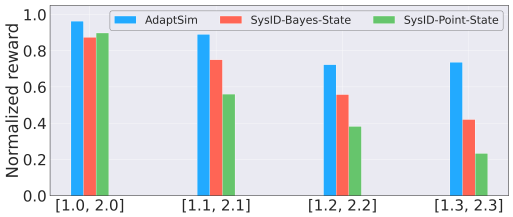


图 A3：在双摆任务的 OOD-1 设置下，AdaptSim 和系统辨识基线方法的适应结果；Ω 中具有不同的 m_2 范围，而目标环境中的 $m_2 = 0.3$ 保持不变。

系统辨识方法的陷阱。图A4 展示了仿真与现实之间的动力学失配，这说明了系统辨识方法的陷阱。我们绘制了从Ω 中随机采样仿真参数后，在固定机器人动作下得到的一组瓶子轨迹。我们还绘制了现实中以相同动作推送重型瓶子时的轨迹。部分真实轨迹与仿真轨迹无法很好地匹配，细微的失配也可能导致最终状态分歧（从而导致任务奖励不同）。

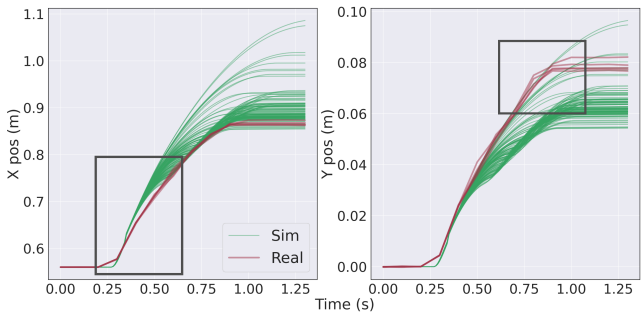


图 A4：在应用相同机器人动作的情况下，仿真域中的轨迹（绿色，使用随机采样的仿真参数设置进行仿真）与现实中重型瓶子的轨迹（红色）对比。真实动力学可能与仿真动力学呈分布外差异（黑色方框），但瓶子的最终位置可能处于分布内。

真实数据预算与任务性能渐近收敛之间的权衡。在第4.2 节中，我们介绍了 N，即阶段 2 开始时采样、随后独立进行自适应的初始仿真参数分布数量。真实数据预算（与 N成线性关系）和任务性能的收敛之间存在权衡。同时自适应更多仿真参数分布，可能有助于任务性能更快收敛，但也需要更多真实数据。图A5 展示了推动任务中轻瓶的实验效果。我们将 N 从 1 变到 4——每个仿真参数分布在每次迭代中使用 2 条轨迹。 $N=1$ 收敛速度较慢，渐近收敛效果也较差，这表明参数分布可能陷入低奖励区域。 $N=2$ 表现最佳，使用的真实轨迹数量最少且收敛最快。 N 越高，收敛越慢。请注意，收敛情况还取决于仿真参数空间 Ω 的维度——我们预计，当维度从推动任务中使用的 4 维增加时，$N>2$ 是获得最佳收敛速度所必需的。

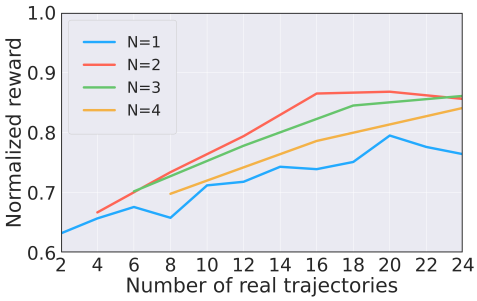


图 A5：在推动任务中使用轻瓶时，随着用于适应的真实轨迹数量变化，任务性能的收敛情况；同时改变 N，即阶段 2 中同时进行适应的仿真参数分布数量。

适应步长的敏感性分析。适应步长 δ 也会影响任务性能收敛—— δ 步长过小会导致收敛缓慢，而 δ 步长过大则可能无法收敛，因为仿真参数分布可能会大幅“越过”最优分布。图A6 展示了双摆任务在 OOD-1 设置下，适应步长从 0.05 到 0.20 的影响。 $\delta=0.10$ 表现最佳，而 $\delta=0.05$ 收敛较慢。 $\delta=0.15$ 也达到了相近的渐近性能，但适应过程中的奖励更稳定；而采用$\delta=0.20$ 时，奖励完全无法收敛。

仿真运行时间比较。与系统辨识基线相比，AdaptSim 在第一阶段训练适应策略所需的仿真运行时间显著更长。例如，在舀取任务中，SysID-Bayes 使用约 6 小时的仿真墙钟时间执行 10 次适应迭代；而 AdaptSim 在第一阶段需要 36 小时，在阶段 2 需要 30 分钟（即每次迭代 3 分钟），两者使用相同的计算配置。不过，我们会在舀取任务中对不同食物块重复使用同一个适应策略，从而摊销仿真成本。

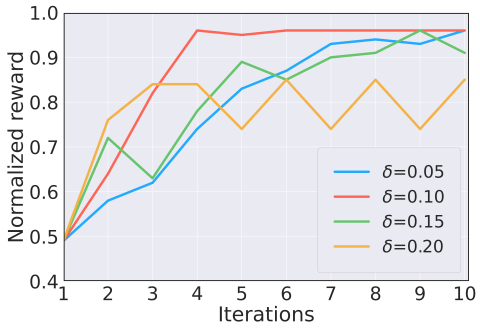
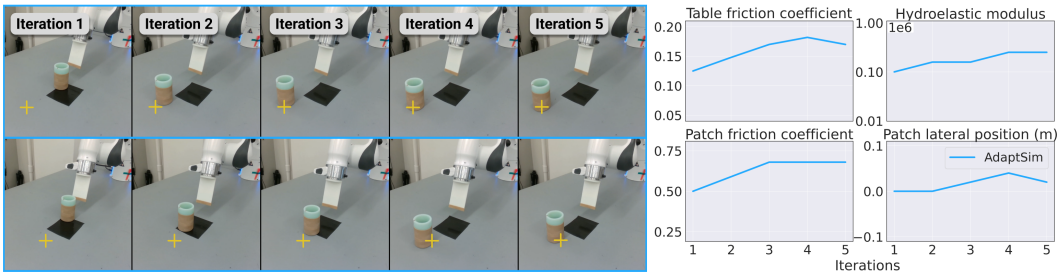


图 A6：摆锤任务的 OOD-1 设置下，使用不同适应步长 δ时每次适应迭代的归一化奖励。



Figu图 A7：推动任务在两个不同目标位置（黄色十字，顶部和底部）下随迭代的适应结果 rows）。右图显示推断出的仿真参数分布（仅显示均值）。

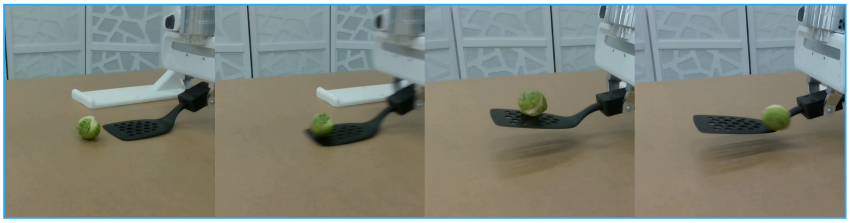


图 A8：AdaptSim 无法为舀起抱子甘蓝合成任务策略。我们认为此类环境与仿真域存在极大的分布外差异。



图 A9：使用 AdaptSim 舀取（上）巧克力葡萄干、（中）蘑菇片和（下）奥利奥饼干的适应结果。