What Went Wrong? Closing the Sim-to-Real Gap via Differentiable Causal Discovery

Anonymous Author(s)
Affiliation
Address
email

Abstract: Training control policies in simulation is more appealing than on real robots directly, as it allows for exploring diverse states in a safe and efficient manner. Yet, robot simulators inevitably exhibit disparities from the real world, yielding inaccuracies that manifest as the simulation-to-real gap. Existing literature has proposed to close this gap by actively modifying specific simulator parameters to align the simulated data with real-world observations. However, the set of tunable parameters is usually manually selected to reduce the search space in a case-by-case manner, which is hard to scale up for complex systems and requires extensive domain knowledge. To address the scalability issue and automate the parameter-tuning process, we introduce an approach that aligns the simulator with the real world by discovering the causal relationship between the environment parameters and the sim-to-real gap. Concretely, our method learns a differentiable mapping from the environment parameters to the differences between simulated and real-world robot-object trajectories. This mapping is governed by a simultaneously-learned causal graph to help prune the search space of parameters, provide better interpretability, and improve generalization. We perform experiments to achieve both sim-to-sim and sim-to-real transfer, and show that our method has significant improvements in trajectory alignment and task success rate over strong baselines in a challenging manipulation task. We include code and videos on our anonymized project website: https://sites.google.com/view/sim2real-compass.

Keywords: sim-to-real gap, reinforcement learning, causal discovery

1 Introduction

Training control policies directly on real robots poses challenges due to the potentially unsafe exploration and the sample complexity of deep reinforcement learning (RL) algorithms. Therefore, training in simulation is often necessary to perform diverse exploration of the state-action space in a safe and efficient manner [1, 2, 3, 4]. However, robot simulators are constructed based on simplified models and are thus approximations of the real world; for example, dynamics such as contact and collision are notoriously difficult to simulate with simplified physics [5, 6]. Even if the dynamics could be simulated accurately, not all physical parameters can be precisely measured in the real world and specified in simulation, e.g., friction coefficients, actuation delay, etc. As a result, a robot that is trained in a biased simulator could have catastrophic performance degradation in the real world [7, 8]. It is, therefore, critical to use simulators that closely mimic real-world dynamics to reduce this sim-to-real gap [9, 10, 11, 12].

Existing literature has proposed to close the sim-to-real gap by adjusting the parameters of the simulator to align the simulated data with the observed real data. For example, the `robosuite` [13] simulator provides APIs to modify over 400 different environment parameters. However, the search space grows exponentially with the dimension of the environment parameters. To mitigate this issue, various existing methods attempt to modify the parameters more efficiently, using gradient-based or

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

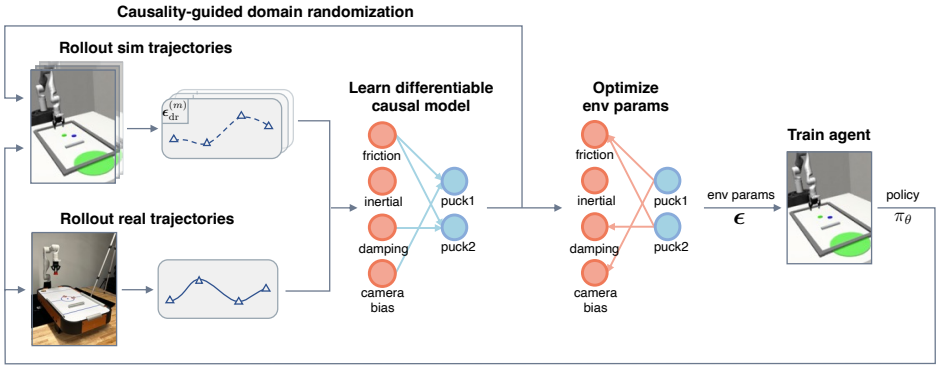

Figure 1: Overview of the COMPASS framework.

gradient-free sampling-based techniques [14, 15]. For example, in tasks involving quasi-static manipulation, such as sorting pegs or opening drawers, Chebotar et al. [14] chose to modify parameters related to the size, positions, and compliance; in tasks involving rich contacting, Muratore et al. [15] chose to modify mass, friction, and restitution coefficients.

However, this parameter selection process is typically carried out on a case-by-case basis, necessitating substantial domain knowledge to restrict the scope of environment parameters. This becomes challenging when dealing with simulations involving multiple interacting objects [16]. Furthermore, most existing methods lack the capability to offer explicit insights that can effectively guide the future deployment of more complex systems. They fail to offer direct answers to the question "*What went wrong with my simulator*" or, more specifically, "*What parameters should I tune to reduce the sim-to-real gap*" without *post hoc* analysis of the final parameters. In contrast, humans are good at analyzing complex events, identifying and eliminating irrelevant factors, and uncovering crucial cause-and-effect relationships. Such causal discovery capability enables an efficient and interpretable search for differences between two systems, providing a promising direction to bridge the gap between simulation and reality.

In this work, we propose a method that aims to align the simulator with the real world by discovering the **c**ausality between envir**o**n**m**ent **pa**rameter**s** and the **s**im-to-real gap (COMPASS) as illustrated in Figure 1. COMPASS learns a differentiable mapping, from the simulation environment parameters to the differences between simulated and real-world trajectories of dynamic robot-object interactions, governed by a simultaneously-learned causal graph. With the differentiable causal model fixed, COMPASS back-propagates gradients to optimize the simulation environment parameters in an end-to-end manner to reduce the domain gaps. Beyond the interpretability, the causal graph also helps to prune the parameter search space, thus improving the efficiency of domain randomization as well as the scalability. We summarize our contributions as follows:

1. We propose a new causality-guided parameter estimation framework to close the sim-to-real gap and improve agent performance in the real world.
2. We design a fully-differentiable model that explicitly embeds the causal structure to provide better interpretability, prune the search space of parameters, and improve generalization.
3. We empirically evaluate our method in both the simulation and the real world, which outperforms baselines in terms of trajectory alignment and task success rate with the same sample size.

2 Related Work

Closing the sim-to-real gap in robotic control tasks is often approached through domain randomization (DR) [7, 17, 18, 19, 20, 21, 22]. Although DR proves successful in numerous applications, particularly when access to real environments or data collected therein is unavailable, it is recognized that vanilla DR may result in overly conservative policies when the range of randomization is broad [23]. As an alternative approach for achieving sim-to-real transfer, system identification aims

to ascertain the parameters of the environment through limited interactions with real environments [15, 24, 16, 25, 26, 27] and has been combined with DR methods [14, 28]. We draw inspiration from this line of work in developing our parameter estimation framework, which learns causal relationships from dynamic robot-object interactions in order to facilitate the sim-to-real transfer.

Gradient-free Parameter Estimation for Sim-to-Real Transfer. Gradient-free parameter estimation methods typically utilize sampling-based methods to update the simulation environment parameters. Chebotar et al. [14] propose the SimOpt framework, which iteratively alters the distribution of environment parameters in simulation to mirror real environment rollouts via the cross-entropy method. Moving away from the assumption that the distribution of environment parameters follows a Gaussian distribution, as adopted in [14], Ramos et al. [28] develop BayesSim, which uses a Gaussian mixture model and optimizes the parameter distribution from a Bayesian perspective. Muratore et al. [15] propose Neural Posterior Domain Randomization (NPDR) which further removes assumptions on the environment parameter distribution by utilizing neural likelihood-free inference methods and could handle correlated parameters. It is worth noting that scaling up sampling-based methods becomes challenging when dealing with a large number of parameters. In contrast, `COMPASS` learns a difference-prediction model, leverages gradients to adjust the simulation parameters, and further improves scalability with learned causal structures.

Gradient-based Parameter Estimation for Sim-to-Real Transfer. Gradient-based methods typically employ a neural network to encapsulate the gradient landscape of parameter differences [24, 16, 25] or to model environment dynamics [26]. TuneNet [24] uses a neural network to predict the discrepancies in parameters based on the observations derived from two distinct environments. The Search Parameter Model (SPM) [25] is a binary classifier, with rollouts and parameters as input, which predicts whether a set of parameters is higher or lower than the target ones. Unlike TuneNet or SPM, we opt to predict observation differences using environment parameters as inputs. Allevato et al. [16] further expand the capabilities of TuneNet from handling a single parameter to managing a model with 5 parameters. In contrast, we demonstrate `COMPASS` is able to scale up to a 64-parameter system, a capacity notably larger than existing works. EXI-Net [26] implements a dynamics predictive model, conditioned on environment parameters, and identifies the most suitable parameters via back-propagation. While EXI-Net strives to model a broad spectrum of environments by separately modeling the known/explicit and implicit dynamics parameters, we aim to enhance sim-to-real transfer efficiency by capitalizing on the progressively discovered causal structure.

3 Methodology

3.1 Problem formulation: Markov Decision Processes and sim-to-real gap

A standard finite-horizon Markov Decision Process (MDP) is defined by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, p_0, \gamma, T)$, where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ is a state-transition probability function or probabilistic system dynamics, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $p_0 : \mathcal{S} \to \mathbb{R}_+$ is an initial state distribution, γ is a reward discount factor, and T is a fixed horizon. Let $\tau = (s_0, a_0, \ldots, s_T, a_T)$ be a trajectory of states and actions and $R(\tau) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t)$ is the trajectory reward. The objective of RL is to find parameters θ of a policy $\pi_\theta(a|s)$ that maximize the expected discounted reward over trajectories induced by the policy: $\mathbb{E}_{\pi_\theta}[R(\tau)]$, where $s_0 \sim p_0, s_{t+1} \sim P(s_{t+1}|s_t, a_t)$, and $a_t \sim \pi_\theta(a_t|s_t)$.

In our work, we assume that the simulator system dynamics are conditioned on environment parameters $\boldsymbol{\epsilon} \in \mathbb{R}^{|\mathcal{E}|}$, i.e., $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathbb{R}^{|\mathcal{E}|} \to \mathbb{R}_+$, where $\mathcal{E}$ is the set of all tunable environment parameters and $|\cdot|$ measures the cardinality of the set. Given a simulator parameterized by $\boldsymbol{\epsilon}$, the agent is optimizing $\mathbb{E}_{\pi_\theta, \boldsymbol{\epsilon}}[R(\tau)]$. When the simulation dynamics are very close to the real-world dynamics, one can expect the trajectory rollouts in the simulator to be close to that in the real world as well; here, an optimal agent trained in the simulator would expect near-optimal performance in the real world [29]. However, due to unmodeled dynamics and inaccurate environment parameters, the simulation dynamics are different from the real world (i.e., there exists a sim-to-real gap), resulting in different trajectory rollouts and hence degradation in real-world performance [1, 2, 7, 8, 10].

For better interpretability, we assume a factorized state space, i.e., $\mathcal{S} = \{\mathcal{S}_1 \times \cdots \times \mathcal{S}_K\}$, with $s_{k,t} \in \mathcal{S}_k$ representing the k-th factorized state at time t. Each component usually has explicit semantic meanings (i.e., an event or object's property) [30], which can be satisfied by state and action abstraction in general [31, 32, 33]. Similar to Chebotar et al. [14], we then define a factorized trajectory difference function:

$$d_k(\tau_{\text{sim}}, \tau_{\text{real}}) \coloneqq \sum_{t=0}^{T} \|s_{k,t,\text{sim}} - s_{k,t,\text{real}}\|_2, \quad \text{for } k = 1, 2, \ldots, K \tag{1}$$

The trajectory difference function is then $\boldsymbol{d} \coloneqq [d_1, \ldots, d_K]$, and the trajectory difference $\boldsymbol{d_\tau}$ is the output of $\boldsymbol{d}$ to measure the sim-to-real gap between of a pair of trajectories, $(\tau_{\text{sim}}, \tau_{\text{real}})$. In this work, we aim to find a simulation environment parameter $\boldsymbol{\epsilon}$ that minimizes the expectation of trajectory differences $\boldsymbol{d_\tau}$ under the same policy.

3.2 Learning causality between environment parameters and trajectory differences

To model the causality, `COMPASS` learns a mapping $\boldsymbol{d_\tau} = f_\phi(\boldsymbol{\epsilon}, \boldsymbol{a}; \mathcal{G})$, from the environment parameter $\boldsymbol{\epsilon}$ and action sequence $\boldsymbol{a} = [a_0, \ldots, a_T]$ to the trajectory difference $\boldsymbol{d_\tau}$, with a causal graph $\mathcal{G}$, whose nodes represent the variables to be considered and the edges represent the causal influence from one node to another node. We jointly learn the model parameter ϕ and discover the underlying causal graph $\mathcal{G}$ in a fully differentiable manner.

Causal Graph. The causal graph $\mathcal{G}$ plays a crucial role in the model by providing interpretability, pruning the search space of parameters, and improving the generalization to unseen parameters. Since we focus on the influence of environment parameter $\boldsymbol{\epsilon}$ to the trajectory difference $\boldsymbol{d_\tau}$, we can build the graph of pre-defined directed edges with a binary adjacency matrix of size $|\mathcal{E}| \times K$, where $1/0$ indicates the existence/absence of an edge. Motivated by previous works [34, 35, 36] that formulate the combinatorial graph learning into a continuous optimization problem, we design a sample-efficient pipeline by making the optimization of $\mathcal{G}$ differentiable. We sample elements of the graph $\mathcal{G}$ from a Gumbel-Softmax distribution [37], parametrized by $\psi \in [0, 1]^{|\mathcal{E}| \times K}$, i.e., $\mathcal{G}_{ij} \sim \texttt{GumbelSoftmax}(\psi_{ij}; \mathcal{T} = 1)$, where $\mathcal{T}$ is the softmax temperature. We denote the parameterized causal graph as $\mathcal{G}_\psi$. All elements (i, j) are initialized to ones to ensure the causal graph is fully connected at the beginning.

Structural Causal Model. Since the causal graph only describes the connection between variables, we also need a parameterized model to precisely represent *how* the causes influence the effects. We design an encoder-decoder structure in f, with $\mathcal{G}$ as a linear transformation applied to the intermediate features. First, the encoder operates on each dimension of $\boldsymbol{\epsilon}$ independently to generate features $z_{\boldsymbol{\epsilon}} \in \mathbb{R}^{|\mathcal{E}| \times d_z}$. Then the causal graph is multiplied by the features to generate the inputs for the decoder, i.e., $g_{\boldsymbol{\epsilon}} = z_{\boldsymbol{\epsilon}}^T \mathcal{G} \in \mathbb{R}^{d_z \times K}$, where d_z is the dimension of the feature. Similarly, the action sequence $\boldsymbol{a}$ is passed through the encoder and transformation to produce the feature of the action sequence $g_{\boldsymbol{a}} \in \mathbb{R}^{d_z \times K}$. Finally, $g_{\boldsymbol{\epsilon}} + g_{\boldsymbol{a}}$ is passed through the decoder to output the prediction $\hat{\boldsymbol{d_\tau}}$.

Differentiable Causal Discovery. Given a dataset $\mathcal{D} \coloneqq \{\boldsymbol{\epsilon}^{(m)}, \boldsymbol{a}^{(m)}, \boldsymbol{d_\tau}^{(m)}\}_{m=1,\ldots,M}$, the optimization objective to discover the underlying causal model consists of two terms:

$$\mathcal{L}_{\phi,\psi} \coloneqq \frac{1}{M} \sum_{m=1}^{M} \|f_\phi(\boldsymbol{\epsilon}^{(m)}, \boldsymbol{a}^{(m)}; \mathcal{G}_\psi) - \boldsymbol{d_\tau}^{(m)}\|_2^2 + \lambda |\mathcal{G}_\psi|, \tag{2}$$

where the first term is the mean squared error between the predicted trajectory differences and the real differences, and the second is a regularization term that encourages the sparsity of $\mathcal{G}$ ($|\mathcal{G}_\psi|$ is the number of edges in $\mathcal{G}$) with a positive scalar λ to eliminate the influence of irrelevant environment parameters. The detailed architecture of this causal model can be found in Appendix A.

3.3 Closing the sim-to-real gap via differentiable causal discovery

The main algorithm is shown in Algorithm 1. We highlight two parts of the algorithm here.

Algorithm 1 **C**ausality between envir**O**n**M**ent **PA**rameter**S** and the **S**im-to-real gap (COMPASS)

Input:
$\boldsymbol{\epsilon}_0 \in \mathbb{R}^{|\mathcal{E}|}$: initial guess of environment parameters,
ζ: threshold for sim-to-real gap,
$\texttt{Sim}(\cdot)$: simulator with controllable environment parameters
Output: $\boldsymbol{\epsilon}_i$, π_θ

Initialize agent policy π_θ
Initialize $f_\phi(\boldsymbol{\epsilon}, \boldsymbol{a}; \mathcal{G}_\psi)$ with $\psi \leftarrow \mathbb{1}_{|\mathcal{E}| \times K}$
for $i = 0, 1, 2, \ldots,$ *MaxIter* **do**
  Train π_θ in $\texttt{Sim}(\boldsymbol{\epsilon}_i)$
  $\{\tau_{\text{sim}}^{(n)}\}_{n=1,\ldots,N} \leftarrow$ Rollout N trajectories using π_θ in $\texttt{Sim}(\boldsymbol{\epsilon}_i)$
  $\{\tau_{\text{real}}^{(n)}\}_{n=1,\ldots,N} \leftarrow$ Rollout N trajectories using π_θ in the real environment
  Stop the iterations **if** $\textsc{Average}(\boldsymbol{d}(\tau_{\text{sim}}^1, \tau_{\text{real}}^1), \ldots, \boldsymbol{d}(\tau_{\text{sim}}^N, \tau_{\text{real}}^N)) \leq \zeta$
  $\mathcal{D} \leftarrow \varnothing$
  for $n \in \{1, \ldots, N\}$ **do** ▷ This loop can run in parallel
    $\{\boldsymbol{\epsilon}_{\text{dr}}^{(m)}\}_{m=1,\ldots,M} \leftarrow \textsc{CasualityGuidedDomainRandomization}(\boldsymbol{\epsilon}_i, \psi)$
    for $m \in \{1, \ldots, M\}$ **do**
      $\tau_{\text{sim}}^{(m)} \leftarrow$ Rollout π_θ in $\texttt{Sim}(\boldsymbol{\epsilon}_{\text{dr}}^{(m)})$
      $\boldsymbol{d_\tau}^{(m)} \leftarrow \boldsymbol{d}(\tau_{\text{sim}}^{(m)}, \tau_{\text{real}}^{(n)})$
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{\epsilon}_{\text{dr}}^{(m)}, \tau_{\text{real}}^{(n)}, \boldsymbol{d_\tau}^{(m)}\}$
  Jointly optimize model parameter ϕ and causal graph parameter ψ of $f_\phi(\boldsymbol{\epsilon}, \boldsymbol{a}; \mathcal{G}_\psi)$ ▷ Eq. 2
  $\boldsymbol{\epsilon}_{i+1} \leftarrow \textsc{UpdateEnvParam}(\boldsymbol{\epsilon}_i, f_\phi(\boldsymbol{\epsilon}_i, \boldsymbol{a}; \mathcal{G}_\psi))$ ▷ Eq. 3

Algorithm 2 Casuality-Guided Domain Randomization

function $\textsc{CasualityGuidedDomainRandomization}(\boldsymbol{\epsilon}, \psi)$
  for $r = 1, 2, \ldots, |\mathcal{E}|$ **do**
    if $\max(\psi_r) >$ *Threshold* **then** ▷ ψ_r is the r-th row of ψ
      $\{\boldsymbol{\epsilon}_r^{(m)}\}_{m=1,\ldots,M} \leftarrow \textsc{Uniform}(\boldsymbol{\epsilon}_r - \delta_r, \boldsymbol{\epsilon}_r + \delta_r)$ ▷ $\boldsymbol{\epsilon}_r$ is the r-th dimension
  return $\{\boldsymbol{\epsilon}^{(m)}\}_{m=1,\ldots,M}$

Causality-guided Domain Randomization. The learned causal graph is used to prune the search space. Since each element of the causal graph parameter ψ indicates the probability of an edge from the environment parameter to the trajectory difference, we can use ψ to determine whether to randomize a particular environment parameter or not. For instance, if the learned ψ indicates that there is no causal relationship between the torsional friction and the trajectory difference, the torsional friction will be excluded from randomization in the subsequent iteration, which enhances the efficiency of DR. The randomized environment parameters are sampled uniformly from certain ranges according to the current environment parameters. In this way, COMPASS automatically reduces the search space by orders of magnitude without human supervision or domain knowledge. The detail of causality-guided domain randomization is shown in Algorithm 2.

Environment Parameter Optimization. Owing to the full differentiability of our model, we can back-propagate the gradient information directly to the environment parameters to minimize the predicted sim-to-real gap:

$$J = \frac{1}{K} \sum_{k=1}^{K} f_{\phi,k}(\boldsymbol{\epsilon}, \boldsymbol{a}; \mathcal{G}_\psi), \quad \boldsymbol{\epsilon} \leftarrow \boldsymbol{\epsilon} - \eta \nabla_{\boldsymbol{\epsilon}} J \tag{3}$$

where $f_{\phi,k}$ is the k-th dimension of the output. We use the real action sequences and apply Eq. 3 multiple times until convergence. Note that if there is no causal relationship between an environment parameter and the trajectory difference, the gradient of that environment parameter will be zero since $\mathcal{G}$ is binary. This property could improve the robustness against noisy data and unseen environment parameter values during parameter optimization.

4 Experimental Results

4.1 Experimental setups

Mini Air Hockey with Obstacle. We design a challenging task of playing air hockey with a robot arm. To reach the goal, the agent needs to consider pusher-to-puck, puck-to-puck, puck-to-wall collisions, and surface properties of the hockey table. The task is to manipulate the pusher to hit the first puck, colliding with the second, which in turn needs to avoid the obstacle to reach the goal position by bouncing against the wall. Similar to Evans et al. [38], the action space includes the starting position of the pusher, hitting angle, and velocity. The state space includes the position of the two pucks ($K = 2$). The pusher, puck, and goal have a radius of 3cm, 2.55cm, and 15cm, respectively. This task requires precise actuation, since objects interact multiple times, propagating and compounding sim-to-real mismatches, such that the agent can experience a significant drop in success rate. Additionally, methods need to be robust against noisy and unmodeled dynamics. For instance, the floating force, in reality, is generated by a fan placed at the center of the table, and thus is non-uniform and stochastic. There are 64 tunable environment parameters in our experiments ($|\mathcal{E}| = 64$), and we use parameter notations in the format of *object@param_type@param*. We use Kinova Gen 3 robot arm in the real-world test bench and simulate in the `robosuite` [13] environment with MuJoCo physics engine [5]. More experimental configuration details can be found in Appendix B.

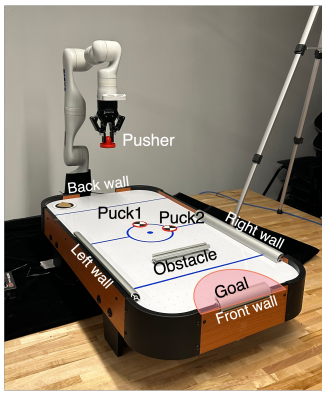

Figure 2: Experimental setup.

Baselines. As baselines, we select the state-of-the-art gradient-free sampling baselines NPDR [15], and two gradient-based baselines, TuneNet [24] and EXI-Net [26], discussed in Section 2.

4.2 Sim-to-sim trajectory alignment with known target environment parameters

In this experiment, we verify whether `COMPASS` can align trajectories between two different environments. We conduct experiments in simulation so that the ground truth environment parameters are known to us. Among the two environments, one is treated as the "real" (target) environment we want to align with, the other is the simulation environment we will fine-tune. We collect rollouts with a scripted stochastic policy. For each method except Tune-Net, we use $\textit{MaxIter} = 10, N = 10, M = 64$. For Tune-Net, we collect a dataset of size $\textit{MaxIter} \times N \times M = 6400$ to train the regression model. More details can be found in Appendix C.

The learned causal graph parameters ψ after 2 iterations ($i = 0, 1, 2$) are shown in Figure. 3. We observe that the learned causal graph is very sparse, reducing the search space by orders of magnitude without extensive domain knowledge. Our method is able to automatically discover different types of relevant environment parameters such as actuation, sensing, and dynamics. It is worth noting that only the damping of the right wall out of 4 sides has causality discovered with the trajectory

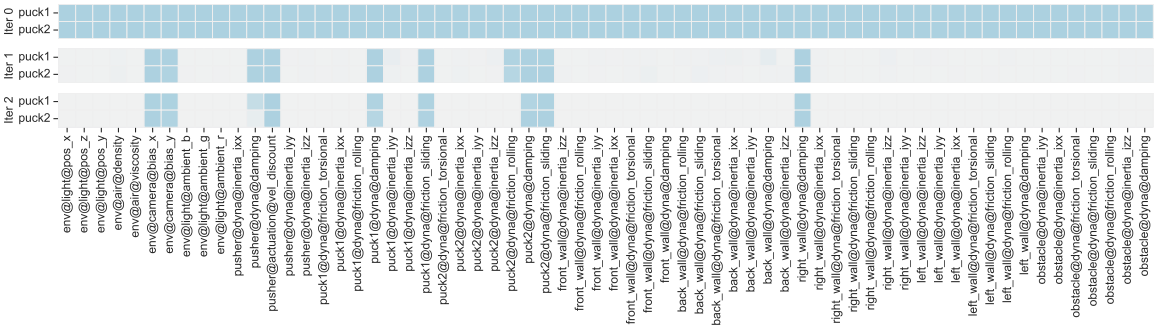

Figure 3: Learned causal graph parameters ψ. Darker colors present values closer to 1. We use parameter notations in the format of *object@param_type@param*.

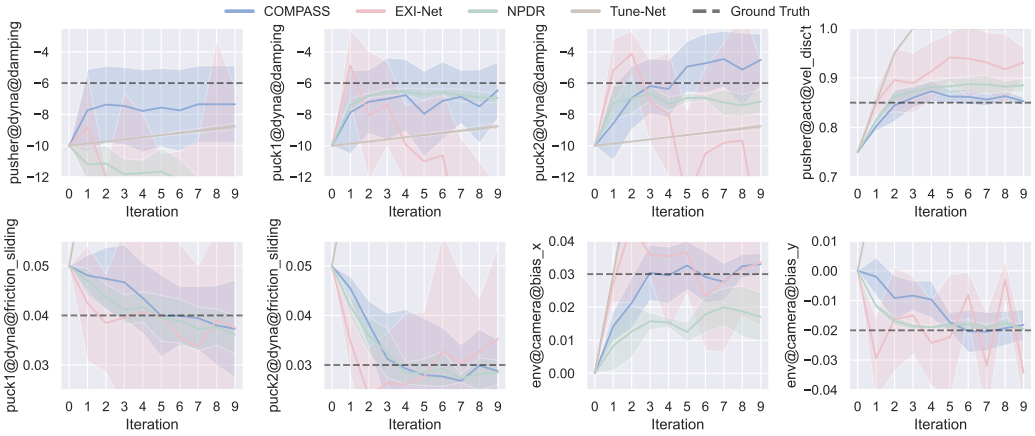

Figure 4: Environment parameters optimization. We show 8 parameters with discovered causality (as shown in Figure 3). The final mean absolute percentage errors (MAPE) are 0.22, 0.95, 0.29, 3.85 for COMPASS, EXI-Net, NPDR, Tune-Net, respectively. The solid lines represent the mean value across 5 random seeds, and the shaded area represents the standard deviation. The damping is negative due to the sign convention in MuJoCo.

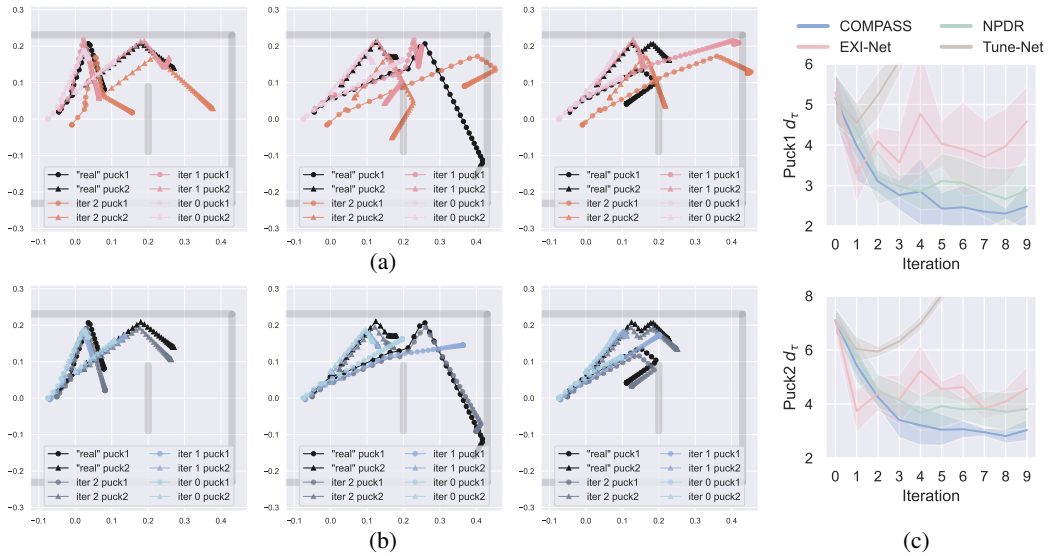

Figure 5: (a) and (b) visualize sampled trajectory-aligning results using EXI-Net and COMPASS, respectively. The overlapping of trajectories and the obstacle are due to the camera bias. (c) shows the mean trajectory differences throughout 10 iterations, evaluated with 5 random seeds and 10 pairs of rollouts per seed.

difference because we initialized the position of puck1 and puck2 in such a way that they almost only collided with the right wall. Though occasionally pucks did collide with the front wall or the obstacle, the speed at contacting is usually low and has minimal effects on the trajectory; therefore COMPASS placed low attention to such causality. Compared with the baseline methods, our proposed method provides unique interpretability that could guide the sim-to-real transfer in other systems.

The optimization process of 8 environment parameters with discovered causality is shown in Figure 4. We observe that COMPASS optimizes the environment parameters to approach the ground truth gradually, while baseline methods struggle to converge, especially for damping. Since Tune-Net trains a regression model to predict the difference between the current and the target set of environment parameters given trajectories, it does not work well when most of the environment parameters have negligible effects on the trajectories. Note that our method does NOT necessarily converge to the ground-truth environment parameters; rather, it aims to find a combination of environmental parameters to minimize the trajectory difference (more discussion in Section 5). The trajectory-aligning results are shown in Figure. 5(a)(b). It is observed that our method outperforms all the baselines in terms of the trajectory difference using the same number of "real" and simulation rollouts.

Table 1: Average trajectory difference of Puck1 and Puck2. "±" represents the standard deviation. We evaluate the results using 5 policies generated from independent runs and collect 10 trajectories for each run.

Traj. diff.	Low fan speed			High fan speed		
	Nominal	NPDR	COMPASS	Nominal	NPDR	COMPASS
Min	3.68 ± 0.07	2.81 ± 0.16	$\mathbf{2.37 \pm 0.10}$	2.23 ± 0.37	3.05 ± 0.46	$\mathbf{1.41 \pm 0.25}$
Max	10.77 ± 0.05	7.34 ± 1.41	$\mathbf{5.71 \pm 0.23}$	9.84 ± 0.56	10.36 ± 0.82	$\mathbf{8.17 \pm 1.34}$
Mean	7.60 ± 0.03	5.18 ± 0.77	$\mathbf{4.02 \pm 0.07}$	6.07 ± 0.40	5.63 ± 0.5	$\mathbf{3.97 \pm 0.45}$

Table 2: Agents' performance in the real environment.

	Low fan speed			High fan speed		
	Nominal	NPDR	COMPASS	Nominal	NPDR	COMPASS
Puck2 final dist. to goal center (↓)	0.35 ± 0.04	0.18 ± 0.05	$\mathbf{0.12 \pm 0.03}$	0.29 ± 0.09	0.15 ± 0.07	$\mathbf{0.13 \pm 0.02}$
Success rate (↑)	0.00 ± 0.00	0.39 ± 0.33	$\mathbf{0.80 \pm 0.09}$	0.20 ± 0.07	0.47 ± 0.41	$\mathbf{0.75 \pm 0.22}$

4.3 Sim-to-real with policy optimization in the loop

In this experiment, we first trained the agent in the initial simulation environment parameters with Soft Actor-Critic (SAC) [39]. Then, we applied COMPASS and the best-performing baseline in the sim-to-sim experiment, NPDR, to update the environment parameters and retrained the agent in the new simulation environment parameters. Finally, we deployed the agent in the real environment and reported the evaluation statistics. We set up two real environments by powering the electric fan at different speeds, referred to as *low fan speed* and *high fan speed*.

Upon inspecting Table. 1 and Table. 2, we first observe that (i) COMPASS consistently outperforms the nominal simulator and the NPDR baseline in terms of trajectory difference and success rate. Notably, COMPASS improved success rate by 105.1% and 59.6% compared with NPDR in the low and high fan speed settings, respectively. We hypothesize that COMPASS is more robust to the noisy and unmodeled dynamics in the real environment owing to the sparse causal model learned during the process. We also observe that (ii) the agent's real-world performance is correlated with the trajectory alignment performance. This is as expected given that previous works have proved that the policy performance degradation is bounded by the difference in transition distributions between two systems [29].

5 Discussion and Conclusion

In conclusion, COMPASS is a novel causality-guided framework to identify simulation environment parameters that minimize the sim-to-real gap. It has three salient features. Firstly, COMPASS requires little domain knowledge of the randomized environment parameters, enabling a more automated process for sim-to-real transfer. Secondly, COMPASS learns an interpretable causal structure, providing better generalization during environment parameter optimization and robustness against observational noise in real rollouts. Lastly, COMPASS employs a fully differentiable model to update the environment parameters, which mitigates the curse of dimensionality of the existing sampling-based methods. Through both simulation and real-world experiments, we verify that our proposed method outperforms the existing gradient-free and gradient-based parameter estimation methods in terms of trajectory alignment accuracy and the agent's success rate, while offering interpretability.

Limitations. With all the advantages of COMPASS discussed, some limitations of our method also suggest directions for future work. Similar to the existing gradient-based methods [24, 26, 25, 16], COMPASS could converge to local minima since the identifiable set of parameters may be coupled [40, 41], which would result in multiple local minima in the parameter space. Indeed, if the purpose is to find a specific combination of parameters that minimize the sim-to-real gap, it becomes less important whether it converges to the global optimum or not [24]. In addition, COMPASS finds a single combination of environment parameters rather than a distribution of them. Nevertheless, our method can maintain several particles of environment parameters as an empirical distribution without extensive modifications to the core algorithm [42].

References

[1] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[2] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[3] J. Francis, B. Chen, S. Ganju, S. Kathpal, J. Poonganam, A. Shivani, S. Genc, I. Zhukov, M. Kumskoy, A. Koul, et al. Learn-to-race challenge 2022: Benchmarking safe learning and cross-domain generalisation in autonomous racing. *arXiv preprint arXiv:2205.02953*, 2022.

[4] S. H. Park, G. Lee, J. Seo, M. Bhat, M. Kang, J. Francis, A. Jadhav, P. P. Liang, and L.-P. Morency. Diverse and admissible trajectory forecasting through multimodal context understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 282–298. Springer, 2020.

[5] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[6] B. Ellenberger. Pybullet gymperium. https://github.com/benelot/pybullet-gym, 2018–2019.

[7] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[8] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677, 2020.

[9] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics: Results of the 11th International Conference*, pages 621–635. Springer, 2018.

[10] J. Francis, N. Kitamura, F. Labelle, X. Lu, I. Navarro, and J. Oh. Core challenges in embodied vision-language planning. *Journal of Artificial Intelligence Research*, 74:459–515, 2022.

[11] S. Carpin, J. Wang, M. Lewis, A. Birk, and A. Jacoff. High fidelity tools for rescue robotics: results and perspectives. In *RoboCup 2005: Robot Soccer World Cup IX 9*, pages 301–311. Springer, 2006.

[12] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9474–9481. IEEE, 2021.

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

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

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

[16] A. Allevato, M. Pryor, and A. L. Thomaz. Multiparameter real-world system identification using iterative residual tuning. *Journal of Mechanisms and Robotics*, 13(3), 2021.

[17] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[18] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang. Understanding domain randomization for sim-to-real transfer. *arXiv preprint arXiv:2110.03239*, 2021.

[19] Q. Vuong, S. Vikram, H. Su, S. Gao, and H. I. Christensen. How to pick the domain randomization parameters for sim-to-real transfer of reinforcement learning policies? *arXiv preprint arXiv:1903.11774*, 2019.

[20] F. Muratore, C. Eilers, M. Gienger, and J. Peters. Data-efficient domain randomization with bayesian optimization. *IEEE Robotics and Automation Letters*, 6(2):911–918, 2021.

[21] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 36(1):1–14, 2019.

[22] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, et al. Domain randomization and generative models for robotic grasping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3482–3489. IEEE, 2018.

[23] O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv preprint arXiv:1908.05224*, 2019.

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

[25] Y. Du, O. Watkins, T. Darrell, P. Abbeel, and D. Pathak. Auto-tuned sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1290–1296. IEEE, 2021.

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

[27] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu. Sim-to-real transfer for biped locomotion. In *2019 ieee/rsj international conference on intelligent robots and systems (iros)*, pages 3503–3510. IEEE, 2019.

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

[29] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[30] W. Ding, H. Lin, B. Li, and D. Zhao. Generalizing goal-conditioned reinforcement learning with variational causal reasoning. *arXiv preprint arXiv:2207.09081*, 2022.

[31] D. Abel. A theory of abstraction in reinforcement learning. *arXiv preprint arXiv:2203.00397*, 2022.

[32] M. Shanahan and M. Mitchell. Abstraction for deep reinforcement learning. *arXiv preprint arXiv:2202.05839*, 2022.

[33] D. Abel, D. Arumugam, L. Lehnert, and M. Littman. State abstractions for lifelong reinforcement learning. In *International Conference on Machine Learning*, pages 10–19. PMLR, 2018.

[34] P. Brouillard, S. Lachapelle, A. Lacoste, S. Lacoste-Julien, and A. Drouin. Differentiable causal discovery from interventional data. *Advances in Neural Information Processing Systems*, 33:21865–21877, 2020.

[35] Y. Yu, J. Chen, T. Gao, and M. Yu. Dag-gnn: Dag structure learning with graph neural networks. In *International Conference on Machine Learning*, pages 7154–7163. PMLR, 2019.

[36] S. Lachapelle, P. Brouillard, T. Deleu, and S. Lacoste-Julien. Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*, 2019.

[37] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

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

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

[40] P. K. Khosla and T. Kanade. Parameter identification of robot dynamics. In *1985 24th IEEE conference on decision and control*, pages 1754–1760. IEEE, 1985.

[41] N. Fazeli, R. Tedrake, and A. Rodriguez. Identifiability analysis of planar rigid-body frictional contact. *Robotics Research: Volume 2*, pages 665–682, 2018.

[42] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1162–1176. PMLR, 30 Oct–01 Nov 2020. URL https://proceedings.mlr.press/v100/mehta20a.html.

[43] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: An open-source robot operating system. In *Workshops at the IEEE International Conference on Robotics and Automation*, 2009.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[45] A. Tejero-Cantero, J. Boelts, M. Deistler, J.-M. Lueckmann, C. Durkan, P. J. Gonçalves, D. S. Greenberg, and J. H. Macke. Sbi – a toolkit for simulation-based inference, 2020.

[46] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

[47] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.