From Simulation to Reality: Data-Efficient Evaluation of Causal Bayesian Networks

Zhitao Liang[1], **Maximilian Diehl**[1], **Nanami Hashimoto**[1], **Anne Koepken**[2], **Daniel Leidner**[2], **Karinne Ramirez-Amaro**[1], **and Emmanuel Dean**[1]

[1]Faculty of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden.
{zhitao, diehlm, nanami, karinne, deane}@chalmers.se
[2]DLR, Germany. {anne.koepken, daniel.leidner}@dlr.de

Abstract: The use of simulation data for learning Causal Bayesian Networks (CBN) can significantly reduce the need for costly and time-consuming real-world robot interaction. However, sim-to-real evaluation is a necessary step to deploy a simulation-learned CBN in reality due to the discrepancies in sensing, actuation, and environmental dynamics between simulation and reality. The main challenges in sim-to-real transfer are the absence of real-robot evaluation datasets tailored for CBN learning. In this paper, we propose task-agnostic guidelines for real-robot data collection specifically designed to evaluate CBNs. As a proof of concept, we apply them to a robotic platform performing a concrete task, i.e., the robot TIAGo performing a two-cube stacking task, and we collect the real-robot dataset from 100 trials. As a case study, we demonstrate how the dataset can be used to evaluate a simulation-trained CBN on real-robot executions, reporting 10% accuracy drop from sim-to-real transfer. We demonstrate that our approach provides a quantifiable measurement of sim-to-real transfer for CBNs, reducing the need for expensive real-robot data from thousands of trials for training purposes to a small sim-to-real validation set.

Keywords: Sim-to-Real Transfer, Causality in Robotics

1 Introduction

Simulation and a proper sim-to-real transfer can reduce the need for costly and time-consuming real-world robot interaction. Simulation environments allow faster, repeatable experiments, safer exploration of diverse conditions, and more scalable, lower-cost data acquisition than physical robots [1]. However, models trained in simulation often face a sim-to-real gap when transferred and deployed to real robots, caused by discrepancies in sensing, actuation, and environmental dynamics between simulation and reality [2]. Therefore, sim-to-real evaluation is necessary for applying large-scale simulation data as a complement to costly real-world samples.

Causal models, for example, Causal Bayesian Networks (CBNs), are a compelling candidate for investigating how simulation and sim-to-real transfer can reduce the need for expensive real-robot data. Their training processes are inherently data-demanding due to their statistical nature, while they offer a principled way to achieve sample-efficient learning in robotics by identifying the features relevant to task outcomes and therefore reducing data dimensionality crucial for decision-making [3]. Most existing causal learning approaches rely purely on simulation or human demonstration data, even though their models are intended to support real-robot reasoning [4, 5, 6, 7]. For causal methods that model cause-effect relationships between environmental features and their effects on task execution success, the sim-to-real gap leaves a critical question: are the cause–effect relations learned in simulation still valid under real-world conditions?

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

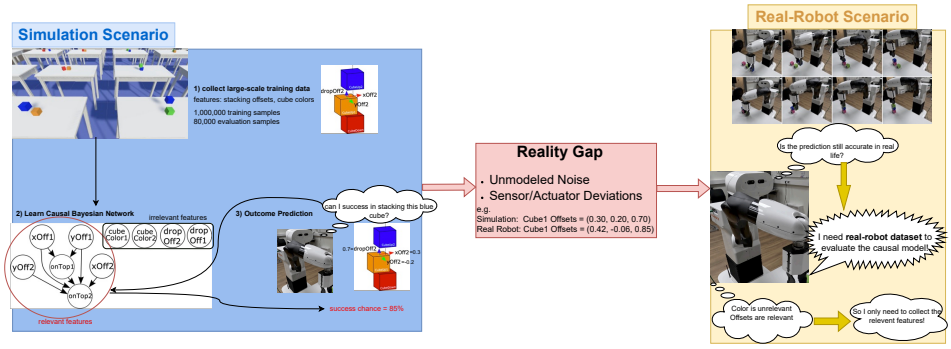


Figure 1: Simulation-learned CBN requires a real-robot dataset to evaluate whether its learned causal relationships transfer to reality. Since CBN predicts the probability of execution outcome using relevant features, the robot can collect the relevant features while ignoring the irrelevant features, improving data efficiency and rationality.

Some prior works attempt to model uncertain noise and include real-world applicability: Cannizzaro [8] modeled execution uncertainty in simulation, while Diehl et al.[9] and Brawer et al.[10] evaluated models with limited real-robot data. However, these real-world datasets are not publicly available, and a detailed description of the robotic setup, the requirements for real-data acquisition, and the data collection protocol are missing.

It is evident that the success of the sim-to-real models greatly depends on the quality of the dataset used to obtain the models. To properly evaluate the robustness of the sim-to-real gap, the models obtained from simulations should be evaluated in real-world experiments. For this, we need to carefully design and collect real-robot datasets under some principles: variables should match simulation definitions and semantics, measurements must be time-aligned, and experiments should span diverse conditions to capture realistic variations and noise. Furthermore, it is essential to understand how much sensing and information is needed: the dataset design must specify the necessary robot platform to perform the same task as in simulation, and the sensor modalities required to observe and measure the corresponding variable values. Despite the strong need, there is currently neither a publicly available real-robot dataset purpose-built for CBN learning and evaluation, nor an established methodology guiding the real-robot causality-structured data collection. Existing manipulation datasets [11, 12, 13] focus on control or perception benchmarking, but do not provide structured cause, effect, and outcome variables that are suitable for causal model learning. To address this gap, our second contribution is a set of task-agnostic real-robot data collection guidelines tailored to evaluate the models from CBN learning.

Furthermore, as a proof of concept, we apply these guidelines to collect and release a 100-trial real-robot dataset on a two-cube stacking task with a TIAGo robot as our third contribution[1], which addresses the problem of the lack of real-robot datasets for causal learning. Finally, as a case study for the use of the dataset, we use it in an existing CBN simulation-learning pipeline by extending a sim-to-real evaluation module to validate simulation-learned CBNs against real-world trials in terms of distribution robustness and predictive accuracy. The experiment demonstrates that our approach provides a quantifiable measurement of how well the CBN obtained from simulated data transfers to reality. Depending on the use case (e.g., the severity of consequences when failures occur), one can decide whether the sim-to-real performance is sufficient for real-robot deployment or whether further refinement with real-world data is required. Importantly, our sim-to-real evaluation quantifies how reliably simulation can serve as a complement for real-world data in CBN learning, therefore reducing costly robot experimentation.

This paper summarizes our main findings, presents ongoing experiments, and invites the community to discuss the definition of guidelines to facilitate real-world data collection.

[1]The Dataset is available at https://gitlab.com/craft_lab/causality-robotics/causalrobot_eurobin/eurobin_cubestacking_tiago_dataset.git

2 Preliminaries of Causal Learning Process

2.1 Causal Bayesian Networks (CBNs)

We adopt CBNs as the fundamental model since they capture both directed causal relations and probabilistic dependencies [14], different from other causal models such as structural equation models [15] and additive noise models [16]. A CBN is defined as a directed acyclic graph (DAG) $\mathcal{G} = (\mathbf{X}, A)$ [17], where nodes $\mathbf{X}$ represent N random variables (e.g., task-relevant states or outcomes), to describe each action; here, an action refers to a specific robot-executed task, such as picking up a cube and placing it at a target location. Arcs A denote causal relations between the variables. The joint distribution factorizes as:

$$P(X_1, ..., X_N) = \prod_{i=1}^{N} P(X_i | \Pi_{X_i}), \tag{1}$$

with Π_{X_i} the parents of X_i. Learning a Bayesian network from data usually involves two steps:

1. **Structure learning.** The process [9] employs the PC algorithm [18], which infers causal edges through conditional independence tests. Since many algorithms, including PC, do not handle causal connections between mixed discrete–continuous variables [9], continuous variables are first discretized into equal-frequency intervals. Alternative approaches are reviewed in [19].

2. **Parameter estimation.** Then, parameter learning is performed to estimate the local probability distributions of the factorization in Equation 1 (denoted as $\boldsymbol{\theta} = P(X_i|\Pi_{X_i})$). To fit $\boldsymbol{\theta}$ from the data, the Maximum Likelihood Estimation (MLE) is used in the pipeline, estimating all entries in the conditional probability tables $\theta_{x|\boldsymbol{u}}$, $\forall x \in \text{Val}(X_i)$, and $\forall \boldsymbol{u} \in \text{Val}(\Pi_{X_i})$, where $\text{Val}(X_i)$ signifies all possible values of X_i, and $\text{Val}(\Pi_{X_i})$ encompasses all combinations of potential values of Π_{X_i}.

2.2 Simulation-Based CBN Learning Pipeline

The existing simulation-based CBN learning pipeline from prior work [9, 20] is illustrated in the left part of Fig. 1. In the process, Simulation data is used to train and evaluate the obtained causal model. A large dataset is generated in a simulation environment (e.g., Unity 3D) by executing task variants with commanded variables. Then, a CBN is learned from simulations. This model is used to predict the success of an action given the current state representation and search for corrective action (variable combination predicted as successful) in case the action is expected to fail.

3 Guidelines for Real-Robot Dataset Collection

3.1 Dataset Principles for CBN learning

Datasets for causal models sim-to-real transfer must satisfy the following causal modelling and inference principles: (P1) The dataset should include the task outcome variable and all ancestor variables of it in the DAG. This specification allows the estimation of the corresponding Markov kernels $P(X_i|\Pi_{X_i})$ along all causal paths leading to task outcome using the datasets [14]; (P2) The dataset should include sufficient coverage of parent variable configurations for the estimation of conditional distributions [21]; (P3) Evaluating causal effects requires interventions (i.e., setting cause variables to specific values) to evaluate the effect of causes on outcomes [14]; (P4) CBN variables often abstract dynamic states at specific timepoints. In the dataset, maintaining semantic consistency across trials preserves the meaning of the conditional probability distributions [21]; (P5) The dataset should capture real-world noise and deviations [9].

3.2 Assumption

Since the causal model is first learned from simulation, we assume that all variables relevant to the task outcomes are defined and observed in the learning process and that there are no unmeasured

confounders affecting the learned causal graph. This assumption is reasonable in our sim-to-real setting, as the evaluation focuses on validating whether the simulation-discovered cause-effect relations hold on the real robot, rather than discovering new causal relations. In future work, the assumption could be relaxed by defining measurable metrics to verify the faithfulness of the learned DAG (e.g., measuring the identifiability of causal effects [22]).

3.3 Task-Agnostic Guidelines

1. **Define relevant cause variables and goal variables based on the causal graph learned from simulation. (P1)** Given the CBN over $\mathbf{X}$ learned from simulation, we extract subsets of variables relevant for evaluation. Specifically, we define cause variables $\mathbf{C} \subseteq \mathbf{X}$ and effect variables $\mathbf{E} \subseteq \mathbf{X}$. Variables outside $\mathbf{C} \cup \mathbf{E}$ are treated as irrelevant for sim-to-real evaluation. The success of an action is specified by a set of goal variables $\mathbf{G} \subseteq \mathbf{E}$ with predefined success conditions $\mathbf{G}_{\text{succ}}$. For example, in a cube-stacking task, a goal variable $\mathbf{G}$ may be `onTop`, with success defined as `onTop = TRUE`.

2. **Assess variable observability and sensor requirements. (P1)** Each variable in $\mathbf{C}, \mathbf{E}, \mathbf{G}$ must be observable from the robot's sensor and perception system.

3. **Include diverse data samples and controlled variations. (P2)** To make the CBN realistic about learning the causal relations, the data set should include sufficient variation between the values of the key variables. This can be achieved by systematically commanding different values of the controllable variables $\mathbf{C}$ during data collection.

4. **Ensure interventional ability of cause variables. (P3)** To ensure data diversity, the causal variables $\mathbf{C}$ should be actively manipulable on the real robot, reflecting the range of possible interventions available in simulation. This means that the robot should be able to directly set values of $\mathbf{C}$ independently of other variables, as in the simulation setup.

5. **Capture single-value data with time-aligned observations. (P4)** Rather than using continuous, time-varying sensor streams (e.g., end-effector trajectory), CBNs commonly work on static, single-value variables where each variable has a single value per trial. During real robot execution, these variables are single measurements extracted from raw sensor streams at specific and well-defined moments. For example, instead of storing the whole end-effector trajectory, we should only record the final placement position. To make such single-value observations comparable across trials, the extraction must be time-aligned. The measurement can be triggered by time stamps or sensor signals.

6. **Record real-world noise and execution deviations. (P5)** Since the values of $\mathbf{C}$ are precommanded before execution in the real-robot setup, there might be a deviation between the command and the actual execution. Recording the noisy real-world signals instead of the planned values allows us to evaluate the causal model under realistic conditions.

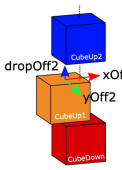
4 Experiments

As a proof-of-concept for the real-robot data collection guidelines and the sim-to-real evaluation stages of our pipeline, we implement a physical robot experiment in a concrete scenario: a Cube-Stacking environment. As defined in previous work [9], the Cube-Stacking environment consists of three cubes *CubeUp1*, *CubeUp2* and *CubeDown* (see the left part of Fig. 2). The goal is to place *CubeUp2 on top* of *CubeUp1* and *CubeUp1 on top* of *CubeDown*. All cubes have a side length of 5 cm. We describe the Cube-Stacking task with the help of ten variables:

$$\mathbf{X} = \{\texttt{xOff1}, \texttt{yOff1}, \texttt{dropOff1}, \texttt{onTop1}, \texttt{cubeColor1}, \texttt{xOff2}, \texttt{yOff2}, \texttt{dropOff2}, \texttt{onTop2}, \texttt{cubeColor2}\}$$

4.1 CBN Learning in Simulation

Following the simulated CBN learning step, we collect a training dataset of 1,000,000 samples and an evaluation dataset of 80,000 samples. Each data sample takes six seconds to generate in the simulation. All task variables $\mathbf{X}$ are sampled according to the right part of Fig. 2 for each 2-Stack



Variable	Meaning
$\texttt{xOff}_2$	$\text{cubeUp}_2.x - \text{cubeUp}_1.x$ (distance between cube centers)
$\texttt{yOff}_2$	$\text{cubeUp}_2.y - \text{cubeUp}_1.y$
$\texttt{dropOff}_2$	$(\text{cubeUp}_2.z - \text{cubeUp}_1.z)$ $-\text{cubeLength}$ (distance between cube surfaces)
$\texttt{onTop}_2$	indicates if cubeUp_2 is on top of cubeUp_1 after the stacking process
$\texttt{cubeColor}_2$	denotes cube color

Variable	Distribution	Range / Notes
`xOff1, xOff2`	$\mathcal{N}(0, 2)$ (in cm)	Truncated to $[-3, 3]$ cm
`yOff1, yOff2`	$\mathcal{N}(0, 2)$ (in cm)	Truncated to $[-3, 3]$ cm
`dropOff1, dropOff2`	$\mathcal{N}(1, 3)$ (in cm)	Truncated to $[0.4, 10]$ cm
`cubeColor1, cubeColor2`	$\mathcal{U}$(Red, Green, Blue, Orange)	Categorical uniform
`onTop1, onTop2`	[TRUE, FALSE]	Automatically determined post-drop

Figure 2: Left: Defines the causal BN variables for the 2-Stack task (variables that describe the stacking relationship between *CubeUp1* and *CubeDown* are defined analogously) [9]. Right: Distribution of initialization variables in simulation experiments.

experiment in the Unity3D simulation environment, where cubes drop according to the sampled parameters without a simulated robot. Fig. 3 illustrates the graphical structure of the causal model $\mathcal{G} = (\boldsymbol{V}, A)$ learned from the simulation training dataset.

4.2 Applying the Guidelines in the Real-Robot Setup

4.2.1 Define relevant cause variables and goal variables (P1)

Based on the obtained causal graph (see Fig. 3), we can define the outcomes $\mathbf{G} = \{\texttt{onTop1}, \texttt{onTop2}\}$, success condition $\mathbf{G}_{\text{succ}} = \{\texttt{onTop1} = 1 \,\&\, \texttt{onTop2} = 1\}$ and causes $\mathbf{C} = \{\texttt{xOff1}, \texttt{yOff1}, \texttt{xOff2}, \texttt{yOff2}\}$. We could define $\{\texttt{dropOff1}, \texttt{dropOff2}, \texttt{cubeColor1}, \texttt{cubeColor2}\}$ as irrelevant features for causal evaluation. We still record `dropOff1` and `dropOff2` in the dataset: While they are not used in our causal evaluation (the simulation-trained model excludes them), they can capture variation in cube placement and may prove useful for future analyses. These variables are therefore included and clearly labeled in the dataset for completeness. For sim-to-real evaluation, they can be optionally omitted to reduce dataset dimensionality.

4.2.2 Variable Observability and Sensor Requirements (P1)

In this experiment, outcomes `onTop1`, `onTop2` were observed by a human, while the robot has a sufficient perception setup to measure the execution offsets during an action.

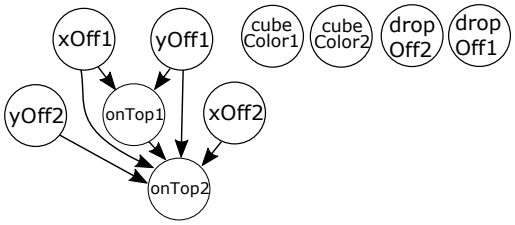


Figure 3: The obtained causal graph. Directed edges indicate causal dependencies between variables, for example, `onTop1` is causally influenced by `xOff1` and `yOff1`. Conversely, the absence of edges signifies no detected causal relationships; for instance, cube colors were found to have no causal effect on any other variables.

Because the robot's onboard camera was often occluded by its own arm during manipulation, we used an external Intel RealSense camera d435i for tracking. This camera was positioned on the opposite side of the table relative to the robot, overlooking the workspace, as shown in Fig. 4a. A unique ArUco marker was attached to one face of each cube, allowing cube detection, tracking and pose estimation using RGB images. These settings ensure the variables $\mathbf{X}$ are observable and measurable.

4.2.3 Diverse Data Samples and Controlled Variations (P2)

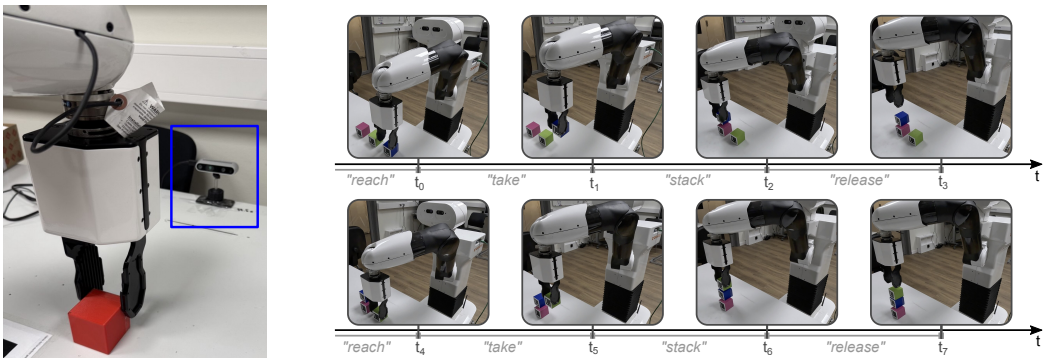
During each stack execution, the target stacking positions were determined before moving the robot's arm to each respective goal location (e.g., the position of *CubeDown* was selected before stacking *CubeUp1*). For instance, in Fig. 4b (top row, second image from the left), the stacking position for *CubeUp1* is established before the arm transitions into the pre-stack pose. To ensure a diverse dataset, we vary the commanded stacking offsets for 100 experiment trials, which allows us to include both successful and less optimal stacking scenarios. The full set of commanded stacking positions is presented in the second column of Table 1. We selected these commanded positions to cover various stack configurations, including nominal target locations and deliberate positional

offsets in different directions and heights. This strategy allows for capturing both successful stacks and failure cases caused by positioning errors.

Table 1: Summary statistics (mean ± SD) for each trial group.

Trials	Commanded	xOff1 (cm)	yOff1 (cm)	dropOff1 (cm)	xOff2 (cm)	yOff2 (cm)	dropOff2 (cm)
1–5	(0, 0, 0.5), (0, 0, 0.5)	-0.503 ± 0.229	-0.249 ± 0.134	0.586 ± 0.051	-0.858 ± 0.329	-0.269 ± 0.104	0.592 ± 0.065
6–10	(1, 0, 0.5), (1, 0, 0.5)	-0.530 ± 0.389	-0.164 ± 0.058	0.625 ± 0.088	-0.436 ± 0.495	-0.143 ± 0.074	0.590 ± 0.040
11–15	(2, 2, 0.5), (2, 2, 0.5)	1.56 ± 0.427	1.80 ± 0.012	0.670 ± 0.100	1.83 ± 0.157	1.84 ± 0.034	0.591 ± 0.051
16–20	(1, -1, 0.5), (1, -1, 0.5)	0.570 ± 0.302	-1.41 ± 0.070	0.610 ± 0.044	0.370 ± 0.257	-1.43 ± 0.100	0.595 ± 0.070
21–30	(1.5, -1.5, 0.5), (-1.5, 1.5, 0.5)	0.583 ± 0.425	-1.79 ± 0.068	0.638 ± 0.081	-2.32 ± 0.568	1.31 ± 0.102	0.601 ± 0.044
31–40	(0, -1.5, 0.5), (0, 0, 0.5)	-0.968 ± 0.622	-2.00 ± 0.068	0.655 ± 0.067	-0.743 ± 0.328	-0.459 ± 0.058	0.596 ± 0.060
41–45	(0, 1.0, 0.5), (0, 1.0, 0.5)	-0.807 ± 0.583	0.854 ± 0.319	0.651 ± 0.073	-0.430 ± 0.325	0.916 ± 0.337	0.577 ± 0.048
46–50	(0, 0, 1.0), (0, 0, 1.0)	-0.254 ± 0.367	-0.440 ± 0.040	1.11 ± 0.037	-0.422 ± 0.263	-0.440 ± 0.055	1.09 ± 0.061
51–55	(1, 0, 1.0), (1, 0, 1.0)	0.540 ± 0.251	-0.514 ± 0.045	1.11 ± 0.046	0.834 ± 0.166	-0.482 ± 0.041	1.03 ± 0.068
56–60	(2, 2, 1.0), (2, 2, 1.0)	1.72 ± 0.313	1.54 ± 0.031	1.11 ± 0.029	1.81 ± 0.311	1.71 ± 0.110	1.01 ± 0.046
61–65	(1, -1, 1.0), (1, -1, 1.0)	0.921 ± 0.361	-1.44 ± 0.091	1.05 ± 0.057	0.690 ± 0.240	-1.51 ± 0.092	1.09 ± 0.081
66–75	(1.5, -1.5, 1.0), (-1.5, 1.5, 1.0)	1.21 ± 0.309	-2.07 ± 0.042	0.949 ± 0.061	-1.84 ± 0.253	0.924 ± 0.042	1.20 ± 0.052
76–85	(0, -1.5, 1.0), (0, 0, 1.0)	-0.384 ± 0.262	-1.97 ± 0.039	1.08 ± 0.078	-0.533 ± 0.210	-0.471 ± 0.038	1.12 ± 0.056
86–90	(0, -1.0, 1.0), (0, -1.0, 1.0)	-0.373 ± 0.256	-1.38 ± 0.163	1.06 ± 0.082	-0.380 ± 0.173	-1.39 ± 0.132	1.08 ± 0.073
91–95	(0, -1.5, 0.5), (1.0, -1.0, 0.5)	-0.48 ± 0.385	-1.91 ± 0.114	0.649 ± 0.0406	0.649 ± 0.488	-1.41 ± 0.0649	0.53 ± 0.0474
96–100	(0, 1.5, 0.5), (1.0, 1.0, 0.5)	-0.437 ± 0.395	1.20 ± 0.0305	0.679 ± 0.0518	0.586 ± 0.126	0.705 ± 0.0504	0.595 ± 0.0890

Note: All values are in centimeters. Commanded offsets represent the target position of the object stack as $(\texttt{xOff1}, \texttt{yOff1}, \texttt{dropOff1})$, $(\texttt{xOff2}, \texttt{yOff2}, \texttt{dropOff2})$. Due to sensor and motor inaccuracies, the actual positions were different from the originally commanded ones.



(a) Camera positioning.

(b) Stacking process.

Figure 4: Experimental setup for real-robot data collection with the TIAGo platform.

4.2.4 Interventional Ability of Cause Variables (P3)

We use the TIAGo robot, equipped with a 7-DoF arm, an adjustable torso, and a parallel gripper, which is capable of performing the cube stacking task. Motion planning and execution for pick&place actions are conducted using MoveIt!. This setup allows us to stack the cube in a selected gripper position. Before moving to the stacking position, the robot should pre-command the x, y, and z offsets, which requires transforming the measured cube positions from the external camera frame to the robot's frame. To accurately align external camera observations with the robot's kinematic frame, we perform a calibration between the RealSense camera frame and the robot's $base\_link$ frame. This guarantees consistency between detected cube positions and robot motion planning. TIAGo's internal joint encoders and forward kinematics are used to estimate the end-effector pose during the plan and execution of each stacking action.

4.2.5 Capture Single-Value, Time-aligned Measurements (P4)

In our setup, the high-level motion planner defines a symbolic sequence of actions (e.g., *reach*, *take*, *stack*, *release*) that the robot must execute to complete the stacking tasks. For instance, the *stack* action is predefined in the planner, such as the end-effector moves to the target cube-dropping position with the gripper closed to 5 cm holding *CubeUp1*, and the *release* action corresponds to opening the gripper to 7cm to drop the holding cube. We use the action sequence to trigger time-aligned measurement. For example, we measure and record `xOff1`, `yOff1`, `dropOff1` after the first

stack event is finished and just before the first *release* event is executed (Fig. 4b, top row, third image from the left, timestamp t_2). The measurement for `xOff2`, `yOff2`, `dropOff2` is similar (Fig. 4b, timestamp t_6).

4.2.6 Record Real-World Noise and Execution Deviations (P5)

In addition to commanded $\mathbf{C}$, we store the actual measured offsets from the perception system. This allows us to capture discrepancies caused by motion errors, perception noise, or slippage. The deviations between commanded and measured values are shown in Table 1.

5 Case Study: Using the Dataset for Sim-to-Real Evaluation

After collecting the real-robot dataset, we demonstrate its usability by performing sim-to-real evaluation of a simulation-trained causal Bayesian Network (CBN). A sim-to-real evaluation module is integrated into the existing CBN learning pipeline (Sec. 2.2), as illustrated in Fig. 5.

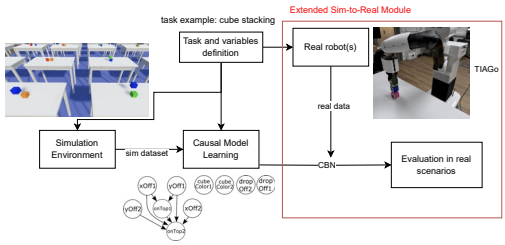


Figure 5: The pipeline for collecting and utilizing robotic data to learn and evaluate Causal Bayesian Networks in task-agnostic settings.

This case study focuses on how the collected dataset can quantitatively assess the transfer of causal knowledge from simulation to real-world trials. We first introduce the evaluation metrics, then assess the obtained causal model on the simulation test dataset, and finally apply the real-robot dataset to validate the model's generalizability from sim to real.

5.1 Evaluation Metrics

• **Mean Error:** The **Mean Error** measures the average absolute difference between the Conditional Probability Distributions (CPDs) of CBN learned from the simulation-trained data and those estimated from the real-robot dataset. Particularly, it compares the learned parameters $\theta^{\text{train}}_{x|\boldsymbol{u}}$ with their empirical counterparts $\theta^{\text{test}}_{x|\boldsymbol{u}}$ over all configurations of the variable X_i and its parent variables Π_{X_i}. This metric could assess the parameter robustness of a simulation-trained causal model when deployed in real-world conditions.

$$\frac{1}{|\text{Val}(X_i)| \cdot |\Pi_{X_i}|} \sum_{x \in \text{Val}(X_i)} \sum_{\boldsymbol{u} \in \Pi_{X_i}} \left| \theta^{\text{test}}_{x|\boldsymbol{u}} - \theta^{\text{train}}_{x|\boldsymbol{u}} \right| \tag{2}$$

This formulation assigns equal weight to the CPD parameter of each variable, reflecting an assumption that every conditional probability is equally important in representing the joint distribution of the CBN. This choice is motivated by the fact that errors in any part of the CBN can propagate through the graph and potentially affect downstream inference. However, we acknowledge that not all CPDs may contribute equally to the final goal outcome G. Future work could introduce goal-sensitive weightings, for example, by using the causal graph structure to prioritize CPDs with higher influence on $\mathbf{G}$ (e.g., by weighting according to path-specific causal effects or mutual information with $\mathbf{G}$). Such extensions would allow the metric to focus on the parameters that most critically impact task success.

• **Accuracy, Precision, Recall, F1-Score:** These metrics evaluate the predictive performance of the simulation-trained model on real-robot data. The ground-truth outcome is given by the binary success variable G, while the model outputs a probability distribution $P(\mathbf{G}_{\text{succ}}|\mathbf{C})$ for each test sample. To obtain binary predictions, we apply a decision threshold ϵ, classifying a trial as successful if $P(\mathbf{G}_{\text{succ}}|\mathbf{C}) \geq \epsilon$. We adopt $\epsilon = 0.5$ following the Bayes decision rule, which selects the class with the highest posterior probability under symmetric misclassification costs [23]. The resulting binary

predictions are compared against ground-truth labels to compute Accuracy, Precision, Recall, and F1-Score. While the existing pipeline reports only Accuracy, we extend it with additional metrics for a more comprehensive evaluation of predictive reliability.

5.2 Obtained Causal Model

To select the optimal number of discretization intervals for the continuous variables $\{\texttt{xOff1}, \texttt{yOff1}, \texttt{dropOff1}, \texttt{xOff2}, \texttt{yOff2}, \texttt{dropOff2}\}$, we performed a 10-fold cross-validation over a range of candidate values. Based on the mean validation performance, we chose to use seven equal-width discretization intervals within the value ranges (See the right part of Fig. 2).

Subsequently, we assessed the model's performance on the separate simulation evaluation dataset (Section 4.1), using the metrics defined in Section 5.1. The resulting performances are presented in the second column of Table 2. A failure prediction accuracy of 0.9425 is achieved.

5.3 Sim-to-Real Evaluation

To evaluate how well the causal model reflects the real-world scenarios, we assessed the obtained causal model on the 100 real-world samples using the same metrics (Section 5.1) to evaluate the probability distribution difference and the predictive performance. During the prediction, the stacking offsets recorded from the real-world experiments were discretized using the same discretization intervals learned during the training.

Table 2: Failure Prediction Results (Averaged over All Samples) for Simulation and Real Robot.

Metric	Simulation	Real Robot
Mean Error	0.04	0.18
Accuracy	0.94	0.82
Precision	0.93	0.97
Recall	0.95	0.79
F1 Score	0.94	0.89

The sim-to-real performance of the cube-stacking task (Tab. 2, third column) shows a 14% increase in mean error and a 10% decrease in accuracy from the simulated evaluation result. To the best of our knowledge, while prior work has focused on causal learning in either simulation or real settings, the average sim-to-real performance decrease has not been quantified in the literature. Thus, the extended sim-to-real evaluation module supported by the real-robot dataset provides a quantitative measure of how well the CBN obtained from simulated data transfers to reality. Depending on the application context (e.g., the severity of potential failure consequences), the researcher can assess whether the achieved accuracy is sufficient and either proceed with deployment or refine the model using real-world measurements.

6 Conclusion

In this work, we presented a case study using a collected real-robot dataset to perform sim-to-real evaluation of simulation-trained CBNs. We defined task-agnostic guidelines for real-robot data collection and demonstrated how the resulting structured dataset can be used to validate whether causal relations learned in simulation hold on a physical robot. Experiments on the TIAGo platform confirmed the method's ability to support systematic sim-to-real assessment, with ongoing tests on a different robot platform (Justin, see Appendix) indicating potential transferability.

Our results show that simulation-trained CBNs can be reliably evaluated with a relatively small number of real-robot trials, reducing the need for large-scale data collection while still providing meaningful validation. One limitation is that our approach is sensitive to environmental changes, as small variations in RGB-D sensor placement or gripper configuration can non-linearly distort pose estimates and thus alter the learned causal model. Future work should consider methods to quantify and mitigate these effects when extending sim-to-real evaluation to new experiment configurations.

Acknowledgments

This work is supported by EuRobin Project.

References

[1] 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. *CoRR*, abs/1703.06907, 2017. URL http://arxiv.org/abs/1703.06907.

[2] 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, 2019. URL https://arxiv.org/abs/1810.05687.

[3] M. Cibula, M. Kerzel, and I. Farkaš. *Learning Low-Level Causal Relations Using a Simulated Robotic Arm*, page 285–298. Springer Nature Switzerland, 2024. ISBN 9783031723599. doi:10.1007/978-3-031-72359-9_21. URL http://dx.doi.org/10.1007/978-3-031-72359-9_21.

[4] T. E. Lee, J. Zhao, A. S. Sawhney, S. Girdhar, and O. Kroemer. Causal reasoning in simulation for structure and transfer learning of robot manipulation policies, 2022. URL https://arxiv.org/abs/2103.16772.

[5] O. Ahmed, F. Träuble, A. Goyal, A. Neitz, Y. Bengio, B. Schölkopf, M. Wüthrich, and S. Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning, 2020. URL https://arxiv.org/abs/2010.04296.

[6] L. Castri, G. Beraldo, S. Mghames, M. Hanheide, and N. Bellotto. Ros-causal: A ros-based causal analysis framework for human-robot interaction applications, 2024. URL https://arxiv.org/abs/2402.16068.

[7] C. Uhde, N. Berberich, K. Ramirez-Amaro, and G. Cheng. The robot as scientist: Using mental simulation to test causal hypotheses extracted from human activities in virtual reality. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8081–8086, 2020. doi:10.1109/IROS45743.2020.9341505.

[8] R. Cannizzaro, M. Groom, J. Routley, R. O. Ness, and L. Kunze. Physics-based causal reasoning for safe & robust next-best action selection in robot manipulation tasks. *CoRR*, 2024.

[9] M. Diehl and K. Ramirez-Amaro. A causal-based approach to explain, predict and prevent failures in robotic tasks, 2022. URL https://arxiv.org/abs/2209.05255.

[10] J. Brawer, M. Qin, and B. Scassellati. A causal approach to tool affordance learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8394–8399, 2020. doi:10.1109/IROS45743.2020.9341262.

[11] A. Hundt, V. Jain, C.-H. Lin, C. Paxton, and G. D. Hager. The costar block stacking dataset: Learning with workspace constraints, 2019. URL https://arxiv.org/abs/1810.11714.

[12] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei. Roboturk: A crowdsourcing platform for robotic skill learning through imitation, 2018. URL https://arxiv.org/abs/1811.02790.

[13] A. K. et al. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL https://arxiv.org/abs/2403.12945.

[14] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 052189560X.

[15] K. A. Bollen. *Structural Equation Models with Observed Variables*, chapter Four, pages 80–150. John Wiley Sons, Ltd, 1989. ISBN 9781118619179. doi:https://doi.org/10.1002/9781118619179.ch4. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118619179.ch4.

[16] J. Peters, J. Mooij, D. Janzing, and B. Schölkopf. Causal discovery with continuous additive noise models, 2014. URL https://arxiv.org/abs/1309.6779.

[17] M. Scutari. Learning bayesian networks with the bnlearn r package. *Journal of Statistical Software*, 35(3):1–22, 2010. doi:10.18637/jss.v035.i03. URL https://www.jstatsoft.org/index.php/jss/article/view/v035i03.

[18] D. Colombo and M. H. Maathuis. Order-independent constraint-based causal structure learning. *J. Mach. Learn. Res.*, 15(1):3741–3782, Jan. 2014. ISSN 1532-4435.

[19] M. J. Vowels, N. C. Camgoz, and R. Bowden. D'ya like dags? a survey on structure learning and causal discovery. *ACM Comput. Surv.*, 55(4), Nov. 2022. ISSN 0360-0300. doi:10.1145/3527154. URL https://doi.org/10.1145/3527154.

[20] M. Diehl and K. Ramirez-Amaro. Why Did I Fail? A Causal-Based Method to Find Explanations for Robot Failures. *IEEE Robotics and Automation Letters*, 7(4):8925–8932, Oct. 2022. ISSN 2377-3766. doi:10.1109/LRA.2022.3188889. URL https://ieeexplore.ieee.org/abstract/document/9816136.

[21] J. Peters, D. Janzing, and B. Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT press, 2017.

[22] D. Kong, S. Yang, and L. Wang. Identifiability of causal effects with multiple causes and a binary outcome. *Biometrika*, 109(1):265–272, 2022.

[23] R. Golden. Statistical pattern recognition. In N. J. Smelser and P. B. Baltes, editors, *International Encyclopedia of the Social Behavioral Sciences*, pages 15040–15044. Pergamon, Oxford, 2001. ISBN 978-0-08-043076-8. doi:https://doi.org/10.1016/B0-08-043076-7/00619-7. URL https://www.sciencedirect.com/science/article/pii/B0080430767006197.

Appendix

A Transferability to Different Robot Platforms

We have also initiated data collection using the Justin robot platform (see Fig. 6). So far, we have conducted ten repetitions and achieved a sim-to-real success rate of xyz, serving as a proof of concept for the transferability of our causal model to different robot platforms.

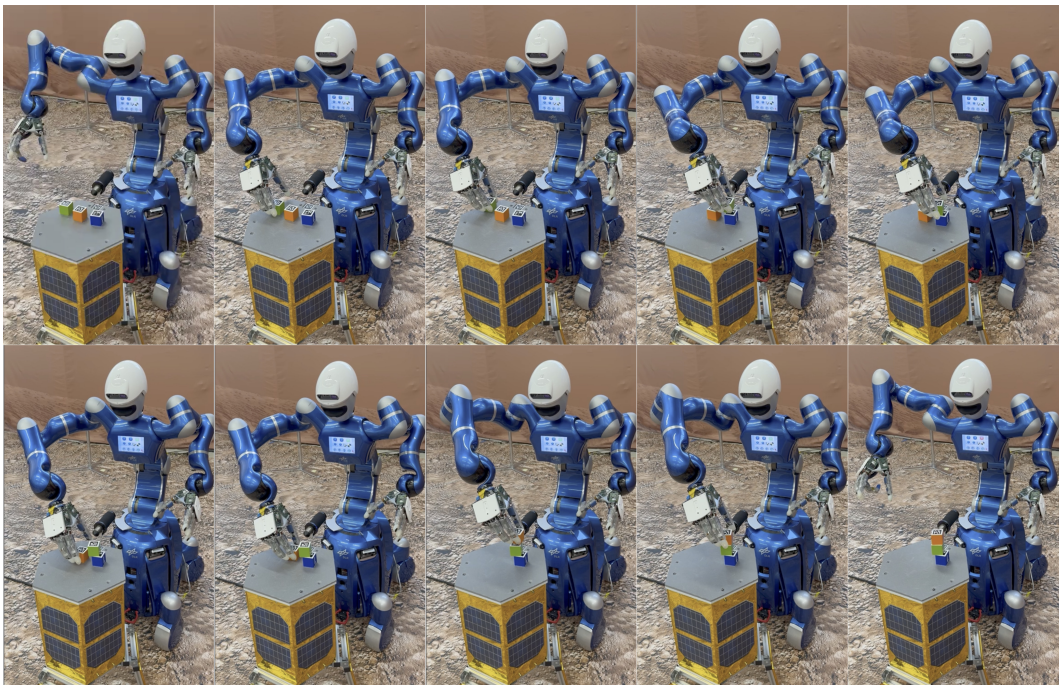


Figure 6: Stacking experiments performed with the Justin robot.