Why Did I Fail? A Causal-Based Method to Find Explanations for Robot Failures

Maximilian Diehl, *Graduate Student Member, IEEE*, and Karinne Ramirez-Amaro, *Member, IEEE*

***Abstract*—Robot failures in human-centered environments are inevitable. Therefore, the ability of robots to explain such failures is paramount for interacting with humans to increase trust and transparency. To achieve this skill, the main challenges addressed in this letter are I) acquiring enough data to learn a cause-effect model of the environment and II) generating causal explanations based on the obtained model. We address I) by learning a causal Bayesian network from simulation data. Concerning II), we propose a novel method that enables robots to generate contrastive explanations upon task failures. The explanation is based on setting the failure state in contrast with the closest state that would have allowed for successful execution. This state is found through breadth-first search and is based on success predictions from the learned causal model. We assessed our method in two different scenarios I) stacking cubes and II) dropping spheres into a container. The obtained causal models reach a sim2real accuracy of 70% and 72%, respectively. We finally show that our novel method scales over multiple tasks and allows real robots to give failure explanations like "the upper cube was stacked too high and too far to the right of the lower cube."**

***Index Terms*—Acceptability and trust, learning from experience, probabilistic inference.**

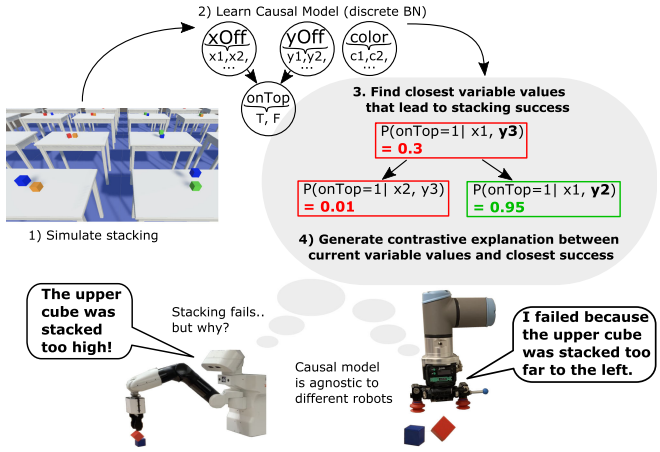


Fig. 1. Depicts our method that allows robots to explain their failures. First, we learn a causal model from simulations (steps 1, 2). A contrastive explanation is generated upon task failures (steps 3, 4). Finally, the obtained models are evaluated on two different tasks (cube stacking and sphere dropping) and transferred to two different robots that provide explanations when they commit errors.

I. Introduction

ONE important component in human interactions is the ability to explain one's actions, especially when failures occur [1], [2]. It is argued that robots need this skill if they were to act in human-centered environments on a daily basis [3]. Moreover, explainability is shown to increase trust and transparency in robots [1], [2], and the diagnoses capabilities of a robot are crucial for correcting its behavior [4].

There are different types of failures, e.g., task recognition errors (an incorrect action is learned) and task execution errors (the robot drops an object) [5], [6]. In this work, we focus on explaining execution failures. For example, a robot is asked to stack two cubes (see Fig. 1). The robot will first pick up a cube and move its gripper above the goal cube. However, due to sensor and motor inaccuracies, the robot places its gripper slightly shifted to the left, which results in an imperfect cube alignment between the cubes. Therefore, the upper cube lands on the goal but bounces to the table. In such a situation, we expect the robot to reason about what went wrong and generate an explanation based on its previous experience, e.g., "I failed because the upper cube was dropped too far to the left of the lower cube."

Typically, explanations are based on the concept of causality [7]. Obtaining a causal model of the environment is often addressed through statistical methods that learn a mapping between possible causes (preconditions) and the action-outcome [4], [8]. However, such statistical models alone are not explanations in itself [1] and require another layer that interprets these models to produce explanations. Another problem is that a considerable amount of data is needed to learn cause-effect relationships. In this case, training such models using a simulated environment will allow a faster and more extensive experience acquisition [4].

In this letter, we propose a method for generating causal explanations of failures based on a causal model that provides robots with a partial understanding of their environment (see Fig. 1). First, we learn a causal Bayesian network from simulated task executions, tackling the problem of knowledge acquisition. We also show that the obtained model can transfer the acquired knowledge (experience) from simulation to reality and is agnostic to several real robots with different embodiments. Second, we propose a new method to generate explanations of execution failures based on the learned causal knowledge. Our method is based on a contrastive explanation comparing the variable

Manuscript received 24 February 2022; accepted 17 June 2022. Date of publication 6 July 2022; date of current version 12 July 2022. This letter was recommended for publication by Associate Editor D. Losey and Editor G. Venture upon evaluation of the reviewers' comments. This work was supported by Chalmers AI Research Centre (CHAIR). *(Corresponding author: Maximilian Diehl.)*

The authors are with the Faculty of Electrical Engineering, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden (e-mail: diehlm@chalmers.se; karinne@chalmers.se).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3188889, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3188889

parametrization associated with the failed action with its closest parametrization that would have led to a successful execution, which is found through breadth-first search (BFS). Finally, we analyze the benefits of this method in two different scenarios: I) stacking cubes and II) dropping spheres into a container.

To summarize, our contributions are as follows:

1) We present a novel method to generate contrastive causal explanations of action failures based on causal Bayesian networks.
2) We demonstrate how causal Bayesian networks can be learned from simulations, exemplified in a cube stacking and sphere dropping scenario and provide extensive real-world experiments that show that the obtained causal models are transferable from simulation to reality without any retraining. Our method is agnostic to various robot platforms with different embodiments and scales over multiple tasks and scenarios. We, thus, show that the simulation-based model serves as an excellent prior experience for the explanations, making them more generally applicable.

II. Related Work

A. Causality in Robotics

Despite being acknowledged as an important concept, causality is relatively underexplored in the robotics domain [2], [9]. Some works explore causality to distinguish between task-relevant and -irrelevant variables [10]. For example, CREST [11] uses causal interventions on environment variables to discover which of the variables affect an RL policy. They find that excluding irrelevant variables positively impacts generalizability and sim-to-real transfer. In [12] a set of causal rules is defined to learn to distinguish between unimportant features in physical relations and object affordances. Brawer *et al.* present a causal approach to tool affordance learning [9]. Some works explore Bayesian networks to learn statistical dependencies between object attributes, grasp actions, and a set of task constraints from simulated data [13]. While the main objective is to use graphical models to generalize task executions, these works don't look into the question of how these models can be utilized for failure explanations. A different letter [14] investigates the problem of learning causal relations between actions in household-related tasks. They discover, for example, that there is a causal connection between opening a drawer and retrieving plates from human demonstrations. They only retrieve causal links between actions, while we focus on causal relations between different environment variables, like object features and the action outcome.

B. Learning Explainable Models of Cause-Effect Relations

In the planning domain, cause-effect relationships are represented through (probabilistic) planning operators [15]. Mitrevksi *et al.* [4] propose the concept of learning task execution models, which consists of learning symbolic preconditions of a task and a function approximation for the success model, based on Gaussian Process models. They noted that a simulated environment could be incorporated for a faster and more extensive experience acquisition, as proposed in [13]. Human virtual demonstrations have been used to construct planning operators to learn cause-effect relationships between actions and observed state-variable changes [15]. However, even though symbolic planning operators are considered human-understandable, they are not explanations in themselves, thus requiring an additional layer to interpret the models and generate failure explanations.

Some other works also aim to learn probabilistic action representations experience to generalize the acquired knowledge. For example, learning probabilistic action effects of dropping objects into different containers [8]. Again, the main objective is to find an intelligent way of generalizing the probability predictions for a variety of objects, e.g., bowl vs. bread box, but their method does not include any understanding of why there is a difference in the dropping success probabilities between these different objects.

C. Contrastive Explanations

Contrastive explanations are deeply rooted in the human way of generating explanations [1]. This also had a significant impact on explanation generation in other fields like Explainable AI Planning (XAIP) [16]. In XAIP, typical questions that a machine should answer are *why a certain plan was generated vs. another one?* or *why the plan contains a particular action a_1 and not action a_2?* [16], [17]. We, however, are mostly interested in explaining why specific actions failed based on environment variables like object features. A method for explaining synthesis failures of high-level robot task specifications (encoded through Linear temporal logic formulae) is presented in [18]. However, the causes need to be explicitly modeled (violations of specification constraints), while, in our approach, the causes are automatically detected during the BN learning process. Das *et al.* generate verbal failure explanations [19], by learning an encoder-decoder network that maps current information about the robot and environment state into a vector of words. However, the method does not scale well since it requires data with annotations about each failure cause. Our approach only requires annotations regarding the action success, which can be binary and are generally easier obtainable. Additionally, we encode the explanations directly in the causal structure of the different state variables instead of learning a black-box model. In a follow-up study [20], the authors use MOTIFNET [21] to autonomously detect spatial relationships and object attributes in a given scene. Then, pairwise ranking is used to filter out the subset of relevant relations for a particular explanation. Annotations for pairwise preferences of one relation over another need to be provided for training an SVM, which cannot be easily automated since they require human input. Due to the close relation to our approach, we discuss these works [19], [20] in more detail in Section V-D.

III. Our Approach to Explaining Failures

Our proposed approach consists of three main steps: A) Identification of the variables used in the analyzed task; B) Learning a Bayesian network which requires to 1) Learn a graphical representation of the variable relations (structure learning) and 2) to learn conditional probability distributions (parameter learning); and C) Our proposed method to explain failures, based on the previously obtained model.

A. Variable Definitions and Assumptions

Explaining failures, requires to learn the connections between possible causes and effects of an action. We describe an action via a set of random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$, which need to be defined by the experiment designer during the experiment setup. We require $\mathbf{X}$ to contain a set of treatment

variables $C \subset \mathbf{X}$, which describe potential causes, and outcome (effect) variables $E \subset \mathbf{X}$. Then, the goal of causal inference is to estimate the effect of C on E [22].

Data samples for learning the causal model can, in principle, be collected in simulation or the real world. A data sample d consists of a particular parametrization of $\mathbf{X}$, which we define as $d = \{X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n\}$, where n denotes the number of variables. It is important to sample values for possible causes C randomly. Randomized controlled trials are referred to as the gold standard for causal inference [3] and allow us to rule out the possibility of unmeasured confounders. Consequently, all detected relations between the variables $\mathbf{X}$ are indeed causal and not merely correlations. Besides the apparent advantage of generating truly causal explanations and avoiding the danger of possible confounders, causal models can also answer interventional questions. In contrast, non-causal models can only answer observational queries. The experiment must satisfy the sampled variable values before executing the action for data collection. E is measured at the end of the experiment.

We define another set $X_{\text{goal}} = \{ d_{\text{goal}_1}, d_{\text{goal}_2}, \ldots, d_{\text{goal}_h} \}$ that contains all possible variable parametrizations that denote a successful action execution. Then, an action is successful *iff* its parametrization $d \in X_{\text{goal}}$. Note, that it is out of scope of this letter, to discuss methods that learn X_{goal}, but rather assume X_{goal} to be provided a priori. In other words, we assume that the robot knows how an unsuccessful task execution is defined in terms of its outcome variables E and is thus able to detect it by comparing the action execution outcome with X_{goal}. Note, however, that the robot has no a-priori knowledge about which variables in $\mathbf{X} = X_1, X_2, \ldots, X_n$ are in C or E, nor how they are related. This knowledge is generated by learning the Bayesian network.

To efficiently learn a Bayesian network, some assumptions are needed to handle continuous data [23], mainly because many structure learning algorithms do not accept continuous variables as parents of categorical variables [24]. In our case, this means that some effect variables from E could not have continuous parent variables out of C, which would likely result in an incorrect Bayesian network structure. As a preprocessing step, we therefore discretize all continuous random variables out of $\mathbf{X}$ into intervals (X_{int}) with an equal number of samples.

B. Our Proposed Pipeline to Learn the Causal Model

Formally, Bayesian networks are defined via a graphical structure $\mathcal{G} = (\mathbf{V}, A)$, which is a *directed acyclic graph* (DAG), where $\mathbf{V} = \{X_1, X_2, \ldots, X_n\}$ represents the set of nodes and A is the set of arcs [24]. Each node $X_i \subseteq \mathbf{X}$ represents a random variable. Based on the dependency structure of the DAG and the *Markov property*, the *joint probability distribution* of a Bayesian network can be factorized into a set of *local probability distributions*, where each random variable X_i only depends on its direct parents Π_{X_i}:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | \Pi_{X_i}) \tag{1}$$

Learning a Bayesian network from data consists of two steps:

1) Structure Learning: The purpose of this step is to learn the graphical representation of the network $\mathcal{G} = (\mathbf{V}, A)$ and can be achieved by a variety of different algorithms. An extensive survey of potentially equally valid structure learning algorithms, like [25] is presented in [22]. For the remainder of this letter, we choose the Grow-Shrink [26] algorithm (gs) to learn $\mathcal{G}$. gs falls into the category of *constraint-based-algorithms*, which use statistical tests to learn conditional independence relations (also called "constraints") from the data [27]. Note that learning plausible assumptions about causal relations is one of the biggest challenges in the whole process of causal inference [28]. For example, in some cases, it is challenging to determine the direction of causal relations purely from the joint distribution of the observational data without additional interventional experiments, additional domain knowledge, or certain assumptions about the data distribution [29]. Structure learning is an active field of research [28], and we will use the learned structure to generate causal-based explanations of failures. Therefore, we assume that the outcome of the structure learning step is indeed correct or has been manually revised based on domain knowledge.

2) Parameter Learning: The purpose of this step is to fit functions that reflect the *local probability distributions*, of the factorization in formula (1). We utilize the maximum likelihood estimator (*mle*) to generate a conditional probability table based on the previously obtained network structure.

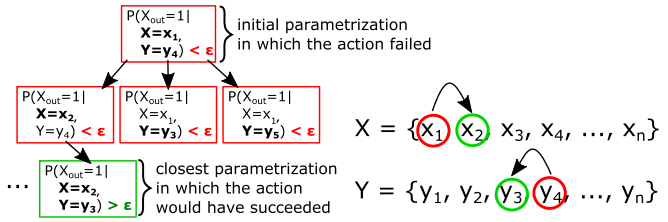


Fig. 2. Exemplifies how contrastive explanations are generated from the BFS search tree.

C. Our Proposed Method to Explain Failures

Our proposed method to generate contrastive failure explanations uses the obtained causal Bayesian network to compute success predictions and is summarized in algorithm 1. In (L-2 Algorithm 1)), a matrix is generated which defines transitions for every single-variable change for all possible variable parametrizations. For example, if we had two variables X_1, X_2 with two intervals x_1, x_2 each. Then, the possible valid transitions for $node = (X_1 = x_1, X_2 = x_1)$ would be $child_1 = (X_1 = x_1, X_2 = x_2)$ or $child_2 = (X_1 = x_2, X_2 = x_1)$. Lines 5-15 (Algorithm 1) describe the adapted BFS procedure, which searches for the closest variable parametrization that fulfills the goal criteria of $P(d \in X_{\text{goal}} | \Pi_{child}) > \epsilon$, where ϵ is the success threshold, which can be heuristically set. The concept of our proposed method is to generate contrastive explanations that compare the current variable parametrization associated with the execution failure $x_{\text{current}_{\text{int}}}$ with the closest parametrization that would have allowed for a succesfull task execution $x_{\text{solution}_{\text{int}}}$. Consider Fig. 2 for a visualization of the explanation generation, exemplified on two variables X and Y, which are both causally influencing the variable X_{out}. Furthermore, it is known that $x_{\text{out}} = 1 \in X_{\text{goal}}$. The resulting explanation would be that the task failed because $X = x_1$ instead of $X = x_2$ and $Y = y_4$ instead of $Y = y_3$.

IV. Experiments

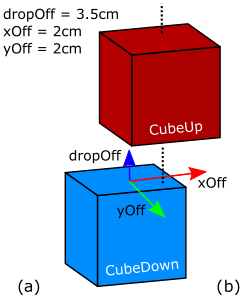
We evaluate our method to find causal explanations of failures based on two different scenarios. The goal of *experiment 1* is to

Algorithm 1: Failure Explanation.

Input: failure variable parameterization x_{failure}, graphical model $\mathcal{G}$, structural equations $P(X_i|\Pi_{X_i})$, discretization intervals of all model variables X_{int}, success threshold ϵ, goal parametrizations X_{goal}

Output: solution variable parameterization $x_{\text{solution}_{\text{int}}}$, solution success probability prediction p_{solution}

$x_{\text{current}_{\text{int}}} \leftarrow$ GETINTERVALFROMVALUES$(x_{\text{failure}}, X_{\text{int}})$
$P \leftarrow$ GENERATETRANSITIONMATRIX(X_{int})
$q \leftarrow [x_{\text{current}_{\text{int}}}]$
$v \leftarrow []$
while $q \neq \emptyset$ **do**
  $node \leftarrow$ POP(q)
  $v \leftarrow$ APPEND$(v, node)$
  for each transition $t \in P(node)$ **do**
    $child \leftarrow$ CHILD$(P, node)$
    if $child \notin q, v$ **then**
      $p_{\text{solution}} = P(d \in X_{\text{goal}}|\Pi_{child})$
      if $p_{\text{solution}} > \epsilon$ **then**
        $x_{\text{solution}_{\text{int}}} \leftarrow child$
        RETURN$(p_{\text{solution}}, x_{\text{solution}_{\text{int}}})$
      $q \leftarrow$ APPEND$(q, x_{\text{current}_{\text{int}}})$



Variable	Meaning
`xOff`	cubeUp.x − cubeDown.x (distance between cube centers)
`yOff`	cubeUp.y − cubeDown.y
`dropOff`	(cubeUp.z − cubeDown.z) −cubeLength (distance between cube surfaces)
`colorUp`	color of CubeUp
`colorDown`	color of CubeDown
`onTop`	denotes if CubeUp is on top of CubeDown after the stacking process

Fig. 3. (a) Visualises the used variables $\mathbf{X}$ in experiment 1. (b) Describes their meaning.

stack one cube on top of another. The goal of *experiment 2* is to drop a sphere into different containers.

A. Experiment 1: Stacking Cubes

In the cube stacking scenario, the environment contains two cubes: *CubeUp* and *CubeDown* (see Fig. 3). The goal of the stacking action is to place CubeUp *on top* of CubeDown. We define six variables as follows: $\mathbf{X} = \{\texttt{xOff}, \texttt{yOff}, \texttt{dropOff}, \texttt{colorDown}, \texttt{colorUp}, \texttt{onTop}\}$. Both cubes have an edge length of 5 cm.

1) Cube Stacking Simulation Setup: We run the simulations in Unity3d, which bases its physics behavior on the Nvidia PhysX engine. For training the Bayesian network, we generate 20,000 samples on 500 parallel table environments (see Fig. 1). We randomly sample values for $\texttt{xOff}, \texttt{yOff} \sim \mathcal{U}_{[-3,3]}$ (in cm), $\texttt{dropOff} \sim \mathcal{U}_{[0.4,10]}$ (in cm), $\texttt{colorUp}, \texttt{colorDown} = \{\text{Red}, \text{Blue}, \text{Green}, \text{Orange}\}$. $\texttt{onTop} = \{\text{True}, \text{False}\}$ is not sampled but automatically determined after the stacking process.

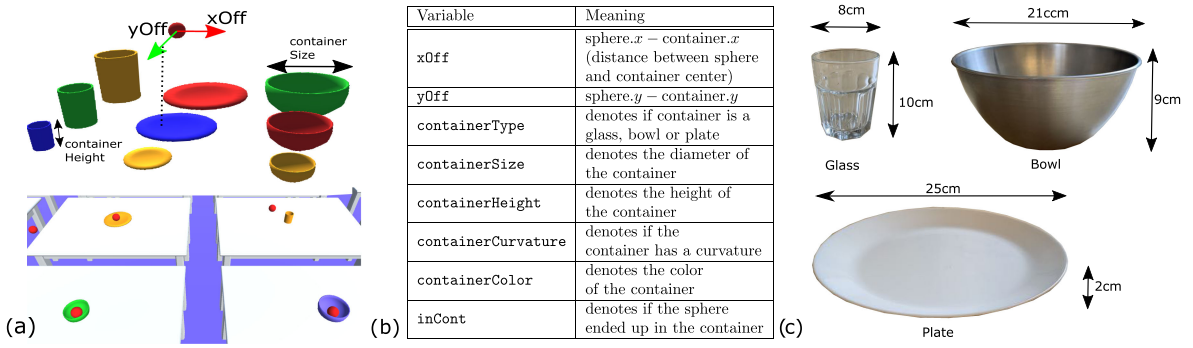
2) Cube Stacking Robot Experiments Setup: We run and assess our experiments on two different robotic platforms (Fig. 1): the TIAGo service robot with one arm and parallel gripper and the UR3 with a piCOBOT suction gripper. The real cubes are 3D printed out of PLA (polylactic acid) and weigh around 25 grams each. For each robot, we run 180 stacking trials. Instead of randomly sampling values for the variables, as we do for training the causal model, we evaluate the real-world behavior at 36 different points (all possible combinations of $\texttt{xOff}, \texttt{yOff} = \{0, 1, 2\}$ (in cm) and $\texttt{dropOff} = \{0.5, 2, 3.5, 5\}$ (in cm)). These 36 points were empirically chosen because they cover an area where the ideal conditions of the simulation (e.g., collisions without any rotation due to the gripper motors) could potentially have the most significant effect on behavior discrepancies. Once the upper cube is too far outside (> 2.5 cm), it doesn't play a role how it was dropped, so we have not included points with larger offsets than 2 cm. For each unique stacking setup instantiation, we conduct five iterations. After each trial, the cubes are re-adjusted into an always similar pre-stack position by the operator. The stacking outcome (`onTop` value) was also determined by the operator. Note that the purpose of the robot experiments is not to modify the causal model that we learned from the simulation but to evaluate the model transferability to the real environment.

B. Experiment 2: Dropping Spheres Into Containers

In our second experiment, the robot needs to drop spheres into different containers. The environment contains a *Sphere* and one of several possible *Containers*, which are shaped like a plate, bowl or glass (see Fig. 4). We define eight variables as follows: $\mathbf{X} = \{\texttt{xOff}, \texttt{yOff}, \texttt{inCont}, \texttt{contHeight}, \texttt{contSize}, \texttt{contType}, \texttt{contCurvature}, \texttt{contColor}\}$. This new list of variables is required because experiment 2 contains a different set of objects and new randomization parameters like the size, which we did not consider in experiment 1. The sphere has a diameter of 6.6 cm and the size of the containers is randomized. We chose a constant dropping height of 0.4 m.

1) Sphere Dropping Simulation Setup: For training the Bayesian network, we generate 800,000 samples, on 400 parallel table environments [see Fig. 4(a)], in Unity3d. We randomly sample values for $\texttt{xOff}, \texttt{yOff} \sim \mathcal{U}_{[-6,6]}$ (in cm), $\texttt{contColor} = \{\text{Red}, \text{Blue}, \text{Green}, \text{Orange}\}$, $\texttt{contType} = \{\text{Glass}, \text{Plate}, \text{Bowl}\}$. We do not directly set `contHeight` and `contSize`, but manipulate these variables via a $\texttt{scalingFactor} \sim \mathcal{U}_{[0.4,1]}$. In Unity3d, the scaling factor can be utilized to manipulate the size of objects. We use the same scaling factor in all three dimensions, thus the height (`contHeight`) to diameter (`contSize`) ratio was constant for each object type respectively. $\texttt{inCont} = \{\text{True}, \text{False}\}$ is not sampled but automatically determined after the stacking process.

2) Sphere Dropping Robot Experiment Setup: We run and assess the sphere dropping experiment on the TIAGo service robot (as introduced in Section IV-A2). As a sphere, we use a regular tennis ball, which weighs around 58 g and has a diameter of around 6.6 cm. We chose three containers which each possess a height and size parametrization that was captured in the model [see Fig. 4(c)]. We evaluate each container at nine different points (all possible combinations of $\texttt{xOff}, \texttt{yOff} = \{0, 3, 6\}$ (in cm)). For each unique sphere dropping setup instantiation, we conduct five iterations. Similar to the prior experiment, the containers are re-adjusted by a human operator, who is also determining the dropping outcome.



Variable	Meaning
xOff	sphere.x − container.x (distance between sphere and container center)
yOff	sphere.y − container.y
containerType	denotes if container is a glass, bowl or plate
containerSize	denotes the diameter of the container
containerHeight	denotes the height of the container
containerCurvature	denotes if the container has a curvature
containerColor	denotes the color of the container
inCont	denotes if the sphere ended up in the container

Fig. 4. (a) Visualises some of the variables $\mathbf{X}$ from experiment 2 and the simulation setup. (b) Describes the variable meanings. (c) The containers that were used for the real-world experiment.

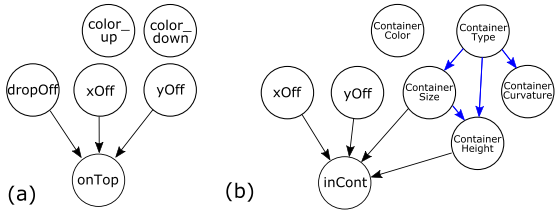


Fig. 5. Obtained BN for (a) the cube stacking and (b) the sphere dropping experiment. The blue edges have been detected by the structure learning algorithm but had to be directed manually.

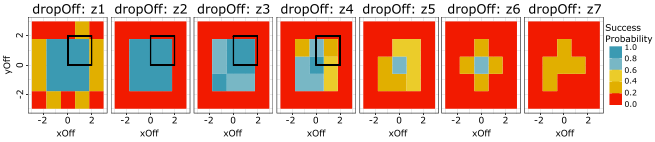


Fig. 6. Visualisation of the conditional probability table for $P(\texttt{onTop} = 1|\Pi_{\texttt{onTop}})$. xOff, yOff are discretized into 5 intervals and dropOff into 7. Values are in cm. The black rectangles denote the xOff and yOff value range evaluated in the real-world experiments.

V. Results and Discussion

A. Analysis of the Obtained Causal Models

We first present and discuss the learned causal model of the cube stacking scenario. 10-fold cross-validation reports an average loss of 0.10269 and a standard deviation of 0.00031, and Fig. 5(a) displays the resulting DAG. The graph indicates, that there are causal relations from xOff, yOff and dropOff to onTop, while the two color variables colorDown and colorUp are independent. In other words, it makes a difference from which position the cube is dropped, but the cube color has no impact on the stacking success. We obtained the following dropOff (z) intervals (in cm):

$$z_1 : [0.4, 1.8],\ z_2 : (1.8, 3.2],\ z_3 : (3.2, 4.5],\ z_4 : (4.5, 5.9],$$
$$z_5 : (5.9, 7.3],\ z_6 : (7.3, 8.6],\ z_7 : (8.6, 9.9]$$

and the following xOff/yOff intervals (in cm):

$$x/y_1 : [-3, -1.8],\ x/y_2 : (-1.8, -0.6],\ x/y_3 : (-.6, .6],$$
$$x/y_4 : (.6, 1.8],\ x/y_5 : (1.8, 3]$$

The conditional probabilities $P(\texttt{onTop} = 1|\Pi_{\texttt{onTop}})$ are visualized in Fig. 6. These plots allow us to conclude that stacking success decreases the greater the drop-offset and the more offset in both x- and y-direction. In particular, there is a diminishing chance of stacking success for the values $|\text{xOff}| > 1.8$ or $|\text{yOff}| > 1.8$, no matter the dropOff height.

The obtained DAG for the sphere dropping experiment is visualized in Fig. 5(b). The algorithm detected causal links between contHeight contSize, contType, contCurvature (marked in blue), but, initially, was not able to direct these four edges (which is a common problem in structure learning as discussed in Section III-B1). Since it is not possible to fit a conditional probability table based on an undirected graph, we directed these edges manually based on our domain knowledge of the task. 10-fold cross validation reports an average loss of 0.184 and a standard deviation of 0.000052. The graph indicates causal links between contType and inCont via contHeight and contSize, while contColor and contCurvature do not impact the dropping success.

We obtained the following discretization intervals for the four variables that affect inCont (all values are in cm). We obtained the following contSize (s) intervals (in cm):

$$s_1 : [6.3, 11.2],\ s_2 : (11.2, 14],\ s_3 : (14, 17],\ s_4 : (17, 22],$$
$$s_5 : (22, 30],$$

the following contHeight (h) intervals (in cm):

$$h_1 : [1.5, 2.9],\ h_2 : (2.9, 6.3],\ h_3 : (6.3, 9.5],\ h_4 : (9.5, 12.3],$$
$$h_5 : (12.3, 19],$$

and the following xOff/yOff intervals (in cm):

$$x/y_1 : [-6, -3.6],\ x/y_2 : (-3.6, -1.2],\ x/y_3 : (-1.2, 1.2],$$
$$x/y_4 : (1.2, 3.6],\ x/y_5 : (3.6, 6]$$

Querying the causal model for sphere dropping success given the object type reveals that the task is most likely successful for bowls (69%) followed by plates (59%) and glasses (25%). The model also indicates that larger-sized containers are more tolerant to x/y-offsets and yield a higher chance of dropping success. (Fig. 7). Surprisingly, a similar trend cannot be determined with the container height (Fig. 7). The reasons are twofold: First, the container height depends not only on the size but also on the type of container (e.g., glasses typically have a larger height than plates). Consequently, not all object types are represented in all height intervals, e.g., plates are only covered in h1 and h2, whereas cups are distributed among h2-h4. Second, cups and

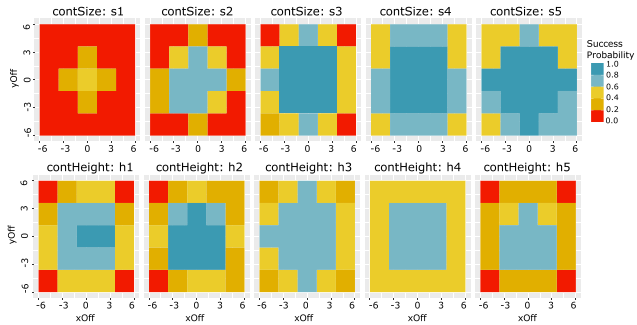


Fig. 7. Visualisation of the conditional probability table for $P(\texttt{inCont} = 1|\Pi_{\texttt{inCont}})$. All values are in cm.

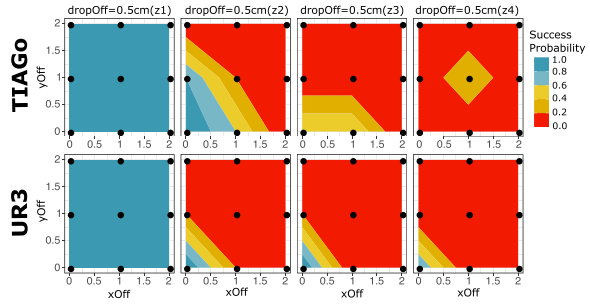


Fig. 8. Shows real-world success probabilities for the task of stacking cubes, evaluated for two robots (TIAGo and UR3) at 36 different points (black dots). The probabilities are interpolated between the nine measurement points for each `dropOff` value.

plates were found to contribute to a higher dropping success chance. As a result, the largest success chances can be obtained in interval h2.

Overall, we conclude that the obtained success probabilities resemble our intuitive understanding of the physical processes for both scenarios. Nevertheless, real-world experiments have a higher complexity due to the many environment uncertainties. We, therefore, expect the simulation to be less conservative than reality, as we have higher control over the variables involved in the stacking process.

B. *sim2real Accuracy of the Causal Models*

To evaluate how well the causal model and the real-world match, we introduce the sim2real accuracy score. It is defined as the normalized difference in predicted probabilities over the set of points that were evaluated in real-world experiments.[1]

The results for the real-world experiments of the cube stacking scenario are presented in Fig. 8, where the black points indicate the nine stacking locations (all possible combinations of x- and y-offset values) for each of the four drop-off heights. The plots show the contours of the probabilities, meaning the stacking success probabilities are interpolated between the nine measurement points. The sim2real accuracy amounts to 71% for the TIAGo and 69% for the UR3. The largest discrepancy between model and reality can be determined for the higher drop-off positions. For the real-world measurements, the stacking success drops earlier, at around 2 cm or 3.5 cm. It is also interesting to compare similarities regarding probability outcomes between the two differently embodied robots. The similarity with respect to the 36 measured positions amounts to 85%.

The results for the sphere dropping experiments are presented in Fig. 9. We obtained 72% sim2real accuracy for the tested data points. We observed the most significant discrepancy for

[1]A visual demonstration of our experiments can be found under https://youtu.be/rl_zDfUZ2Kk.

TABLE I
THREE EXAMPLES OF FAILURE EXPLANATIONS FOR BOTH SCENARIOS OF CUBE STACKING AND SPHERE DROPPING. INTERVALS THAT WERE SUBJECT TO CHANGES IN THE CLOSEST SOLUTION ARE MARKED IN BOLD LETTERS

input	input interval	curr. succ. probability	closest solution interval	exp. succ. probability
Cube Stacking - Example 1:				
xOff = 0.0 yOff = 0.02 dropOff = 0.08	x_3 $\boldsymbol{y_5}$ $\boldsymbol{z_6}$	0.0	x_3 $\boldsymbol{y_4}$ $\boldsymbol{z_3}$	0.85
Explanation: The upper cube was stacked too high and too far to the front of the lower cube.				
Cube Stacking - Example 2:				
xOff = 0.015 yOff = 0.0 dropOff = 0.05	x_4 y_3 $\boldsymbol{z_4}$	0.58	x_4 y_3 $\boldsymbol{z_3}$	0.91
Explanation: The upper cube was stacked too high.				
Cube Stacking - Example 3:				
xOff = -0.015 yOff = 0.015 dropOff = 0.02	$\boldsymbol{x_1}$ $\boldsymbol{y_1}$ z_2	0.017	$\boldsymbol{x_2}$ $\boldsymbol{y_2}$ z_2	1.0
Explanation: The upper cube was stacked too far to the left and too far to the back of the lower cube.				
Sphere Dropping - Example 1:				
xOff = 0.015 yOff = 0.015 ContSize = .08 ContHeight = .1	x_4 y_4 $\boldsymbol{s_1}$ $\boldsymbol{h_4}$	0.189	x_4 y_4 $\boldsymbol{s_3}$ $\boldsymbol{h_3}$	0.957
Explanation: The container was too small and too high (interpretation: take a bowl instead of a cup, which is bigger but has less height)				
Sphere Dropping - Example 2:				
xOff = 0.059 yOff = 0.059 ContSize = .21 ContHeight = .09	$\boldsymbol{x_5}$ y_5 s_4 h_3	0.727	$\boldsymbol{x_4}$ y_5 s_4 h_3	0.98
Explanation: The sphere was dropped too far to the right.				
Sphere Dropping - Example 3:				
xOff = -0.03 yOff = -0.03 ContSize = .15 ContHeight = .017	x_2 y_2 $\boldsymbol{s_3}$ h_1	0.58	x_2 y_2 $\boldsymbol{s_4}$ h_1	0.933
Explanation: The container (plate) was too small.				

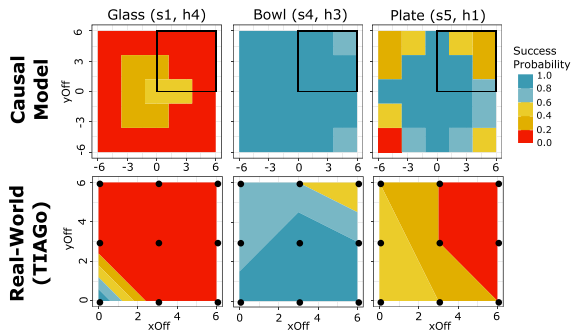


Fig. 9. Compares success probabilities for the task of sphere dropping between the real experiments (evaluated at 9 different points marked as black dots) and the causal model. The black rectangles denote the `xOff/yOff` value range (in cm) evaluated in the robot experiments.

TABLE II
COMPARISON OF OUR EXPLANATION GENERATION PIPELINE WITH OTHER APPROACHES

	Method	Output	Detect. of caus. relevant variables	Learning Prerequisites	Task Succ. Predict.
CB-H [19]	Fault trees + encoder-decoder network	language model (spoken failure expl.)	no differentiation between causally relevant and irrelevant variables	failure-cause annotated simulations	no
SSG-R [20]	MOTIFNET [21] + pairwise ranking	list of relevant spatial/object relations	informally, through pairwise ranking	relationship ranking labels	no
ours	causal BNs + contrastive BFS	contrastive failure explanation	formally, through BN structure learning	task simulations (incl. action outcome)	MLE (or similar like Bayesian est.)

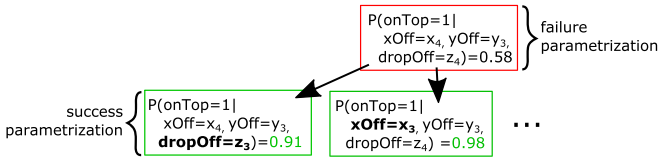


Fig. 10. BFS for explaining Cube Stacking - Example 2 from Table I.

the plate, which was predicted to have a much larger success probability by the model. One reason could have been, for example, that the surface of the real plate was not perfectly flat as in the simulations.

We can conclude that for both tested scenarios, the probability model obtained from simulated data matches reasonably well with reality and thus can be utilized for the explanation of failures that occur in the real world. Furthermore, the model generalizes well to differently embodied robots. We want to emphasize that the causal model was not retrained or adapted when the real scenarios were tested. If we had obtained a lower sim2real accuracy or more significant differences between the two robots, it would be advisable to include robot-specific variables (such as the gripper type and orientation) and adapt the model with real-world data. But even then, the model that we obtain from the simulation can be used as an excellent experience prior, allowing for faster applicability and learning.

C. Explainability Capabilities

Finally, we provide several concrete examples to showcase how our method finds explanations for robot failures. We set the probability threshold that distinguishes a failure from success to $\epsilon = 0.8$ for all examples. Table I provides three examples for both scenarios of stacking a cube and dropping a sphere. Cube Stacking - Example 2 is particularly interesting, as it showcases that there are often multiple correct explanations for the error. In this case it would have been possible to achieve a successful stacking by either going from $\texttt{dropOff} = z_4$ to $\texttt{dropOff} = z_3$ or by changing $\texttt{xOff} = z_4$ to $\texttt{xOff} = z_3$ (the search tree is visualised in Fig. 10). Which solution is found first, depends on the variable prioritization within the tree search due to the BFS principle.

Our closest solution will lead to a minimal number of interval changes and thus provides the simplest solution in terms of Occam's razor principle [1]. For instance, in Sphere Dropping - Example 2, it would have also been possible to change the container to a larger bowl. But instead, the search process found it was easier to adapt the xOff position. The advantage of the current uninformed BFS is that this principle is always applicable and does not require any human domain knowledge.

D. Comparison of Our Failure Explanation Approach With Baseline Methods

We compare our method of finding explanations of robot task failures with the two closely related methods of Context-Based History (CB-H) [19] explanations, and the ranked Semantic Scene Graph method (SSG-R) [20], based on the criteria that are summarized in Table II. For CB-H all failures and their causes need to be manually defined in the form of Fault Trees. In SSG-R failures are not modeled, but explained in form of a list of spatial relations (like *close to* or *occluded*) and object features (like *fragile* or *heavy*), automatically detected through the semantic scene graph model MOTIFNET [21]. We explain failures via contrastive variable parametrizations. Due to these differences in failure representation, all three methods have different requirements during the learning phase. For learning the encoder-decoder network that generates language failure explanations for CB-H, simulations must be annotated with the respective failure cause. In [19], 2100 annotated time-steps were used to train for six different failure causes. However, the number of required samples will drastically increase for the two discussed examples of cube stacking and sphere dropping due to the increased number of failure possibilities. Additionally, samples are more expensive than in our method since it is required to label the failure cause instead of a simple binary action success label. In SSG-R, pairwise ranking distinguishes between relevant and irrelevant relations. Pairwise relation preferences must be provided via domain knowledge of the failure scenario and which are more expensive than the automatically retrievable binary action success labels from our method. Another difficulty in terms of applicability to the presented scenarios of cube stacking and sphere dropping provide the continuous variables (e.g., contSize or xOff), which are discretized into more than two categories (as opposed to binary object relations). For these variables, MOTFNET is not applicable. While, in principle, a range of variables was detected to influence the action outcome causally, it is due to a specific variable parametrization that they lead to the action failure. Our method automatically discerns between relevant and irrelevant relations. Last but not least, neither CB-H nor SSG-R learn an action success model, which can be useful for other tasks beyond failure explanation, e.g., failure prediction and prevention.

To conclude, both CB-H and SSG-R would require significant changes and adaption to find explanations for the experiment scenarios discussed in this letter. One of the most significant differences of both methods with ours is the requirement of failure-cause labels instead of action success labels, which are typically easier to obtain.

VI. CONCLUSION

This letter presents our novel approach to finding causal explanations for robot failures. First, we learn a causal Bayesian

network from simulated task executions. We show that the model is transferable to the real world with 70% and 72% accuracy over two tasks of stacking cubes and dropping a sphere into different containers and is agnostic to differently embodied robots. Furthermore, we propose a new method to generate explanations of execution failures based on the causal model. This method finds a contrastive explanation comparing the action parametrization of the failure with its closest parametrization that would have led to a successful execution, which is found through breadth-first search (BFS). For future work, we would like to incorporate a language model that automatically encodes the contrastive failure explanations into a vector of words, such that it can be communicated more intuitively to a wide range of potential users. Furthermore, we want to investigate how the obtained causal models can also be used to predict and prevent failures from happening.

REFERENCES

[1] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, 2019.

[2] T. Hellström, "The relevance of causation in robotics: A review, categorization, and analysis," *Paladyn, J. Behav. Robot.*, vol. 12, no. 1, pp. 1–2, 2021.

[3] J. Pearl and D. Mackenzie, *The Book of Why: The New Science of Cause and Effect*. New York, NY, USA: Basic Books, Inc., 2018.

[4] A. Mitrevski, P. G. Plöger, and G. Lakemeyer, "Representation and experience-based learning of explainable models for robot action execution," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5641–5647.

[5] K. Lee, Y. Su, T.-K. Kim, and Y. Demiris, "A syntactic approach to robot imitation learning using probabilistic activity grammars," *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1323–1334, 2013.

[6] S. Karapinar and S. Sariel, "Cognitive robots learning failure contexts through real-world experimentation," *Auton. Robots*, vol. 36, no. 4, pp. 469–485, 2015.

[7] D. Lewis, "Causal explanation," in *Philosophical Papers* vol. II, New York, USA: Oxford University Press, 1986, pp. 214–240.

[8] A. S. Bauer, P. Schmaus, F. Stulp, and D. Leidner, "Probabilistic effect prediction through semantic augmentation and physical simulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 9278–9284.

[9] J. Brawer, M. Qin, and B. Scassellati, "A causal approach to tool affordance learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 8394–8399.

[10] K. C. Stocking, A. Gopnik, and C. Tomlin, "From robot learning to robot understanding: Leveraging causal graphical models for robotics," in *Proc. 5th Conf. Robot Learn.*, 2022, vol. 164, pp. 1776–1781.

[11] T. E. Lee, J. A. Zhao, A. S. Sawhney, S. Girdhar, and O. Kroemer, "Causal reasoning in simulation for structure and transfer learning of robot manipulation policies," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 4776–4782.

[12] A. A. Bhat, V. Mohan, G. Sandini, and P. G. Morasso, "Humanoid infers archimedes' principle: Understanding physical relations and object affordances through cumulative learning experiences," *J. Roy. Soc. Interface*, vol. 13, no. 120, 2016, Art. no. 20160310.

[13] D. Song, K. Huebner, V. Kyrki, and D. Kragic, "Learning task constraints for robot grasping using graphical models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 1579–1585.

[14] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 8081–8086.

[15] M. Diehl, C. Paxton, and K. Ramirez-Amaro, "Automated generation of robotic planning domains from observations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 6732–6738.

[16] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "The emerging landscape of explainable automated planning & decision making," in *Proc. Int. Joint Conf. Artif. Intell.*, 2020, pp. 4803–4811.

[17] B. Seegebarth, F. Müller, B. Schattenberg, and S. Biundo, "Making hybrid plans more clear to human users — a formal approach for generating sound explanations," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2012, pp. 225–233.

[18] V. Raman and H. Kress-Gazit, "Explaining impossible high-level robot behaviors," *IEEE Trans. Robot.*, vol. 29, no. 1, pp. 94–104, Feb. 2013.

[19] D. Das, S. Banerjee, and S. Chernova, "Explainable AI for robot failures: Generating explanations that improve user assistance in fault recovery," in *Proc. IEEE/ACM Int. Conf. Hum.-Robot Interaction*, 2021, pp. 351–360.

[20] D. Das and S. Chernova, "Semantic-based explainable AI: Leveraging semantic scene graphs and pairwise ranking to explain robot failures," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 3034–3041.

[21] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural motifs: Scene graph parsing with global context," in *IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5831–5840.

[22] M. J. Vowels, N. C. Camgoz, and R. Bowden, "D'ya like dags? A survey on structure learning and causal discovery," *Assoc. Comput. Machinery Comput. Surv.*, 2021.

[23] Y.-C. Chen, T. A. Wheeler, and M. J. Kochenderfer, "Learning discrete bayesian networks from continuous data," *J. Artif. Intell. Res.*, vol. 59, no. 1, pp. 103–132, 2017.

[24] M. Scutari, "Learning bayesian networks with the bnlearn R package," *J. Stat. Softw.*, vol. 35, no. 3, pp. 1–22, 2010.

[25] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, "Dags with no tears: Continuous optimization for structure learning," *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 1–2, 2018.

[26] D. Margaritis, "Learning bayesian network model structure from data," Ph.D. dissertation, Pittsburgh School Comput. Sci., Carnegie Mellon Univ., 2003.

[27] R. Nagarajan and M. Scutari, *Bayesian Networks in R With Applications in Systems Biology*. Berlin, Germany: Springer-Verlag, 2013.

[28] A. Sharma, V. Syrgkanis, C. Zhang, and E. Kiciman, "DoWhy: Addressing challenges in expressing and validating causal assumptions," *ICMAL Workshop: The Neglected Assumptions In Causal Inference*, 2021.

[29] J. Peters, D. Janzing, and B. Schlkopf, *Elements of Causal Inference: Foundations and Learning Algorithms*. Cambridge, MA, USA: MIT Press, 2017.