2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Representation and Experience-Based Learning of Explainable Models for Robot Action Execution

Alex Mitrevski[†§], Paul G. Plöger[†], and Gerhard Lakemeyer[‡]

***Abstract*— For robots acting in human-centered environments, the ability to improve based on experience is essential for reliable and adaptive operation; however, particularly in the context of robot failure analysis, experience-based improvement is practically useful only if robots are also able to reason about and explain the decisions they make during execution. In this paper, we describe and analyse a representation of execution-specific knowledge that combines (i) a *relational model* in the form of qualitative attributes that describe the conditions under which actions can be executed successfully and (ii) a *continuous model* in the form of a Gaussian process that can be used for *generating* parameters for action execution, but also for *evaluating* the expected execution success given a particular action parameterisation. The proposed representation is based on prior, modelled knowledge about actions and is combined with a learning process that is supervised by a teacher. We analyse the benefits of this representation in the context of two actions - grasping handles and pulling an object on a table - such that the experiments demonstrate that the joint relational-continuous model allows a robot to improve its execution based on experience, while reducing the severity of failures experienced during execution.**

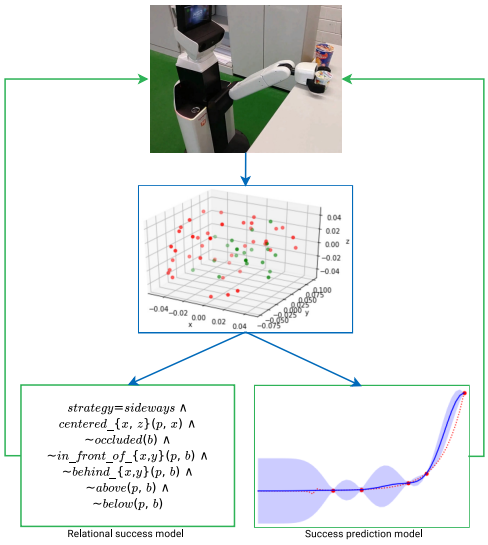


Fig. 1: An overview of our experience-based framework for representing robot actions

I. INTRODUCTION

For robots that act in human-centered environments, identifying the reasons for execution failures - in other words, performing diagnosis - is an important aspect, as such robots need the ability to explain the decisions they make and correct their behaviour accordingly. Both explanation and correction require a suitable knowledge representation framework, which a robot can use to reason about its execution and that can be improved based on the robot's own experiences [1], [2]. One way to encode both properties is through a combination of relational and continuous models.

A relational model of execution can serve multiple purposes. Such a model could simplify the task of analysing a robot's behaviour for robot developers and maintainers, since the decisions of the robot are represented in an explicit manner; in other words, a relational model of execution can be used to explain why a robot performs actions in a certain way, for instance why an object was grasped in a given manner.[1] Relational success models could also supplement automated failure diagnosis; as in consistency-based diagnosis, failures can be examined as violations of the model [4], [5]. A continuous model is also useful in different contexts. Such a model can be used to parameterise actions so that the likelihood of success is reasonably large. Similarly, an appropriate formalisation can provide means of predicting the likelihood of success if a robot commits to a certain parameterisation, as demonstrated in [6], [7]. Finally, a continuous representation can be improved based on robot experience using a number of optimisation procedures.

This paper improves upon our work in [8], where we introduced a data-driven execution model for object placing. In this paper, we address the problem of learning general experience-driven robot action models, such that we modify the representation in [8] so that the model applies to different types of actions that may need to be executed in various *qualitative contexts*, with a particular focus on manipulation actions. The representation we present here combines a relational model[2] extracted from a set of action-specific qualitative relations with a predictive success function in the form of a Gaussian process, such that both representations are learned from direct robot experiences which are evaluated by a human teacher. The general framework we present here is illustrated in Fig. 1.

We demonstrate our framework by letting a Toyota Human

*This work was supported by the B-IT foundation

[†]Alex Mitrevski and Paul G. Plöger are with the Department of Computer Science, Hochschule Bonn-Rhein-Sieg, Sankt Augustin, Germany <aleksandar.mitrevski, paul.ploeger>@h-brs.de

[‡]Gerhard Lakemeyer is with the Department of Computer Science, RWTH Aachen, Aachen, Germany gerhard@informatik.rwth-aachen.de

[§]Corresponding author

[1]It should however be noted that there are alternative models of explanation, such as the one in [3].

[2]Or multiple such models in the case of multiple qualitative contexts.

Support Robot (HSR) [9] learn two actions: grasping handles and pulling an object to a goal region.[3] For both cases, we present relations that we use for extracting execution rules and show that the learned representation improves the execution success of the robot, but also reduces the severity of failures experienced by the robot during execution.[4]

II. RELATED WORK

Action modelling and execution are often considered separately in the literature, although there have been efforts that have looked at the two in conjunction. Kaelbling and Lozano-Perez [7] consider the problem of learning the preconditions of action models and address the question of improving those models based on the individual experiences of a robot. The primary objective here is learning a function that maps action parameters, such as the distance to push an object or the direction to push from, to predicted execution success values. During execution, this mapping is used in an inverse manner, namely the goal is to find parameters that guarantee some desired effect, such as the position of an object after pushing. In [10], Karapinar and Sariel discuss an application of inductive logic programming, where the objective is to discover execution failure rules that explain a set of training examples; these rules are then used to define a failure context, which is added to the preconditions of an appropriate planning operator. Rules here are represented using predicates for qualitative object aspects (colour, shape, category) as well as through numerical attributes, such as the location of an object in the world frame. Stulp et al. [11] introduce a model that relates the position of a robot to the positions of objects that it needs to manipulate, such that the model represents the probability that a particular manipulation action, such as grasping an object, would succeed from a given - potentially uncertain - position of the robot. As in [7] and [10], the model is learned from experience; in particular, success models are first learned in various robot-object configurations and are then combined into a generalisable representation.

Our work builds on these representations in various respects. As in [7], we aim for a model that predicts whether a particular set of action parameters leads to execution success and can also be used for sampling parameters where success is likely; however, as in [12], we use a Gaussian process as a predictive model in order to encode uncertainty about the predictions, and additionally learn a symbolic representation of the preconditions. The latter aspect is much like in [10], but instead of learning a symbolic representation of when an action fails, we learn a model of when the action succeeds, as learning all possible failure modes may be an intractable problem. As in [11], we use a probabilistic model of success, but represent actions with perception in the loop in order to take into account the relative positioning of objects with respect to the robot and with respect to each other.

[3]The second use case is a variation of the pushing scenario in Kopicki et al. [6] and Kaelbling and Lozano-Perez [7].

[4]An accompanying video for this paper can be found at `https://youtu.be/crPxLQ2d0MY`

Direct learning from experience is also typically done in the context of reinforcement learning [13], [14], where an agent collects experiences of interacting with the environment and modifies its behaviour in order to maximise a reward signal. Policies learned by reinforcement can be versatile, but may also be difficult to interpret and are not necessarily suitable for automated reasoning, both properties being important for diagnosing failures and grounded experiential improvement. This problem is addressed in [15], where Höfer and Brock present an algorithm for learning action models in both symbolic and continuous space in an expectation-maximisation fashion, where the symbolic model is used as a forward model for predicting action effects and creating execution plans, while the continuous model is used for selecting optimal execution parameters, such as the direction of pulling when opening a door. On the level of robot motions, Orthey et al. [16] study the problem of finding motion primitive parameters that increase the likelihood that some desired symbolic effects are achieved by the motion; the parameter optimisation here is thus driven by the desired effects. It is worth mentioning that in both of these cases, noisy deictic rules are used due to their ability to model uncertain effects. In our work, we use conventional first-order logic and qualitative relations, and model uncertainty through the continuous representation over the execution parameters.

III. REPRESENTING AND LEARNING EXPLAINABLE ACTION EXECUTION MODELS

Our objective in this paper is to develop a generic representation of the preconditions for executing a particular action so that the action's desired effects are achieved, such that the representation should satisfy several constraints: (i) it should be possible to learn the preconditions by experience (either real or simulated), (ii) during the learning process, modelled knowledge should be readily usable, (iii) the representation should be predictive in the sense that it allows to assess the expected outcome of the execution given a set of execution parameters, but also generative so that a robot can sample execution parameters that would increase its chances of being successful. Our representation aims to satisfy these constraints by combining a relational model of execution with a continuous, predictive counterpart.

A. Notation

Given a predicate P encoding a spatial relation, we use $P_{x,y,z}$ to denote three different versions of P, one for each of the three coordinate axes. Given the bounding box B of an object, we use $\min(B_x)$ and $\max(B_x)$ to denote the minimum and maximum coordinate along a given coordinate axis x, and $\overline{B_x}$ for representing the mean coordinate.

B. Action Representation

Our action model is defined in terms of relations based on which different *qualitative* descriptions of the execution context of an action can be made. This allows actions to exist in different qualitative modes.[5]

[5]For instance, for our object pulling action, we can represent the distance of the object to the robot qualitatively as *short* and *long*.

Definition 1: Given a relation Q with m possible values, the execution of an action A given a value $Q_i, 1 \leq i \leq m$ is *execution under a qualitative mode* of A.

We model actions using the notion of an execution model, which is defined below and is a generalisation of the δ model presented in [8].

Definition 2: An *execution model* of an action A is a tuple $M = (R = \{R_1, ..., R_m\}, F)$, where each $R_i, 1 \leq i \leq m$ is a set of preconditions for the action to be executed successfully, one for each qualitative mode, and $F : \mathbb{R}^w \rightarrow [0, 1]$ is a function on action parameters and constraints $\mathbf{x} \in \mathbb{R}^w$, encoding a mapping between $\mathbf{x}$ and the expected execution success.

This representation is quite generic and allows combining a collection of qualitative models with a single continuous model; this can be thought of as a form of multitask learning, which has been shown to improve generalisation [17]. It should be noted that Q defines the level of desired modelling granularity for an action; a coarse model may have a single qualitative mode, while a more detailed model could represent multiple execution contexts through various qualitative modes. As in [8], an execution model contains two distinct representations of virtually the same knowledge since they serve different purposes: R can be used for high-level explanation and reasoning about the execution, while F is used for grounding the execution to the physical world. Unlike in the case of [8], however, this definition is not tied to a particular action.

In our action model, we distinguish between two types of actions: (i) actions with a constrained objective and (ii) free, unconstrained actions. The object pulling action we consider in this paper has a constrained objective since the object needs to end up in a predefined region; the handle grasping action is, on the other hand, free since there is no constraint on how the handle is grasped. The type of action influences the input $\mathbf{x}$ to F: the input $\mathbf{x}$ to a constrained action includes the constraints in addition to any other action parameters.

C. Action Learning

As an execution model M is composed of a symbolic and a continuous part, the problem of learning M is twofold: we need relational learning for the preconditions R and function approximation for the success model F. For this, we assume that we are given a set of n experiences E, which contains the values of various state and execution parameters[6], as well as a set of labels $\mathbf{y}$, such that $\mathbf{y}_i$ corresponds to whether the execution given parameters E_i was successful. We assume that the labels $\mathbf{y}$ are provided by a human teacher during learning, although it is also possible to automate the labelling, particularly in a simulated scenario, by verifying that the action effects have been achieved, as in [18], [8]. For relational learning, we convert E to a symbolic form S with a predefined set of relations using a mapping $r(E) \rightarrow \mathbb{N}^{n \times k}$, where k varies depending on the action (as discussed in the next section, different relations are used for different actions). On the other hand, for learning a success model, a mapping $g(E) \rightarrow \mathbb{R}^{n \times w}$ extracts from the raw data in E a preprocessed representation X that models the execution parameters of an action A; these preprocessed experience data are used as an input to the continuous model F.

1) Relational Learning: Let us denote the subset of S that corresponds to successful action executions by S^+, and assume that we have l such examples. Given S^+, we learn the set of preconditions using the procedure in [19], [20], which takes as preconditions those relations whose values are observed to be mostly constant in the successful executions. This procedure is briefly described below.

As a result of $r(E)$, each *column* j of S^+, which we will denote S^+_j, represents the observed values of a given relation P. The probability that S^+_j takes on a particular value P_q of P is then

$$P(S^+_j = P_q) = \frac{\sum_{i=1}^{l} \mathbb{I}_{S^+_{i,j} = P_q}}{l} \tag{1}$$

where $\mathbb{I}$ is an indicator function. Given these probabilities, we calculate the entropy $H(S^+_j)$

$$H(S^+_j) = -\sum_{i=1}^{q} P(S^+_j = P_i) \log_2 P(S^+_j = P_i) \tag{2}$$

and take S^+_j as a precondition if $H(S^+_j) < \delta$ for a predefined threshold δ. The value r^* of P that is taken as a precondition of the action is then

$$r^* = \arg\max_q P(P_q) \tag{3}$$

In case of m qualitative modes, the above learning procedure is performed m times, once for each qualitative mode.

2) Learning a Success Model: We model F as a Gaussian process (GP)

$$F(\mathbf{x}) = GP(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{4}$$

where $\mu(\mathbf{x})$ is the GP's mean at $\mathbf{x}$ and $k(\mathbf{x}, \mathbf{x}')$ is its covariance. A GP is a general non-parametric function approximator and additionally models the uncertainty of predictions, both attributes being particularly desirable for F: the former since a robot generally has no knowledge of the underlying success distribution and the latter since it allows making more informed predictions.[7] As in [22], we assume a linear regression model of the form $F(\mathbf{x}) = \phi(\mathbf{x})^T \omega$, such that we place a zero mean prior on the GP and use a squared exponential kernel of the form

$$k(\mathbf{x}_1, \mathbf{x}_2) = C \exp\left(-\frac{1}{2}\left\|\frac{\mathbf{x}_1 - \mathbf{x}_2}{\alpha}\right\|^2\right) \tag{5}$$

[6]For instance, in the case of grasping, this would be a representation of the object to be grasped as well as the chosen grasping pose and grasping strategy.

[7]Additionally, as demonstrated by Deisenroth et al. [21] in the discussion of the PILCO algorithm, GPs can also be very useful as forward models, particularly to speed up learning.

Algorithm 1 Model learning with teacher feedback

function LEARNACTIONMODEL(n, Q)
  $E \leftarrow \{\}$
  $\mathbf{y} \leftarrow \{\}$
  for $i \leftarrow 1$ **to** n **do**
    $\mathbf{e} \leftarrow$ selectExecutionParameters()
    $outcome \leftarrow$ executeAction($\mathbf{e}$)
    $y \leftarrow$ getTeacherFeedback($outcome$)
    $E \leftarrow E \cup \mathbf{e}$
    $\mathbf{y} \leftarrow \mathbf{y} \cup y$
  $S \leftarrow r(E)$
  $R \leftarrow$ extractPreconditions(S^+, $\mathbf{y}$, Q)
  $X \leftarrow g(E)$
  $F \leftarrow$ GP(X, $\mathbf{y}$)
  $M \leftarrow (R, F)$
  return M

where α is a scaling parameter for the lengths of $\mathbf{x}_1$ and $\mathbf{x}_2$ and C is a positive constant. The values of C and α are optimised from the training set X.

Given X, the posterior predictive distribution at a new set of points X_* has the parameters [22]

$$\mu_* = K(X_*, X)K(X, X)^{-1}\mathbf{y} \quad (6)$$

$$\Sigma_* = K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*) \quad (7)$$

where $K(X, X)$ is the kernel matrix between the training points, $K(X_*, X)$ is the kernel matrix between the new and the training points, and $K(X_*, X_*)$ is the kernel matrix between the new points.

It should be noted that, in this paper, we assume that $\mathbf{y}$ only takes the values 0 or 1, and, as such, a classification rather than a regression model may seem more appropriate. The reason for using a regression model is however twofold: (i) since we use a squared exponential kernel, which guarantees smoothness of F, the predicted function values and their uncertainties can be interpreted as confidence regions that move smoothly around the function spectrum[8] and (ii) in general, a more sophisticated success model may be incorporated based on which action executions are evaluated based on some degree - or preference - of success, much like in a reinforcement learning scenario.[9]

A summary of the learning procedure is provided in Algorithm 1. It should be noted that, as in [7], the learned model can clearly be refined by lifelong learning. The use of a GP as a continuous representation could potentially make this process even more efficient, as Bayesian optimisation could be used for executing actions at informative points of the function spectrum [12], [23]. We do not deal with this aspect in this paper however.

[8]It is in fact the case that predicted function values near the training points are around 1 with a high certainty, but the value decreases and the uncertainty increases the further away one goes from the training data.

[9]As an example, grasping an object successfully while knocking another object down is less preferable than grasping the same object without knocking any other objects; a success model may be able to encode that in a continuous form.

D. Models for Grasping and Pulling

As mentioned above, a relation set is required for learning the action preconditions R. In this paper, we focus on two actions as use cases: grasping a handle for opening a furniture item and pulling an object to a predefined region. We suggest qualitative relations for these actions below.

1) Handle Grasping: We model a grasping action through the relation between the robot's end effector and the handle to be grasped. The relations used for extracting grasping rules, which are given below, are thus defined in terms of the position of the gripper frame p and the bounding box B of the handle, both of which are assumed to be expressed in a common right-handed coordinate frame with a z-axis facing upwards.

$$\begin{aligned} in\_front\_of_{x,y}(p, B) &\coloneqq p_{x,y} < \min(B_{x,y}) \\ behind_{x,y}(p, B) &\coloneqq p_{x,y} > \max(B_{x,y}) \\ above(p, B) &\coloneqq p_z > \max(B_z) \\ below(p, B) &\coloneqq p_z < \min(B_z) \\ centered_{x,y,z}(p, B) &\coloneqq \left|p_{x,y,z} - \overline{B_{x,y,z}}\right| \leq \epsilon \end{aligned} \quad (8)$$

We consider grasping to be a free action, such that the input to F is the relative position between the gripper and the handle. The action is defined in a single qualitative mode.

2) Object Pulling: Unlike grasping, we model object pulling as a constrained action since the object needs to end up in a predefined region. For simplicity, we assume that this region is always behind the object so that the direction vector for pulling is towards the robot. We additionally assume that the position of the region relative to the object as well as the distance of the object to the surface edge are both known. Under these assumptions, we can vary the pulling duration as an action parameter.[10] Our relational model encompasses two separate aspects of the pulling action in order to take the temporal aspect of the action into account: (i) the preconditions under which pulling is allowed (both spatial and temporal) and (ii) constraints on the pulling duration. To represent the spatial preconditions for pulling, we use variations of the relations defined above, but defined in terms of the object and region poses p_1 and p_2 respectively:

$$\begin{aligned} in\_front\_of_{x,y}(p_1, p_2) &\coloneqq p_{1_{x,y}} < p_{2_{x,y}} \\ behind_{x,y}(p_1, p_2) &\coloneqq p_{1_{x,y}} > p_{2_{x,y}} \\ above(p_1, p_2) &\coloneqq p_{1_z} > p_{2_z} \\ below(p_1, p_2) &\coloneqq p_{1_z} < p_{2_z} \\ centered_{x,y,z}(p_1, p_2) &\coloneqq |p_{1_{x,y,z}} - p_{2_{x,y,z}}| \leq \epsilon \end{aligned} \quad (9)$$

The relational model includes the temporal aspect of pulling through qualitative modes depending on the distance between the object and the region. More precisely, we split the action into qualitative modes using a relation $distance$,

[10]We could obviously calculate the duration explicitly given the known information, but we would instead like the robot to learn *why* it needs to pull for a certain duration.

whose possible values are *short* and *long*, and use a relation $duration$ to also represent the pulling duration qualitatively, also as *short* and *long*. These relations are defined below:

$$distance = \begin{cases} \text{short}, & d < d_\tau \\ \text{long}, & \text{otherwise} \end{cases}$$
$$duration = \begin{cases} \text{short}, & t < t_\tau \\ \text{long}, & \text{otherwise} \end{cases} \quad (10)$$

where d is the distance between the object and the goal region, t is the pulling duration, and d_τ and t_τ are predefined distance and time thresholds respectively. It should be noted that the $distance$ relation effectively defines two branches in R: one for the preconditions when the distance is short, and one for when the distance is long. The pulling duration is also represented through the continuous function F, whose input is composed of (i) the relative position between the object and the goal region, (ii) the distance from the object to the surface edge and (iii) the pulling duration. As described in section III-B, F encompasses both qualitative pulling modes.

E. Sampling Execution Parameters

The procedure for sampling execution parameters differs slightly depending on whether an action is free or constrained. To find execution parameters for a free action, we first greedily sample a point $\hat{\mathbf{x}}$ from X at which the GP confidence of execution success is higher than a threshold τ. The execution parameters $\mathbf{x}$ are then sampled from the GP at $\hat{\mathbf{x}}$. For a constrained action, given a constraint $\mathbf{c}$, we first find a set of points N from X at which the values of the constraints are closest to $\mathbf{c}$.[11] We then sample $\hat{\mathbf{x}}$ from N and proceed with the parameter sampling as in the case of a free action. The given sample is only accepted if the learned preconditions are satisfied under the given mode q; this ensures that the sampling procedure does not return parameters that violate the action preconditions. A summary of the sampling procedure is given in Algorithm 2.

Algorithm 2 Sampling parameters for action execution. The higher-order function `sample` returns a training point at which the GP's confidence of execution success is higher than the threshold τ, while the function `sampleGP` samples from the GP at a given point. The function `f` predictively applies the execution parameters and verifies that the action preconditions are satisfied after the application under the given mode q.

function SAMPLEPARAMETERS((R, F), X, $\mathbf{y}$, $\mathbf{c}$, q, τ)
  if $\mathbf{c} \neq \varnothing$ **then**
    $X \leftarrow$ knn(X, $\mathbf{c}$)
    $\mathbf{y} = \mathbf{y}_X$
  $\mathbf{x} \leftarrow \varnothing$
  $sample\_found \leftarrow$ **false**
  while $sample\_found =$ **false do**
    $\hat{\mathbf{x}} \leftarrow$ sample(F, X, $\mathbf{y}$, τ)
    $\mathbf{x} \leftarrow$ sampleGP(F, $\hat{\mathbf{x}}$)
    if f(R, $\mathbf{x}$, q) **then**
      $sample\_found \leftarrow$ **true**
  return $\mathbf{x}$

IV. EXPERIMENTS

We verify our model by learning how to grasp handles (in particular, drawer and fridge handles) and pull an object (a plastic cup) - with a Toyota HSR robot. The experimental setup, which is illustrated in Fig. 2, is similar in both cases: the robot is placed in front of a drawer and a fridge in the grasping case (we learn two different models, one for each handle) and in front of a table in the pulling case. For the pulling action, we keep the position of the goal region fixed (near the edge of a table) and vary the object position, such that we use $d_\tau = 15cm$ and $t_\tau = 1.5s$.[12] The action of grasping a handle is preceded by detecting the handle's 3D pose.[13] Before pulling, the robot also scans the table to find the location of the object to be pulled and then grasps the object.[14] After detection, execution parameters are sampled and the robot executes the action. For model learning, we collect 50 execution samples: for the grasping action, we sample grasping points within and around the bounding box of the detected handle, while for the pulling action, we sample pulling durations around the theoretically optimal value. After learning, we let the robot perform the actions again 50 times. For both actions, we compare two variations of our sampling procedure: one in which all samples are accepted without verifying the action preconditions, which we call R^--opt, and the one in Algorithm 2, which is referred to as R^+-opt.[15]

(a) Drawer handle (horizontal) (b) Fridge handle (vertical) (c) Object pulling

Fig. 2: Experimental setup

A. Grasping a Handle

The sets of training points for learning handle grasping models for the drawer and fridge handles are shown in Fig. 3. In both cases, the successful grasping points are near the handle centers. In addition, it should be noted

[11]The k-nearest neighbours algorithm is used for this purpose.

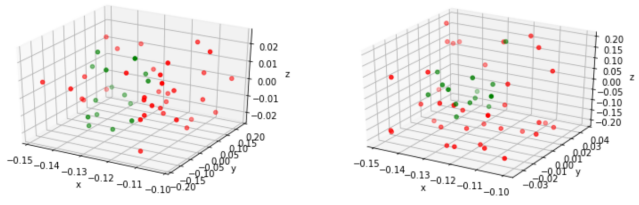
[12]The distance to the goal region is measured manually before each trial.

[13]For recognising handles, we use a Faster R-CNN model [24] trained on our custom handle data set. The trained model can be found at https://github.com/b-it-bots/mas_models/tree/master/perception_models/detectors

[14]We use SSD [25] pretrained on the COCO data set for detecting objects: https://github.com/b-it-bots/ssd_keras_ros. In the pulling experiments, grasping is performed using a simple heuristic - the midpoint of the object's point cloud - rather than with a learned model.

[15]Code for model learning as well as guidelines for reproducing our experiments can be found at https://github.com/alex-mitrevski/explainable-robot-execution-models

that the successful poses along the x-axis are all more than $10cm$ away from the estimated handle pose; this is due to inaccuracies in the handle depth estimate.



(a) Drawer handle (b) Fridge handle

Fig. 3: Relative distances between gripper positions and estimated handle positions (in m) for learning handle grasping models. The green points denote successful grasps, while the red points unsuccessful ones.

The learned precondition model for the drawer handle is given below.

$$centered_z(p, B) \wedge in\_front\_of_x(p, B) \wedge \\ \neg centered_x(p, B) \wedge \neg in\_front\_of_y(p, B) \wedge \\ \neg behind_{x,y}(p, B) \wedge \neg above(p, B) \wedge \neg below(p, B)$$

Since the drawer handle is horizontal, the precondition model correctly encodes that the gripper should be centered along the z-axis and in front of the handle so that the handle can be grasped correctly. Similarly, the precondition model for the fridge handle is given below.

$$centered_y(p, B) \wedge in\_front\_of_x(p, B) \wedge \\ \neg centered_x(p, B) \wedge \neg in\_front\_of_y(p, B) \wedge \\ \neg behind_{x,y}(p, B) \wedge \neg above(p, B) \wedge \neg below(p, B)$$

The fridge handle is vertical, so the preconditions are that the gripper should be centered along the y-axis and, as above, in front of the handle.

The numbers of successful grasps before and after learning are provided in Table I.

TABLE I: Successful handle grasps (out of 50)

Handle \ Grasps	Before learning	R^--opt	R^+-opt
Drawer	15	34	**41**
Fridge	14	33	**44**

As these results show, verifying the preconditions increases the number of successful executions. At the same time, it should be noted that both R^--opt and R^+-opt decrease the number of severe failures: in this case, many failures before learning were caused due to the robot going too far and thus hitting the drawer with the gripper, while after learning, the more common failures were ones in which the robot was too conservative and was not going far enough.

B. Pulling an Object

The set of points for learning a pulling execution model is visualised in Fig. 4. We only show the relative x, y distances between the object and the goal region as well as the pulling times here; the distance to the surface edge is not displayed.

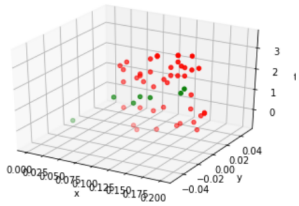


Fig. 4: Relative x, y distances between object and goal positions (in m) and pulling times (in s) for learning an object pulling model. The green points represent successful pulls; the red points unsuccessful ones.

Since the pulling action is split into two qualitative modes, there are two precondition models, one for each qualitative mode. The learned preconditions for a short distance between the object and the goal region are shown below.

$$centered_{y,z}(p_1, p_2) \wedge duration = \text{short} \wedge \\ \neg in\_front\_of_{x,y}(p_1, p_2) \wedge \neg behind_y(p_1, p_2) \\ \wedge \neg above(p_1, p_2) \wedge \neg below(p_1, p_2)$$

As would be expected under our assumption, a precondition is that the object is centered along the y-axis (and z as well since we are pulling on a table). It should be noted that the object being behind the goal region is not necessarily a precondition since the object can also start at the goal region, in which case no pulling is necessary.

The learned preconditions in the case of a long distance between the object and the region are given below.

$$centered_{y,z}(p_1, p_2) \wedge behind_x(p_1, p_2) \wedge \\ duration = \text{long} \wedge \neg in\_front\_of_{x,y}(p_1, p_2) \\ \wedge \neg behind_y(p_1, p_2) \wedge \neg above(p_1, p_2) \\ \wedge \neg below(p_1, p_2) \wedge \neg centered_x(p_1, p_2)$$

In this case, the list additionally includes that the object should be behind the region (expressed by $behind_x$ and $\neg centered_x$), which is in contrast to the short distance case.

The numbers of successful object pulls before and after learning are shown in Table II.

TABLE II: Successful object pulls (out of 50)

Before learning	R^--opt	R^+-opt
7	24	**38**

As in the handle grasping case, both R^--opt and R^+-opt improve the execution success, such that, as before, R^+-opt leads to more successful executions. An additional

interesting observation from these experiments is that, before learning, a significant number of failures were caused by the object being pulled for too long, namely over the surface edge; after learning, only one instance failed in this manner, namely most failures were caused due to the robot not pulling the object for long enough. This means that, as a result of the learning, the number of successes increases, while the number of severe failures decreases.

V. DISCUSSION AND CONCLUSIONS

The representation discussed in this paper allows a robot to ground action parameters in order to maximise its chances of execution success, but also serves as an explicit, explainable model of why a robot makes certain decisions during execution. This is accomplished by describing the preconditions for executing actions with a relational model of execution in the form of action-specific predicates and qualitative attributes; the relational model is combined with a Gaussian process, which creates a mapping from action parameters to predicted execution success. In a set of experiments with a Toyota HSR robot, the joint relational-continuous model was shown to improve the success rate and to reduce the number of severe failures in two scenarios - grasping furniture handles and pulling an object to a target region.

Despite the promising results, there are various aspects that future work needs to address. For the purposes of the experiments in this paper, robot executions were evaluated by a human teacher; however, for faster and more extensive experience acquisition as well as for lifelong learning, a simulated environment could be incorporated, as is common in reinforcement learning [26] or as in [18]. The use of the models in combination with a simulation may also be useful when considering complete tasks, where the outcome of one action may affect the outcomes of subsequent actions; in this context, it would be worthwhile to use the models with a framework for projecting action outcomes over time, such as [27]. For formally guaranteed generalisation of the learned models beyond the learning scenarios, an object and scene ontology should be considered. Finally, in the context of fault diagnosis, where a robot may need to investigate different explanations for a failure, the learned relational models could provide a systematic way of exploring such alternatives, potentially in the form of counterfactual reasoning.

ACKNOWLEDGMENT

We would like to thank Ahmed Abdelrahman and Anirudh Narasimamurthy for various discussions and for supporting the development efforts in the handle grasping scenario.

REFERENCES

[1] M. Ersen, E. Oztop, and S. Sariel, "Cognition-Enabled Robot Manipulation in Human Environments: Requirements, Recent Work, and Open Problems," *IEEE Robotics Automation Magazine*, vol. 24, no. 3, pp. 108–122, Sept. 2017.

[2] D. Paulius and Y. Sun, "A Survey of Knowledge Representation in Service Robotics," *Robotics and Autonomous Systems*, vol. 118, pp. 13–30, 2019.

[3] R. K. Sheh, ""Why Did You Do That?" Explainable Intelligent Robots," in *Workshops at the 31st AAAI Conf. Artificial Intelligence*, 2017.

[4] J. de Kleer and B. C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence*, vol. 32, pp. 97–130, Apr. 1987.

[5] R. Reiter, "A Theory of Diagnosis from First Principles," *Artificial Intelligence*, vol. 32, no. 1, pp. 57–95, Apr. 1987.

[6] M. Kopicki, S. Zurek, R. Stolkin, T. Moerwald, and J. L. Wyatt, "Learning modular and transferable forward models of the motions of push manipulated objects," *Autonomous Robots*, vol. 41, no. 5, pp. 1061–1082, June 2017.

[7] L. Kaelbling and T. Lozano-Perez, "Learning composable models of parameterized skills," in *Proc. 2017 IEEE Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 886–893.

[8] A. Mitrevski, A. Kuestenmacher, S. Thoduka, and P. G. Plöger, "Improving the Reliability of Service Robots in the Presence of External Faults by Learning Action Execution Models," in *Proc. 2017 IEEE Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 4256–4263.

[9] T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, "Development of Human Support Robot as the research platform of a domestic mobile manipulator," *ROBOMECH Journal*, vol. 6, pp. 1–15, 2019.

[10] S. Karapinar and S. Sariel, "Cognitive Robots Learning Failure Contexts through Real-World Experimentation," *Autonomous Robots*, vol. 39, no. 4, pp. 469–485, Dec. 2015.

[11] F. Stulp, A. Fedrizzi, and M. Beetz, "Action-related place-based mobile manipulation," in *2009 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Oct. 2009, pp. 3115–3120.

[12] Z. Wang, L. P. Garrett, C. R.and Kaelbling, and T. Lozano-Pérez, "Active Model Learning and Diverse Action Sampling for Task and Motion Planning," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2018, pp. 4107–4114.

[13] J. Kober and J. Peters, *Reinforcement Learning in Robotics: A Survey*, 2012, pp. 579–610.

[14] M. P. Deisenroth, G. Neumann, and J. Peters, "A Survey on Policy Search for Robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1-2, pp. 1–142, Aug. 2013.

[15] S. Höfer and O. Brock, "Coupled learning of action parameters and forward models for manipulation," in *2016 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 3893–3899.

[16] A. Orthey, M. Toussaint, and N. Jetchev, "Optimizing Motion Primitives to Make Symbolic Models More Predictive," in *2013 IEEE Int. Conf. Robotics and Automation*, May 2013, pp. 2868–2873.

[17] R. Caruana, "Multitask Learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.

[18] A. Kuestenmacher, N. Akhtar, P. G. Plöger, and G. Lakemeyer, "Towards Robust Task Execution for Domestic Service Robots," *Journal of Intelligent & Robotic Systems*, vol. 76, no. 1, pp. 5–33, 2014.

[19] S. Ekvall and D. Kragic, "Learning Task Models from Multiple Human Demonstrations," in *15th IEEE Int. Symp. Robot and Human Interactive Communication*, Sept. 2006, pp. 358–363.

[20] N. Abdo, H. Kretzschmar, L. Spinello, and C. Stachniss, "Learning Manipulation Actions From a Few Demonstrations," in *Robotics and Automation (ICRA), 2013 IEEE Int. Conf.*, May 2013, pp. 1268–1275.

[21] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian Processes for Data-Efficient Learning in Robotics and Control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 408–423, 2015.

[22] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[23] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization," in *2017 IEEE Int. Conf. Robotics and Automation (ICRA)*, May 2017, pp. 1557–1563.

[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems 28*, 2015, pp. 91–99.

[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: https://arxiv.org/abs/1512.02325

[26] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[27] L. Mösenlechner and M. Beetz, "Fast Temporal Projection Using Accurate Physics-Based Geometric Reasoning," in *2013 IEEE Int. Conf. Robotics and Automation*, May 2013, pp. 1821–1827.