Improving the Reliability of Service Robots by Symbolic Representation of Execution Specific Knowledge

Anastassia Kuestenmacher[1*], **Paul G. Plöger**[1]

[1]Hochschule Bonn-Rhein-Sieg University of Applied Science, Germany
anastassia.kuestenmacher@h-brs.de, paul.ploeger@h-brs.de

Abstract

In the field of service robots, dealing with faults is crucial to promote user acceptance. In this context, this work focuses on some specific faults which arise from the interaction of a robot with its real world environment due to insufficient knowledge for action execution.

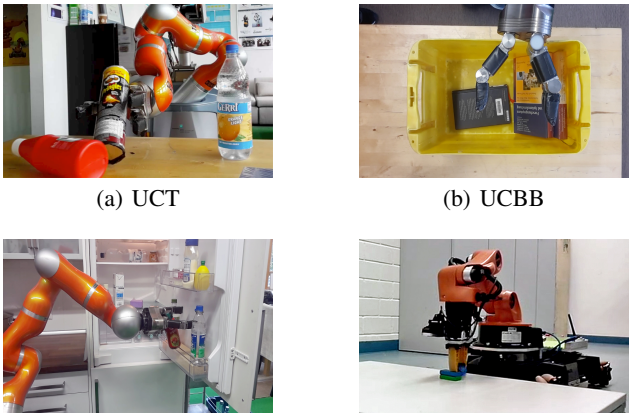
In our previous work [1], we have shown that such missing knowledge can be obtained through learning by experimentation. The combination of symbolic and geometric models allows us to represent action execution knowledge effectively. However we did not propose a suitable representation of the symbolic model.

In this work we investigate such symbolic representation and evaluate its learning capability. The experimental analysis is performed on four use cases using four different learning paradigms. As a result, the symbolic representation together with the most suitable learning paradigm are identified.

1 Introduction

Even the most carefully designed and tested robots may encounter situations that disallow them to execute their tasks in a desirable manner. These situations may be caused by a malfunction of their internal components such as broken hardware or software. However some situations can arise during the communication of a robot with its environment. Such situations are usually unpredictable during the development stage and remain unknown until they occur. In this paper we refer to these situations as external faults.

One of the main origins of external faults can be an imprecise or even wrong model of the execution of an action. For example in order to place a bottle on a surface, it should be released with the correct orientation. If there is no information about orientation available to the robot, it may release the bottle in such a way that it rolls and falls on the table like illustrated in Figure 1(a), we refer to this example as Use Case - Table (UCT). The other example of external fault is Use Case - Book Box (UCBB) shown in Figure 1(b). Here the Care-O-bot III is trying to put a book in a box that contains other books. While doing so the robot releases the book in a position that finally results in this book falling on the other books in the box. Again an imprecise knowledge about releasing a book brings in an unsuccessful completion of an action. In the next example that is Use Case - Fridge (UCF) see Figure 1(c), the Care-O-bot III tries to place an object (a bottle) in the fridge door. The robot is not able to do this correctly i.e. the bottle falls from the door on to the floor. In the final Use Case - Cube Tower the objective of a robot youBot is to stack blocks (Figure 1(d)). However youBot fails to release the green cube successfully. The main cause of external faults in the last two examples is too general knowledge about the actions' execution.



(a) UCT (b) UCBB

(c) UCF (d) UCCT

Figure 1: Examples of external fault.

So, in general the occurrence of external faults can be mitigated by considering more details about the initial state of the manipulated object: its pose and location with respect to other objects in the environment. In fact the information about states of the objects before the execution of the action is encoded in the preconditions of the corresponding planning operator. However this information is too abstract and would not be sufficient to ensure the successful execution of an action. On the other hand the detailed numeric information about position for the execution an action on the low-level is too specific and cannot be generalised for similar scenarios and various environment configurations. We need a tool

*Contact Author

which allows us to combine qualitative and quantitative descriptions of the objects' states in a given environment.

In our previous work [1], we introduced an approach where, the execution specific knowledge is encoded into an action execution model (AEM). The AEM is built as an intermediate level between a high level of semantic knowledge, which includes object-specific information, and a low-level execution module, which uses the solutions produced by AEM and performs the actual task. The symbolic part of AEM encodes qualitative descriptions of the execution of an action. Such symbolic model can be used to describe a certain way of how a robot should perform an action for example of placing one object on a top of the other successfully. For instance, that a manipulated object in a robot's gripper should be fully overlap with a table or an object on which it should be released and a release location should be occupied by other objects. The symbolic model is represented by a set of logical predicates which are defined separately for each action. As shown in [2] such representation of a symbolic model can simplify the task of analysing a robot's behaviour for its developers and designers, since the robot's decisions are represented in an explicit way; in other words, the symbolic model can be used to explain why the robot performs actions in a certain way, such as why an object was released in a certain way. This model can be learned from experiments.

Moreover in that work [1], we identified that the geometric representation presented in [3] is the suitable model to the proposed symbolic model that is viewed as a conjunction of predicates. In this paper, the authors encode the probability distribution of various geometric parameters using Kernel Density Estimation (KDE). In the *release* action example such parameters are orientating the manipulated object, the height of the release as well as the position of the manipulated object with respect to the table. In the work [2] the authors propose to represent the geometric model as the Gaussian processes. Such representations for the geometric model can encode prediction uncertainties. That means, given a particular value for the parameters, the probability can be calculated as regards the likelihood that the action will be executed successfully.

However, a decision still needs to be made about which kind of predicates to use in order to describe symbolic model namely the aspects of the objects involved in the action execution as well as how to combine these predicates into one symbolic representation. Although in this work we concentrate on developing a set of predicates for a *release* action, this can be also applied for presenting other actions which involve spatial information about objects.

Therefore the main contribution of this work is the development and evaluation of a set of the predicates as well as their usage for the constructing a symbolic model. This symbolic model should be further applicable to simplify the mapping to the existing geometric model described in [1]. The other important advantage of such representation is that we can define symbolic model to describe the desired state of objects after the execution of an action so that if we use a simulation environment to generate different examples of the behaviour of the object for learning then such symbolic description of action' effects allows autonomously label these examples as desired (i.e. positive) or undesired (i.e. negative). The labeling process by capturing qualitative description of objects' behaviour is described in [4].

The remainder of this paper is organised as follows: In the next section we present an overview of the literature related to the symbolic representation of an action. Section 3 then specifies a symbolic representation for presenting spatial relations between two objects involved in execution of a *release* action. In Section 4 of this paper we introduce the set of predicates which are used to construct a symbolic representation. Once the symbolic representation is established and set of predicates is specified the methods to learn such representation from the set of training examples are introduced and evaluated in the Section 5. Finally, we draw conclusions about the symbolic representation of execution specific knowledge and discuss the future direction of optimisation of such representation in Section 6.

2 Related Work

To get an idea about predicates, the literature which describe the relationship between objects involved in the action execution is discussed.

Object Relations: Rosman and Ramamoorthy [5] propose the use of supervised learning techniques for predicting spatial relationships between pairs of objects. Having point cloud representations of two objects, the algorithm first trains the SVM to find the separation margin between two point clouds. Next the support vectors of each object are grouped into K clusters. Finally, these clusters are used to establish the object's relations. The main limitation of the proposed algorithm is that it can only support two simple spatial relationships: *On* and *Adjacent*.

Other recent work on learning geometrical relations between pairs of objects is discussed by Sjöö and Jensfelt in [6]. Compared to the previous work by Rosman and Ramamoorthy, the proposed approach learns spatial relations indirectly. It works on such qualitative criteria as: *support*, *support force*, *protection*, *constraint* and *location control*. Since the proposed approach requires considerable training data, the authors use a simulation to generate different types of object behaviours such as pushing, throwing and lifting. The main conclusion with respect to our work is that learning the relationship between objects is easier when the objects are in pairs. Also worth noting is that the absolute poses are insignificant for categorising relations.

The focus of Kulick et al. [7] is to actively learn symbolic relation representations. In this work, a robot learns ground symbols during the experimental session. The training data is not given in advance but obtained by a robot during its interaction with an environment. Kulick et al. use the active learning strategy to directly guide the acquisition of the data.

Here, the first two papers provide us with important insights into how to use a physics engine to verify the given symbolic representation and how to find appropriate geometric positions. The strategies of the other papers illustrate which aspects could be relevant when defining the kind of relations between objects. An important point with respect to our work is that pairwise representation or binary predicates

are simple to describe and can be successfully learned and generalised.

Spatial Relations: Spatial models allow the relations between objects in a scene to be formalised so that the robot is operating in a similar way to how we humans structure our perception of space. Spatial reasoning is a theory concerning representing, analysing and reasoning with regard to various objects in a multidimensional space. It allows topological relationships between two objects to be represented like *connect*, *overlap*, *disjoint* as well as directional concepts like *right*, *left*, *behind*. In this section the literature on representations of spatial relations in three-dimensional space ($\mathbb{R}^3$ or 3D space[1]) is discussed.

One of the first works in the field of presenting relationships between 3D objects in 3D space is given by Egenhofer in [8]. The author uses the 9-intersection model to identify spatial relations between two objects in 3D space. The 9-intersection model is a set of nine possible relations between two objects A and B. Based on this, the amount of all relationships, which can be obtained from the nine intersections, is 2^9 or 512 mutually exclusive topological relations. However, not all of the relationships can occur in reality. In order to define non-realisable relationships the author specifies a set of conditions. It should be noted that only simple objects, e.g bodies without holes are considered.

Zlatanova in [9] continues to investigate the conditions to eliminate non-realisable relationships between multidimensional spatial objects, referred to as *negative conditions*. She notices that *topological relations* only are not sufficiently expressive for the qualitative representation of the execution knowledge. Therefore, in addition to the *topological relations*, the *directional* as well as the *orientation relations* have to be included.

Albath et al. [10] realise that the domain specification also influences the number of relations. In order to achieve the required expressiveness, additional qualitative information is necessary. The authors therefore suggest accounting for the orthogonal projection of 3D objects on the corresponding planes in $\mathbb{R}^3$.

In general, *3D region connection calculus* (RCC-3D) contains 13 spatial relations between two 3D objects in $\mathbb{R}^3$. These 13 relations are based on eight relations of 3D objects and three relations of the 2D projections of these objects. Since the tested 3D objects are taken from images, the authors have to annotate the objects with additional geometrical information such as sizes of objects and their boundaries. That is unlike the domain here, where the simulation environment is used and the CAD-models of all objects are given.

A more expressive RCC-3D is presented by Sabharwal et al. in [11]. The authors describe the enhanced model that extends 13 relations defined in RCC-3D [10] and [8] to 34 relations by adding the information about the distance from the projection plane i.e. a parameter which differentiates between occlusion relations (*in front of* or *behind*). As a result the number of occlusion relations increases from two to eleven.

Spatial Relation in Combination with Robot Planning and Learning: Mojtahedzadeh et al. [12] propose an approach of automatically building and using high-level symbolic representations. Such representations describe physical interactions between objects in static configurations. The authors validate their approach on action of removing objects from piles. Depending on the outcome of a perception component of a robot (if objects in a scene fully or partially observable) two methods are suggested to learn and extract the symbolic representation. In our approach we make an assumption that the surround environment is fully observable and we can upload it to a simulator to generate datasets for the learning purpose. The main goal of our approach is to use the spatial representation to label the examples automatically.

Kunze et al. in [13] show how to use knowledge about landmark objects and their spatial relationship to the sought object, to improve its search by directing the robot towards the most likely object locations. This work gives us the support of the idea that the description of spatial relations between objects provides more detail information about objects in their environment and allows to execute an action more accurately that avoids occurring of external faults.

The majority of research papers for learning symbolic and geometric representation of actions focused on how to enable robots to learn such representations from observations, demonstrations or instructions. However there are less work available on how actions' representation can be corrected if it was learned incorrectly or if it contained incomplete knowledge for successful execution. Moreover how to adopt such representation to changing/dynamic environment or generalise to different contexts. Frasca et al. in [14] address this issue by introducing a framework that allows designers to update available representation of robot's actions through instructions. The authors empirically show that the proposed framework allows to modify robot knowledge in the way that it reduces inefficiencies, fix errors, and enable generalisations,

The other way to learn symbolic representation of an action execution is from the instructions of a human teacher given through natural language and demonstration [15], [16]. Although in our work we use a simulation environment to generate examples for learning symbolic and geometric representation since with its help the sufficient amount of examples can be generated, learning by demonstration can be considered as a future work to extract symbolic representation.

3 Object Relation

To represent all kinds of release actions, we have to consider two main interacting parts: a *manipulated object* (`mo`) (a target object to be placed) and a *destination object* (`do`) (a location where `mo` should be placed). In addition to `mo` and `do`, the surrounding environment may also contain the other objects to which we refer to as *passive objects*. The successful performance of the release action depends on knowledge about the spatial relationship between the manipulated object, the destination object as well as any passive objects involved in

[1]The term *3D space* and the symbol $\mathbb{R}^3$ are used as synonyms and refer to three-dimensional Euclidean space. Similarly *2D space* and $\mathbb{R}^2$ are synonyms for the two-dimensional Euclidean space.

the execution of the action.

Based on the literature, the pairwise relations can simplify the description of a scene, be successfully learned as well as efficiently used for mapping symbolic predicates to geometric states. These properties make pairwise relations, which can be described by binary predicates, the perfect candidates for the predicate pool. Moreover, using the pairwise relation maximises the generalisation aspect. A configuration of two objects which is recorded in action representation using a finite number of binary predicates can be generalised to configurations that have any number of objects. In other words any configuration with an arbitrary number of objects includes an instance of the particular configuration. As have been said, it is easier to extract the geometric state for symbolic predicates which describe the relation between two objects. For example, to solve a task of symbolic to geometric mapping, Dearden and Burbridge [3] consider only symbolic predicates of two objects and demonstrate the way of using the conjunctions of such predicates for various objects in order to find the geometric state representation for environments with multiple objects. Based on described advantages, pairwise relations are suitable candidates to be predicates for predicate pool. The next question which should be considered is which kind of relations should be used.

Spatial relation knowledge is commonsense knowledge that people use for reasoning about everyday situations. A spatial relation specifies how an object is located in space relative to a reference object. Using spatial relations, a robot may reason about space in a way that comes closer to human cognition. These relationships are essential to such robotic aspects as manipulation and decision making. They simplify understanding of task specifications, and enhance robustness in planning. A robot can learn or construct spatial relations from human commands, demonstrations, planning operators, etc.

Despite extended usage of spatial relations in various applications, there are several challenges involved when working with them. Firstly, the theoretical studies and implementations of spatial reasoning mostly focus on 2D relations. Secondly, the existing 3D representations are often too general and not expressive enough to describe detailed execution knowledge. For the symbolic representation of AEM, the relations between pairs of 3D objects should be described. In addition, we have to consider both possibilities such as general, e.g. releasing a manipulated object around some passive object on a table, as well as specific, e.g. releasing a manipulated object in front of and left of the passive object on the table.

In [11] the authors describe the spatial relation between two bodies as the combination of 2D and 1D projections of these bodies on the top-, side-, or front-plane and the axis orthogonal to the selected plane. Following this idea, our definition of the spatial relations between pair of regions (2D projections of the bodies on the plane) deploys the most notable *RCC8* models in combination with the *interval relations* between two line segments (1D projections of the bodies on the axis). Allen's interval algebra [17] was chosen for describing interval relations. The RCC8 relations are binary relations that cover topological arrangements of spatial regions, while Allen's interval algebra defines binary relations that capture the relations between two intervals of time, here these intervals are along coordinate axes.

The general formula for presenting spatial relations between two 3D objects $\mathtt{oo}_1$ and $\mathtt{oo}_2$ is defined as follows:

Definition 3.1 (*Spatial Relation*($\mathtt{oo}_1$, $\mathtt{oo}_2$))**.**
RCC8-$\mathtt{xy}(\mathtt{oo}_1, \mathtt{oo}_2) \wedge$ *Interval*-$\mathtt{z}$ $(\mathtt{oo}_1, \mathtt{oo}_2) \wedge$
RCC8-$\mathtt{yz}(\mathtt{oo}_1, \mathtt{oo}_2) \wedge$ *Interval*-$\mathtt{x}$ $(\mathtt{oo}_1, \mathtt{oo}_2) \wedge$
RCC8-$\mathtt{zx}(\mathtt{oo}_1, \mathtt{oo}_2) \wedge$ *Interval*-$\mathtt{y}$ $(\mathtt{oo}_1, \mathtt{oo}_2)$.

The definition of any spatial relation between two bodies can

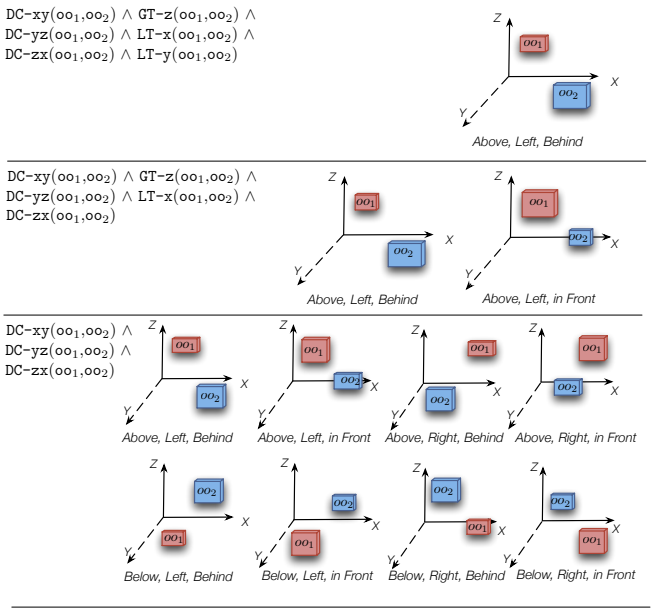


Figure 2: Examples of *spatial relations* between two objects `oo1` and `oo2`

be created by instantiating the corresponding *RCC8* and *interval relations* predicates into this formula. The way the *RCC8* and *interval relations* are used in this work slightly deviates from normal practice. There are two major reasons for this:
(i) In contrast to qualitative spatial reasoning (QSR) or temporal reasoning, this work does not use the relations (i.e. the predicates) for reasoning purposes. They are only used to provide the information about the state of the objects.
(ii) The existence of reference frame(s) is not presumed in both *RCC8* and *interval relations*. However, in robotic manipulation tasks a reference frame(s) is always defined. Therefore these frames are advantageous when defining predicates, they make it possible to find computationally effective implementations of the predicates.

4 Predicate Pool

The major predicates used in this work are shown in Figure 3. The predicates are grouped based on their aspect. Group (a) contains intervals which correspond to the distance aspect, while group (b) consist of eight basic topological RCC8 predicates and group (c) includes unary predicates for describing

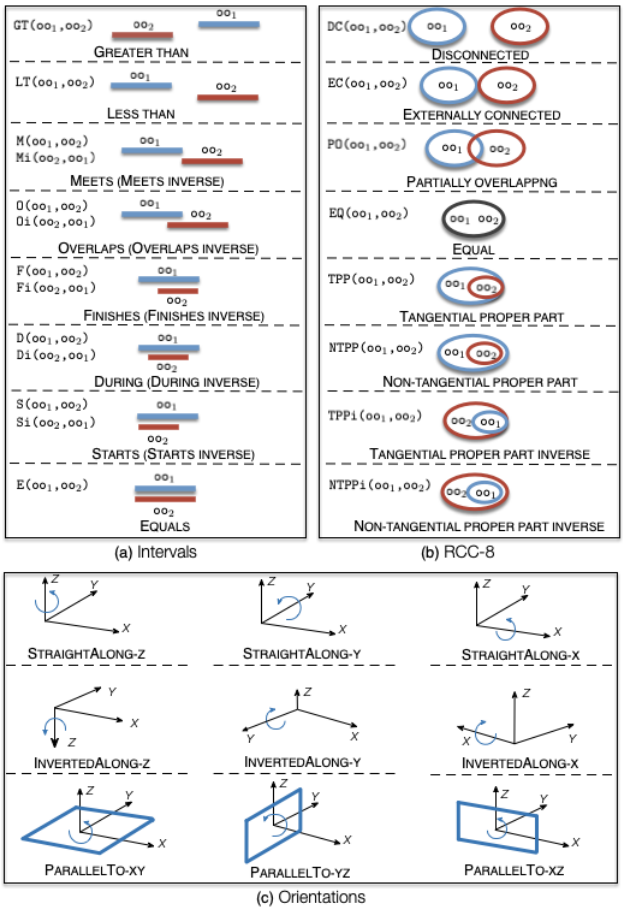


Figure 3: Set of predicates

orientation aspect of manipulated object. The first two groups of predicates are described in the next paragraph whereas the predicates from the last group are considered later in this section.

The predicate pool consists of 73 predicates in total. The RCC8 relations are defined for regions in 2D. These regions are given as projections of an axis-aligned bounding box of an object on the xy-, yz-, zx-planes (8 predicates for each plane that makes 24 predicates in general). The interval relations are specified as projection of the bounding boxes on the orthogonal to the particular plane axis, i.e. for the xy-plane the orthogonal axis is z-axis, or the yz-plane is x-axis and for the zx-plane is y-axis (13 predicates for each axis and 39 in general).

The main advantage of presenting the relationship between 3D objects with the help of their 2D and 1D projections is that such formalisation can be used not only to define a variety of spatial relations but also to present the different levels of generalisation. For instance, consider four examples presented below and visualised in Figure 2 , where each example describes the *spatial relations* between two objects `oo1` and `oo2`:

Example (a):

$\texttt{DC-xy}(\texttt{oo}_1,\texttt{oo}_2)\wedge\texttt{DC-yz}(\texttt{oo}_1,\texttt{oo}_2)\wedge\texttt{DC-zx}(\texttt{oo}_1,\texttt{oo}_2)$

Example (b):

$\texttt{DC-xy}(\texttt{oo}_1,\texttt{oo}_2)\wedge\texttt{GT-z}(\texttt{oo}_1,\texttt{oo}_2)\wedge$
$\texttt{DC-yz}(\texttt{oo}_1,\texttt{oo}_2)\wedge\texttt{DC-zx}(\texttt{oo}_1,\texttt{oo}_2)$

Example (c):

$\texttt{DC-xy}(\texttt{oo}_1,\texttt{oo}_2)\wedge\texttt{GT-z}(\texttt{oo}_1,\texttt{oo}_2)\wedge$
$\texttt{DC-yz}(\texttt{oo}_1,\texttt{oo}_2)\wedge\texttt{LT-x}(\texttt{oo}_1,\texttt{oo}_2)\wedge$
$\texttt{DC-zx}(\texttt{oo}_1,\texttt{oo}_2)$

Example (d):

$\texttt{DC-xy}(\texttt{oo}_1,\texttt{oo}_2)\wedge\texttt{GT-z}(\texttt{oo}_1,\texttt{oo}_2)\wedge$
$\texttt{DC-yz}(\texttt{oo}_1,\texttt{oo}_2)\wedge\texttt{LT-x}(\texttt{oo}_1,\texttt{oo}_2)\wedge$
$\texttt{DC-zx}(\texttt{oo}_1,\texttt{oo}_2)\wedge\texttt{LT-y}(\texttt{oo}_1,\texttt{oo}_2)$

In Example (a) `oo1` and `oo2` are disconnected that is the most general. It contains only information about 2D projections of the object on three planes. If a constraint along the z-axis (i.e. $\texttt{GT-z}(\texttt{oo}_1, \texttt{oo}_2)$ that is defined as the lowest point of $\texttt{oo}_1$ is greater than the highest point of the $\texttt{oo}_2$) is added to this expression, then the number of possible of relations is reduced. Here the object $\texttt{oo}_1$ is in the relation above and disconnected with respect to the $\texttt{oo}_2$. The second expression can be further limited by including the second interval constraint along x-axis. The expression in Example (c) defines that the object $\texttt{oo}_1$ can only be placed above and left of object $\texttt{oo}_2$. Finally, the expression in Example (d) has six projections, i.e. three on 2D planes and three on axis. It is the most specific and leaves open only one possible spatial relation: $\texttt{oo}_1$ is above, left of and behind the $\texttt{oo}_2$.

It should be noted that there is no unique symbolic representation of spatial relations between two objects in space. However, this allows the relations in various levels of abstractions to be represented depending on task requirements, e.g. performing an action of placing a mug in a sink to fill it with water requires a low level of abstraction (i.e. more exact location of an mug) in comparison to a task of placing a mug on a table.

Although it is possible to present many of the spatial configurations between two objects in detail using the described expressions, sometimes this information alone is not sufficient, especially for the release action. Very often to ensure safe release the *orientation* of a manipulated object has to be specified too. Based on the nature of the release action the successful execution of it depends on the orientation of the manipulated object with respect to the destination object frame. Therefore the reference frame is defined as a translated version of the frame of reference attached to the destination object. The predicates that cover the aspect of the orientation of an object are unary. The subset of orientation predicates is given in Figure 3(c). The full set of predicates also includes `ArbitraryOrientation` predicate, which is true when the orientation of the object ($\texttt{oo}_1$) in its argument is neither parallel to any axis nor to any plane formed by any two of the axes of the frame of reference concerned (10 predicates in general). With the help of described binary and unary predicates as well as Formula 3.1, the symbolic representation *sr* of configuration of two objects either `mo` and `do`) or `mo` and $\texttt{oo}_i$, $1 \leq i \leq$ is defined. The *true* predicates are then used to construct the corresponding symbolic representation *sr*. Since the RCC8 and interval algebra predicates are jointly exhaustive and pairwise

disjoint (JEPD), only one predicate for each projection can be evaluated as true. The final symbolic description is the conjunction of the corresponding symbolic representations: $sr(\texttt{mo}) \wedge sr(\texttt{mo,do}) \wedge sr(\texttt{mo,oo}_1) \wedge \ldots \wedge sr(\texttt{mo,oo}_n)$
The symbolic representations for the use cases visualised in Figure 1 are given below:

UCT:
NTPP-xy(mo,do)∧GT-z(mo,do)∧DC-yz(mo,do)∧D-x(mo,do)∧
DC-zx(mo,do)∧D-y(mo,do)∧DC-xy(mo,oo$_1$)∧O-z(mo,oo$_1$)∧
PO-yz(mo,oo$_1$)∧GT-x(mo,oo$_1$)∧DC-zx(mo,oo$_1$)∧
Oi-y(mo,oo$_1$)∧DC-xy(mo,oo$_2$)∧O-z(mo,oo$_2$)∧
PO-yz(mo,oo$_2$)∧LT-x(mo,oo$_2$)∧DC-zx(mo,oo$_2$)∧
O-y(mo,oo$_2$)∧StraightAlong-z(mo)

UCBB:
PO-xy(mo,do)∧D-z(mo,do)∧NTPP-yz(mo,do)∧D-x(mo,do)∧
PO-zx(mo,do)∧D-y(mo,do)∧DC-xy(mo,oo$_1$)∧D-z(mo,oo$_1$)∧
PO-yz(mo,oo$_1$)∧GT-x(mo,oo$_1$)∧DC-zx(mo,oo$_1$)∧Oi-y(mo,oo$_1$)∧
DC-xy(mo,oo$_2$)∧D-z(mo,oo$_2$)∧PO-yz(mo,oo$_2$)∧LT-x(mo,oo$_2$)∧
DC-zx(mo,oo$_2$)∧Oi-y(mo,oo$_2$)∧Parallel-yz(mo)

UCF:
NTPP-xy(mo,do)∧D-z(mo,do)∧NTPP-yz(mo,do)∧D-x(mo,do)∧
NTPP-zx(mo,do)∧D-y(mo,do)∧DC-xy(mo,oo$_1$)∧O-z(mo,oo$_1$)∧
PO-yz(mo,oo$_1$)∧GT-x(mo,oo$_1$)∧DC-zx(mo,oo$_1$)∧O-y(mo,oo$_1$)∧
DC-xy(mo,oo$_2$)∧D-z(mo,oo$_2$)∧PO-yz(mo,oo$_2$)∧LT-x(mo,oo$_2$)∧
DC-zx(mo,oo$_2$)∧Oi-y(mo,oo$_2$)∧StraightAlong-z(mo)

UCCT:
PO-xy(mo,do)∧GT-z(mo,do)∧DC-yz(mo,do)∧D-x(mo,do)∧
DC-zx(mo,do)∧Di-y(mo,do)∧StraightAlong-z(mo)

Now after establishing the symbolic representation and set of predicates the next step is to identify and evaluate how to learn such representation from the set of training examples.

5 Learning Symbolic Model

For the evaluation purpose, 5000 samples[2] for every use case given in Figure 1 are collected. Each sample is generated by randomly selecting an execution pose of the manipulated object around the regions where the actions could be successfully executed. A symbolic model is learned using only the positive training examples, namely 2065 examples in UCT, 2281 examples in UCBB, 1958 examples in UCF and 1423 training examples in the UCCT use case.

Then the symbolic action learning can be performed by various strategies such as finding a hypothesis that explains given training data, learning aspects of an action by analysing the repeated executions, or alternatively constantly updating the existing representation of an action by performing this action and observing its effects Although these approaches have a similar goal, they require different computational and design constraints. The hypothesis search (HS) [18] claims to find a solid hypothesis or a set of the hypotheses assuming that the training data set is sufficiently informative, but it is computationally expensive. The learning by demonstration (LbD) [19] techniques depend on informative replies from a teacher, while the online learning (OL) [20] methods require reliable observations and a sufficient amount of semantic knowledge. Furthermore, in addition to these three learning approaches the learning problem can also be formalised as an optimisation problem. The main advantages of this is computational and design efficiency. For this work the hill climbing (HC) [21] is selected as the optimisation technique. Since HC belongs to the family of local search algorithms, its result may depend on the initial estimation. Therefore for the better analysis of the algorithm's performance, it is run several times and the average of the fitness values of the precondition sets is calculated.

In this section all four learning methods are compared. Each is applied to the UCT, UCBB, UCF and UCCT use cases to learn a symbolic model of the actions. The target predicates are taken from the defined predicate pool. The expected symbolic descriptions of the initial state for each use case are created and presented in the previous section.

The symbolic learning objective is to find a symbolic description d' of objects which minimises a *cost function* c or maximises a *fitness function* f on a positive (D^+) and negative (D^-) training sets. In this work, the cost function c is defined as a function that counts the number of bit errors made by a possible symbolic description d' on a positive training set D^+.

$$c(d') = \sum_{i=1}^{k} \sum_{j=1}^{|PP|} |d'_j - D^+_{ij}| \tag{1}$$

k is the number of training examples and $|PP|$ is the cardinality of the *predicate pool* or the number of the predicates in it. D^+ is a binary matrix in which the entries of the i-th row are derived from a set of object relations that hold before the i-th execution. Instead of directly minimising the cost function c, a symbolic model is learnt by maximising the fitness function

$$f(d') = 1 - \frac{c(d')}{c_{\#}} \tag{2}$$

where $c_{\#} = k * |PP|$ is the total number of errors that d' can make on the training set. The range of the fitness function f is the interval [0,1]. In the case when the symbolic description d' generates no cost, the fitness function achieves the value '1', otherwise when d' is totally contradictory to the positive training data, the value of the fitness function is '0'.

To evaluate the learning paradigms (LbD, HS, OL and HC), the sets of the predicates are compared quantitatively and qualitatively. The quantitative comparison can be carried out with the help of a fitness function defined in Equation 2 and the qualitative comparison by analysing the expected and generated symbolic descriptions. The learning algorithms are also compared with regards to both noise-free and noisy data. The noisy data is created by adding two variations of Gaussian measurement noise: low (0.5cm position and 1 rotation noise) and high (2,5cm position and 5 rotation noise). According to the literature the OL should perform the least effectively in the noise case.

The following three Tables 1, 2 and 3 illustrate the consistency between the learnt or generated models and training data. In the noise-free case, the fitness of the predicate sets is relatively high that means the learnt models do not differ much from the expected outcome. The only OL algorithm has lower fitness values in the UCBB use case because this learnt model is more generally compared to the expected result. As can be seen from Table 2, the values of some fitness

[2] In our previous work [1] we show how to collect training example from a physical simulation.

Table 1: Fitness - noise-free

	LbD	OL	HC	HS
UCT	0.97	0.94	0.97	0.95
UCBB	0.97	0.91	0.97	0.95
UCF	0.98	0.94	0.98	0.97
UCCT	1.00	0.97	1.00	0.98

Table 2: Fitness - low noise

	LbD	OL	HC	HS
UCT	0.96	0.92	0.97	0.97
UCBB	0.96	0.90	0.97	0.96
UCF	0.98	0.93	0.98	0.98
UCCT	0.98	0.95	0.98	0.94

functions in the case of data with low noise decrease. Especially the fitness value for OL in all four use cases and for HS in the UCCT use case decrease because both algorithms produce an incorrect predicate set.

The results presented in the Table 3 show that the values of fitness decrease in all use cases for all considered learning techniques. It is noticeable that the fitness functions have the worst results in the UCBB and UCCT use cases.

In order to validate if the number of samples is sufficient to generate the optimal solution, we suggest comparing the generated solution with the solution produced using a larger number of samples. If these two solutions are similar, then this solution can be seen as optimal.

It should be mentioned that the symbolic learning is performed using only positively labeled data, therefore the actual number of examples used in the experiments is higher. Analysing the results, it can be concluded that the models are monotonically converging to the expected solution: i.e the model learned with 10 samples is the most general symbolic representation whereas each following model learned with a higher number of samples is a more detailed representation of it. Finally, the models learnt using 500 and more samples are equivalent to the expected model. Such monotonic convergence is anticipated for the noise-free case where the training set is correct. The results, received using the training dataset which includes low noise, show that even the model learned with 200 samples is as general as a model with 10 samples. Nevertheless, still starting with 500 samples, the model converges to the desired representation. The other situation is observed in the case of high noise. Here, the result representation with 2000 training examples is rather more general than the expected model. Moreover, the models learned with 10, 50 and 100 training examples are not monotonically converging, they all are similar general representations.

Based on the comparison of four symbolic learning algorithms in general and each learning paradigm in particular, it can be concluded that the HC algorithm produces accurate models learned using 500 or more training samples and is stable to noise in the training data. The evaluation results also showed that the LbD and HC can learn symbolic models in several seconds. Moreover, for both algorithms, the result models are close to the expected models even in the cases

Table 3: Fitness - high noise

	LbD	OL	HC	HS
UCT	0.96	0.92	0.96	0.95
UCBB	0.95	0.90	0.95	0.92
UCF	0.97	0.93	0.97	0.96
UCCT	0.96	0.91	0.96	0.90

with high noise. Since both algorithms provide good results as regards running time and learning performance, each of them can be the right choice for learning the symbolic model. In addition, the HC algorithm can use a previously learnt symbolic model as an initial estimate in order to learn a new representation. This property may speed up the learning process and reduces the size of training data. In the experiments with the geometric model as well as on the real and simulated robots HC for learning symbolic representation is used.

6 Conclusions and Future Work

It is true in general that an unwanted performance of an action can be reduced by using more knowledge about various environmental situations. The execution specific knowledge can be presented as a combination of qualitative and quantitative descriptions of the objects' states in a given environment where qualitative or symbolic representation can be used as a connection between abstract representation of planning operators and numeric information about releasing position.

The main goal of this symbolic model is abstract description of the relationships between objects. In order to find these descriptions a set of predicates is used. The main requirements placed on the functionality of symbolic description are generalisation to another environment configuration, support different manipulated objects and interaction with geometric models. In this work we use spatial relations between 3D objects which are described by RCC8 and interval algebra predicates. Moreover we analysed various learning paradigms by applying them on four various use cases and found that hill climbing optimisation techniques shows the best performers on the proposed predicate pool.

Since the learnt representation can be redundant, its refinement is important aspect for future work. The main advantage of the developed predicates is that they can easily represent large set of possible spatial relations between objects in an environment. Moreover, with the help of these predicates the spatial relations can be constructed at a different level of abstraction and can be effectually used to extract the numerical limits. However, sometimes these spatial relations include redundant predicates, for example the spatial relation object `oo1` is located on object `oo2` and can be correctly represented as:

1. $\texttt{On}(oo_1, oo_2) \iff \texttt{NTPP-xy}(oo_1, oo_2) \wedge \texttt{M-z}(oo_1, oo_2)$ or
2. $\texttt{On}(oo_1, oo_2) \iff \texttt{NTPP-xy}(oo_1, oo_2) \wedge \texttt{M-z}(oo_1, oo_2) \vee \texttt{DC-yz}(oo_1, oo_2) \wedge \texttt{D-x}(oo_1, oo_2) \vee \texttt{DC-zx}(oo_1, oo_2) \wedge \texttt{D-y}(oo_1, oo_2)$

Here, the second representation is redundant because the combination of predicates NTPP-xy and M-z includes the remaining combination of predicates (i.e. DC-yz(oo_1, oo_2) $\wedge$ D-x(oo_1, oo_2) $\vee$ DC-zx(oo_1, oo_2) $\wedge$ D-y(oo_1, oo_2)).
However, there are spatial relations which require more then two predicates for the correct representation, for example LeftOf or Behind. The main challenge with redundant predicates is the requirement for the larger training set for the symbolic learning.

Acknowledgments

The authors would like to thank B-IT Foundation for financial support and Alex Mitrevski for his technical support.

References

[1] 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 *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[2] A. Mitrevski, P. G. Plöger, and G. Lakemeyer. Representation and experience-based learning of explainable models for robot action execution. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5641–5647, 2020.

[3] R. Dearden and C. Burbridge. Manipulation planning using learned symbolic state abstractions. *Robotics and Autonomous Systems*, 62(3):355–365, 2014.

[4] N. Akhtar, A. Kuestenmacher, G. Lakemeyer, and P. G. Plöger. Simulation-based approach for avoiding external faults. In *International conference on advanced robotics (ICAR)*, 2013.

[5] B. Rosman and S. Ramamoorthy. Learning spatial relationships between objects. *I. J. Robotics Res.*, 30(11):1328–1342, 2011.

[6] K. Sjöö and P. Jensfelt. Learning spatial relations from functional simulation. In *IROS 2011*, September 2011.

[7] J. Kulick, M. Toussaint, T. Lang, and M. Lopes. Active learning for teaching a robot grounded relational symbols. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1451–1457. AAAI Press, 2013.

[8] M. J. Egenhofer. Topological relations in 3d. Technical report, University of Maine, USA, 1995.

[9] S. Zlatanova. On 3d topological relationships. In *Database and Expert Systems Applications, 2000. Proceedings. 11th International Workshop On*, pages 913–919. IEEE, 2000.

[10] J. Albath, J. L. Leopold, and A. M. Maglia. Rcc-3d: Qualitative spatial reasoning in 3d. *Spatial Cognition and Computation: An Interdisciplinary Journal*, August 2009.

[11] C. Sabharwal, J. Leopold, and N. Eloe. A More Expressive 3D Region Connection Calculus. In *Proceedings of the 2011 International Workshop on Visual Languages and Computing (in conjunction with the 17th International Conference on Distributed Multimedia Systems (DMS'11))*, pages 307–311, 2011.

[12] R. Mojtahedzadeh, A. Bouguerra, E. Schaffernicht, and A. J. Lilienthal. Support relation analysis and decision making for safe robotic manipulation tasks. *Robotics and Autonomous Systems*, 71:99–117, 2015.

[13] L. Kunze, K. K. Doreswamy, and N. Hawes. Using qualitative spatial relations for indirect object search. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 163–168, 2014.

[14] T. Frasca, B. Oosterveld, M. Chita-Tegmark, and M. Scheutz. Enabling fast instruction-based modification of learned robot skills. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(7):6075–6083, May 2021.

[15] J. K. Behrens, K. Stepanova, R. Lange, and R. Skoviera. Specifying dual-arm robot planning problems through natural language and demonstration. *IEEE Robotics and Automation Letters*, 4(3):2622–2629, 2019.

[16] M. Scheutz, E. A. Krause, B. Oosterveld, T. M. Frasca, and R. Platt. Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture. In Kate Larson, Michael Winikoff, Sanmay Das, and Edmund H. Durfee, editors, *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 1378–1386. ACM, 2017.

[17] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983.

[18] K. Mourão, L. Zettlemoyer, R. Petrick, and M. Steedman. Learning STRIPS operators from noisy and incomplete observations. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012*, pages 614–623, 2012.

[19] S. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, and D. G. Caldwell. Learning symbolic representations of actions from human demonstrations. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3801–3808, May 2015.

[20] X. Wang. Learning planning operators by observation and practice. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems, University of Chicago, Chicago, Illinois, USA, June 13-15, 1994*, pages 335–340, 1994.

[21] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.