Learning and Application of High-Level Concepts with Conceptual Spaces and PDDL

Richard Cubek and **Wolfgang Ertel**
Ravensburg-Weingarten University of Applied Sciences
88250 Weingarten, Germany
{richard.cubek,ertel}@hs-weingarten.de

Abstract

Robots should be able to learn skills from humans not only through kinesthetic teaching, but also by recognizing intentions and abstract concepts in human behavior. This work presents a method that enables robots to learn human-demonstrated concepts on an abstract, first-order logic based representational layer, while addressing the problems of keyframe extraction, concept learning, symbolic grounding and representation in PDDL.

1 INTRODUCTION

1.1 Robot Architectures and Control

Modelling of robot behavior and underlying software architectures emerged into several directions. *Behavior-based* systems are a network of interconnected units, that directly couple sensors to actuators. Each unit implements a specific behavior and depending on the situation specific behaviors can override others. Some behaviors use representations to a certain degree (Brooks 1985; Arkin 1998).

A counter example to behavior-based control is the early *sense-plan-act* paradigm, where based on its perception, the robot builds an abstract model of the world, generates a symbolic plan according to a long-range goal and then applies the plan (Nilsson 1984).

While behavior-based systems are well suited for rapidly changing environments of high stochasticity, they lack in their ability to achieve long range goals. Purely deliberative systems again are not able to deal with dynamic environments. The logical consequence was the appearance of hybrid control systems (*three-layer architectures*), integrating deliberation and reactivity at different layers (Firby 1989; Bonasso 1991). The reactive lower layer corresponds to behavior-based systems, where representations are described in a subsymbolic form (i.e. *meters*). The deliberative layer at the top relies on a symbolic formalism, often based on first-order predicate logic. The *sequencing* layer in the middle connects the deliberative and reactive parts while being responsible for tasks like invoking planning or translating high-level plans to low-level actions (Firby 1989).

Our framework is based on a three-layer paradigm. Basic operations are implemented on the lower layer in a hardware-specific way. Such basic operations are composed to so called *high-level* actions like *pick-object*, *place-object* or *move-to*, which again have a corresponding symbol on the deliberative layer. The underlying architecture now allows to represent goal-oriented plans as sequences of high-level actions since reactivity is covered at a lower layer. The representational gap between the layers, the *symbol grounding* problem (Harnad 1990) is a central question of this work.

1.2 Learning from Demonstration

Learning from demonstration (LfD) aims at robots learning skills being trained by humans. Most often it is based on the recognition of similarities among demonstrations. LfD can be classified into two approaches which are related to the earlier described levels of abstraction (Section 1.1): *trajectory encoding* for low-level representations of generic motions and *symbolic encoding* for high-level representations e.g. of sequences of predefined actions (Billard et al. 2008).

Trajectory encoding allows to describe arbitrary motor primitives and during training often requires direct motion of the robot's actuators by a human trainer (*kinesthetic teaching*). High-level learning requires a predefined set of functional low-level skills and often further prior knowledge. This allows the description of more abstract skills and goal oriented tasks. The presented framework belongs to this category of LfD.

1.3 Motivation

LfD-learned skills are evaluated by their generalization capabilities. To be more precise, the focus lies on their applicability to situations that differ from those during demonstration. The underlying idea now is to create a framework, that provides a robot with the ability to learn the goal behind a human-demonstrated task, to formalize this goal in a symbolic language and to apply a symbolic planner in order to reproduce the task in new situations. Thus, the approach fully complies to the generalization aspect. This basic idea was formulated in (Ekvall and Kragic 2008), which beside other related work will be compared to the presented framework in the next section.

2 RELATED WORK

In (Chella et al. 2004), a model from cognitive science, namely *conceptual spaces*, is proposed as a method to bridge the gap between symbolic and subsymbolic representations

in robotics. In this work, a formalized approach of connecting sensor data to symbols is shown. A similar grounding technique is used in a LfD framework in (Chella, Dindo, and Infantino 2006). Regarding observation, the focus lies on actions and their effects on objects and not on particular properties of motions. Observations are represented in two conceptual spaces, one to discover the type of objects and one to discover spatial relations between them. Learning of similarities among demonstrations is done by clustering in the conceptual spaces. Observed tasks are then encoded as sequences of actions on involved objects. There is no goal abstraction and a planner is not used. The similarity of a new situation and those from demonstrations is used to determine, which action sequence to execute.

The work in (Ekvall and Kragic 2008) was already mentioned in Section 1.3. It learns an abstract task goal from demonstration and describes it in a first-order based logical language. In new situations a symbolic planner is used to generate a plan that reproduces the task goal. The learning process is described as the detection of spatio-temporal constraints. Spatial constraints are learned by finding covariances in the position distances of different objects.

The framework presented in this paper is a combination of ideas based on (Chella, Dindo, and Infantino 2006) and (Ekvall and Kragic 2008). In contrast to the first, we define only one conceptual space where we then apply projections, contexts and conceptual prototypes (introduced in Section 3.2). We aim at creating a more general and formalized approach of learning different concepts from demonstration. Furthermore, we formalize recognized concepts to be applicable for planners. The difference to (Ekvall and Kragic 2008) is the use of conceptual spaces and the integration of object properties in recognized concepts. Furthermore, we do not learn temporal constraints, but rather let the planner take care of action orders based on its world model. Different to both works is the overall approach aiming to reach a concept abstraction as used in natural language, which is in terms of objects, their properties and relations among them.

A very sophisticated further LfD framework is presented in (Knoop, Pardowitz, and Dillmann 2007). Here, primitive movement types are recognized, extracted and abstracted during demonstration. Their pre- and postconditions and elementary operators are described in symbolic form, while symbolic planners can be used to generate sequential tasks. The abstract task knowledge has to be mapped to a target system. The described framework is supported by a whole range of sophisticated sensors. We simulate one vision system only and we concentrate less on action characteristics, but more on special events in key frames. Furthermore we dedicate more attention to the symbol grounding problem.

3 BACKGROUND

3.1 Symbolic Planner

The first planner designed for the purpose of robotics was *STRIPS* (Fikes and Nilsson 1971) where world-models, initial states, goals and actions with preconditions and effects can be defined. STRIPS is based on first-order logic. It was extended to the *Action Description Language* (ADL) (Pednault 1989), supporting conditional effects and quantified variables. Meanwhile, there is a whole research community dealing with the problem of planning. Therefore, a unique modelling language based on STRIPS and ADL has been introduced, the *Planning Domain Definition Language* (PDDL). We use the recent version PDDL2.1 because it supports ADL and the integration of fluents. Furthermore, a plan optimization metric can be defined. The reference specification of PDDL2.1 is (Fox and Long 2003). As symbolic planner, the presented framework uses *Metric-FF* (Hoffmann 2003), a top performing competitor from the *International Planning Competition.*

3.2 Conceptual Spaces

Representing Concepts: Conceptual spaces have been introduced by Gärdenfors as a mean for knowledge representation (Gärdenfors 2000). A conceptual space is built up from a set of quality dimensions within a geometric structure. A concept in a conceptual space is a convex region in that space, while a point in it (vector of quality dimension values or simply *knoxel*) is an instance of a concept. Using the euclidean distance as a metric in this space, conceptual spaces reduce the question of semantic similarity to the question of the euclidean distance between two points.

A knoxel can also define a prototype of a concept. This allows to decide about the membership of an arbitrary knoxel to a specific concept by the distance between that knoxel and the concept prototype. Or, the concept membership of a knoxel can be defined by the nearest concept prototype. The former defines a concept in a geometric form of a hypersphere, the latter causes a *voronoi tesselation* of concepts over the whole space.

Different contexts can be applied by assigning weights to dimensions. Let Q be a set of quality dimensions in a conceptual space, then the distance between two points p_1 and p_2 in a context k is:

$$dist(p_1, p_2, k) = \sqrt{\sum_{i=1}^{n} w_k^{(i)} (p_1^{(i)} - p_2^{(i)})^2} \qquad (1)$$

where $n = |Q|$, while w_k denotes the weight vector for the context k (Adams and Raubal 2009).

The most simple example of a conceptual space in (Gärdenfors 2000) is an one-dimensional space of time. Here, the point *now* divides the space into the concepts *past* and *future*. A further example is the color space of *hue*, *saturation* and *value* where the focal colors like *red* or *green* could be treated as certain color concepts, represented as convex regions in space. A *domain* is described as a a subspace of a conceptual space.

Symbolic grounding: Gärdenfors proposes the use of conceptual spaces also as an intermediate level between symbolic and subsymbolic representations. Concrete numeric values can be represented in single dimensions, while regions in the n-dimensional space can be bound to symbols. An example is the grounding of the term *red* used in natural language to a concrete vector of *hue*, *saturation* and *value*.

In fact, Gärdenfors treats the ability to recognize conceptual similarities as an important property of cognitive skills.

4 PROPOSED METHOD

This section explains the process of learning abstract concepts from demonstration and the transfer to a PDDL-based higher abstraction level.

4.1 Approach

As shown in (Ekvall and Kragic 2008), LfD on a symbolic abstraction layer enables the robot to learn task goals and to achieve them in new situations using a symbolic planner (provided that low-level skills are implemented). In our approach, a further objective is to achieve an abstraction of world and goal descriptions based on natural language, which is in terms of objects, their properties and relations between objects. The reason behind is that first, this enables the robot to learn and execute complex tasks as they are instructed among humans. Second, the effort to equip a robot with *a priori* knowledge or innate skills decreases with higher levels of abstraction.

4.2 High-Level Representation

World Model: The used world model is defined as a PDDL domain. It describes a simulated environment which is explained in detail in Section 4.4. It consists of several workdesks, while various objects are located at each of them. Every workdesk is very similar to a *blocks world*, an often used environment in high-level learning and planning experiments. The robots can move in the environment to approach the workdesks and they can pick objects and place them at arbitrary positions. Such skills (as *pick-object* or *move-to*) are represented as high-level skills. High-level skills refer to a set of low-level operations, implemented at lower layers. Their preconditions and effects can easily be defined manually. Moving between two workdesks produces higher costs than a pick or place operation. This causes the planner to generate optimal plans.

Object properties are formalized in the form of $p(x, c)$, p denoting the property predicate (e.g. $color$), x an object variable and c a concrete property constant (e.g. red). All kinds of spatial relations between objects are defined as certain bivalent predicates.

Learned concepts will be defined as PDDL goals. When applying to new situations, the initial PDDL state depends on the robot's perception. PDDL problem files are dynamically generated based on facts and goals.

4.3 Observations

The aim is to recognize and learn abstract concepts from demonstration. Since it is not possible to determine symbolic data directly from the vision system, the corresponding (subsymbolic) raw data has to be obtained first.

Relevant Parts of a Demonstration: The robot should recognize *what* was done instead of *how* it was done. Therefore, effects of actions are important (instead of kinematic properties of motions). We assume, that important effects occur at the end of an action.

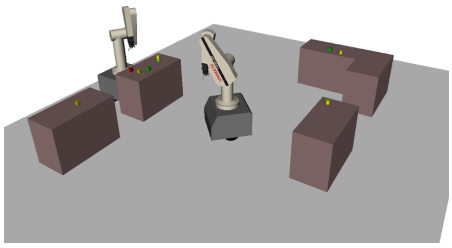

Figure 1: The virtual robot world.

Extraction of Relevant Data: The approach now is to recognize *object manipulation units* (OMUs), a series of vision frames, where the demonstrator manipulates a certain object. To analyze the effect of such a manipulation, the state at the first frame after an OMU is extracted. Such a frame is denoted as *key frame*. The states at the key frames are then used for further investigation.

Concentrating on spatial manipulations, we assume that an object is manipulated when it is moved. An object's motion is determined by the change of its position between two vision frames, which means by its velocity.

4.4 Simulated Environment

Experiments are realized within a virtual environment, created in *OpenRAVE*, a planning and simulation environment for robotics (Diankov 2010) (planning hereby means trajectory planning, not high-level planning as in PDDL). The virtual environment contains several workdesks, where again several objects are located on each of them. The robots can manipulate objects and move among the workdesks. There is a demonstrating and a learning robot, both consisting of a *Puma* robot arm, installed on a mobile platform (Figure 1). Several low-level skills are implemented on each robot, dealing with inverse kinematics, collision-free trajectories etc. The high-level behavior of the first robot is completely programmed manually, while the high-level behavior of the second is entirely learned by observing the first in demonstrations. Demonstrations are performed by manipulating objects at a workdesk.

The simulated robot vision is recognizing objects and their positions, delivering noisy position data in each of the dimensions x, y and z. The used error model is defined by

$$\varepsilon_j^{(i)} \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

where $i \in \{1, 2, 3\}$ and $j \in \{1, 2, ..., n\}$ for n processed frames, i and j denoting the indices of the dimensions and frames, respectively. The used standard deviation is $\sigma = 0.0033\ m$, resulting in 99% of the object positions oscillating $\pm 1cm$ in each dimension. That should lead to noisier data than delivered by most stable visions.

4.5 Keyframe Extraction

The setup of a demonstration is shown in Figure 2. The demonstrating robot is stacking the cuboid building blocks

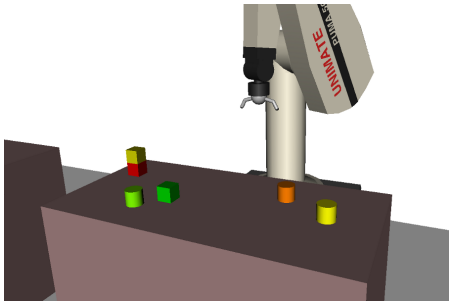

Figure 2: Setup at a certain workdesk in the environment with the demonstrating robot.

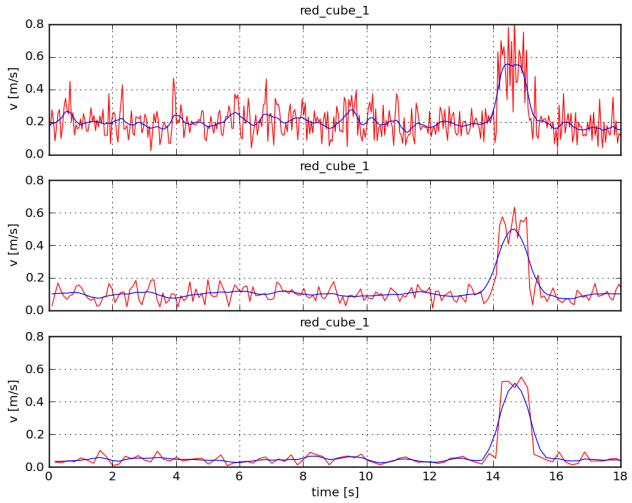

Figure 3: Velocity of the red cube over time (red) and locally weighted regression (blue). The framerates are 30 (top), 10 (middle) and 5 frames per second (bottom).

according to traffic lights: the yellow one on the green and the red one on the yellow. The positions of all blocks are recorded during the whole demonstration. Their velocities are calculated afterwards. Figure 3 shows the velocity of the red cube over time during demonstration. Due to the vision error model, it is very noisy. In order to smooth the data, the framerate can be decreased by skipping frames. Alternatively, *locally weighted regression* (LWR) can be applied (Atkeson, Moore, and Schaal 1996). Both is shown in Figure 3. In the experiments, the framerate is reduced to 5 frames per second. LWR is used additionally.

The noise floor causes the average velocity being > 0. A threshold of $0.07\ m/s$ is defined as a maximum value for noise floor (red areas in Figure 4), while all above is treated as object motion. Detected motion data is now clustered over time, resulting in clusters representing OMUs (green areas in Figure 4), wheras the first frame after an OMU is a key frame. For each key frame, the positions of *all* objects are stored in the observation data. The object positions are averaged over an adjustable amount of n frames, following a key frame.

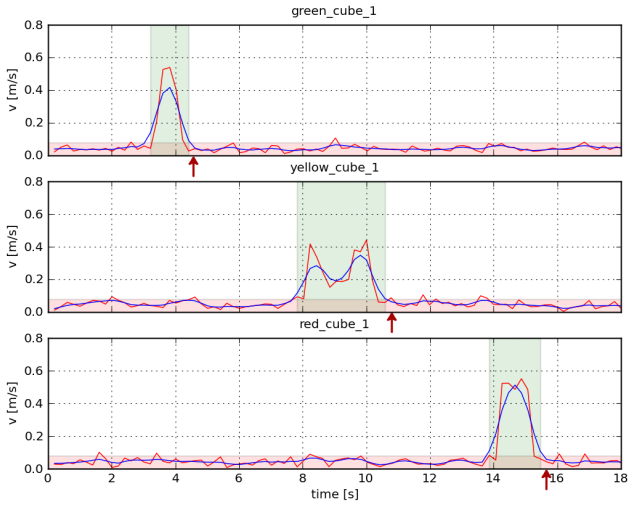

Figure 4: Velocities of the stacked objects over time. The red area is not considered, green areas show found clusters of motions (OMUs). Red arrows mark key frames.

4.6 Representing the Observation

The key frame data is treated as describing the action effects and thus the *key events* of the demonstration, which will be used to learn the underlying concept.

Event Representation in Conceptual Spaces: The learning part in the presented framework is based on the recognition of similarities among demonstrations. They will be determined by investigating the detected key events, considering the described level of abstraction in Section 4.1. Thus, a proper representational structure for the key events is needed. Conceptual spaces as intermediate level between symbolic and subsymbolic representations offer a suitable mean. Abstract concepts can be represented as regions in space, detecting conceptual similarities can be achieved by clustering knoxels in space. The dimensions in the conceptual space and the represenational aspect of a knoxel in it are yet to be explained.

4.7 Learning of Abstract Concepts

Regarding a key frame, we call the manipulated object the *source* object and the remaining objects the *target* objects. In Section 4.5, it was mentioned that at each key frame, positions of all objects are extracted, not only the position of the manipulated one. The reason behind is that the key frames are the moments where relations between objects are formed. Therefore, the detected key events from key frames are events between the manipulated object and the remaining objects. Such a key event will now be represented by a single knoxel in the conceptual space. Thus, a knoxel refers to an event between a source and a target object. This is illustrated in Figure 5. Assuming a scene at a workdesk with n objects, where one object is moved to another position (at the same workdesk). Here, the corresponding key frame from the end of the spatial manipulation generates $n - 1$ key events. One

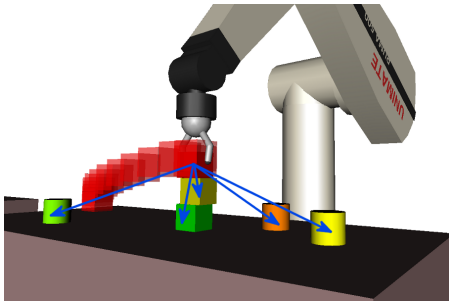

Figure 5: A key frame at the end of an object manipulation unit. The manipulated object (red cube) is the *source* object, the others are *target* objects. Each blue arrow is a key event, referring to a knoxel in conceptual space.

for each potential relation between the manipulated object and each remaining object on that workdesk.

We now define a conceptual space oriented on the kind of the described key events between a source and a target object. If the robotic system is able to deliver n properties for each observed object, then $2n+3$ dimensions are defined for the conceptual space. One for each source object property and one for each target object property ($2n$). Furthermore, one for the spatial relation in each dimension ($+3$) which means the relative position of the source to the target object. This will be the basis to learn in terms of objects, their properties and relations between objects.

If in a demonstration a red cube is put on a green cylinder twice, a cluster of two knoxels will occur. Each knoxel of this cluster will describe the key event between a red cube and a green cylinder. If in another demonstration a red object is put on a green object twice (both objects of different shapes), then a cluster can be found again, if we apply a specific projection on the conceptual space. This is the core idea of the presented work. If all detected key events between source and target objects within a demonstration are represented in the described conceptual space, then under a certain projection, conceptual similarities among key events will always build a cluster. Furthermore, every abstract concept class can be represented with a single projection matrix.

In the search for clusters, various projections have to be tried. Each projection combines only specific source and target object property dimensions. Hereby, the projections change from concrete to more general concepts. That means, that with the first search, the data is projected by an identity projection matrix. At each search step, rows in the projection matrix referring to object property dimensions are changed or removed until one or more clusters are found under a certain projection. All source/target object property combinations can be tried systematically. If n object properties are provided, 2^{2n} projections have to be tried. Thus, having m knoxels in the conceptual space using hierarchical clustering, the computational complexity is $O(2^{2n}m^3)$, which seems very high. In practice, the amount of provided object properties usually is low. Furthermore, in a demonstration of n objects and m OMUs, only $(n-1)m$ knoxels are produced. However, dimensionality reduction methods should be applied, if learning complex concepts on objects of many properties.

If the origin of a workdesk is treated as a virtual target object, absolute positions can be learned, too. This can be useful e.g. to teach a robot how to load objects on certain positions on a machine. For further processings of found clusters, their average knoxel is used, which is called *cluster knoxel*.

4.8 Relation Prototypes

We assume a demonstration where a green, yellow and red object are stacked, and a second demonstration repeating the first. This will generate three clusters. One of them represents the spatial relation between the red object at the top and the green one at the bottom, but we do not want this cluster to be considered for a concept. Furthermore, there can be different types of spatial relations, and the robot should distinguish between them. Therefore, conceptual prototypes (introduced in Section 3.2) can be used.

In the presented framework, a spatial relation is treated as a concept. Thus, it has its own representation in the conceptual space. In the simulated environment, object extents are known. Therefore, a further dimension is added to the perceptual space, describing the distance between the nearest points of source and target objects in z. This enables the definition of the relational prototype on. An object is on another object if their relational positions in x and y and further their distance in z all are about 0. This applies to the simulated world where all objects which the robot can manipulate have primitive shapes and similar sizes. In a more complex world this might not apply, but this does not affect the formalized method. In a more complex world, more detailed object descriptions, further dimensions and more complex prototypes might be required. Another possibility to be more precise in the definition of relation prototypes is to consider more object properties, for example the functional type of an object (e.g *building block*, *container* or *food*).

Since a prototype only considers a subspace, a domain can be defined for it. A domain has its own projection and further an own context (weight vector). For the prototype on, the dimensions for the relations in x and y might be weighted less then the one for the distance in z. Such a resulting prototype has the form of a hyper-ellipsoid.

In the example, each of the three found cluster knoxels is now checked regarding its membership to the prototype on in the corresponding domain. If a cluster knoxel is a member of the prototype (under the corresponding projection and weights), then it is considered for further processing. This will filter the detected relation between the red and the green object at the top and the bottom. On the other hand, a detected relation between an object and the workdesk origin should always be considered as concept.

4.9 Symbol Grounding

The PDDL world model is defined in advance, but we face the symbol grounding problem when recognized concepts (clusters) have to be formulated as PDDL goals. Furthermore, when an initial PDDL state has to be defined in a certain situation. Both have to be derived from sensor data. In

general, three grounding types are used. They are explained in the following.

Direct Mapping (Hash Table): Some object properties are known in advance. Often, they are already discrete and can directly be mapped to symbols in PDDL. This kind of grounding is used to define certain object property constants as the functional type. For example, the properties *building_block* or *container* are simply mapped to PDDL constants of the same name, which can then be used in the predicate *functional_type*.

Prototype Membership in Conceptual Space: This was explained in Section 4.8. In the framework, it is used to determine if a knoxel describes a specific spatial relation (between the source and the target object) in a relation domain (subspace). If so, the prototype name is directly mapped to a corresponding predicate. Such a relation concept has a natural form of a hyper-sphere (or a hyper-ellipsoid if the dimensions are weighted differently). A grounding with an if-else cascade over the same dimensions would have an unnatural, cuboid form. That would not be problematic for simple concepts, but it might be for more complex ones.

Nearest Prototype in Conceptual Space: This is similar to the prototype membership, but here, the membership of a knoxel to a concpt is defined by the nearest concept prototype. In the framework, it is used to define focal colors of objects from sensor data. The prototype name is directly mapped to a corresponding constant. For example, several variations of red all result in a single PDDL constant red.

4.10 Transfer to PDDL

Each found cluster of knoxels refers to a certain concept. Since the aim was to detect similarity clusters in terms of objects, their properties and relations between them, these concepts now have a certain structure. Concretely, each cluster refers to a certain bivalent relation between objects of certain properties, which can now be formalized. Let x and y be variables for source and target objects, P_s and P_t each a conjunction of predicates describing source object and target object properties, and R a bivalent spatial relation, then a realized concept can be described as a fact of the form:

$$\forall x \; \exists y \; P_s(x) \Rightarrow P_t(y) \wedge R(x, y) \tag{3}$$

whereas $P_s(x)$ and $P_t(y)$ are defined as

$$P_s(x) = p_1(x, c_{s1}) \wedge p_2(x, c_{s2}) \wedge ... \wedge p_n(x, c_{sn})$$
$$P_t(y) = p_1(y, c_{t1}) \wedge p_2(y, c_{t2}) \wedge ... \wedge p_n(y, c_{tn})$$

$p_1 ... p_n$ denoting object property predicates (e.g. $color$), whereas $\{c_{s1} ... c_{sn}\}$ and $\{c_{t1} ... c_{tn}\}$ refer to property constants (e.g. red) of source or target objects, respectively. Every found cluster in the conceptual space refers to such a concept. During the transfer of a knoxel cluster to PDDL, the projection matrix $\mathcal{P}$, under which the cluster was found is needed again. Each 1 from the elements of $\mathcal{P}$, which refers to a source or target object property dimension, activates a certain property predicate in $P_s(x)$ or $P_t(y)$, respectively. Activation means, that this property predicate will be present in the term derived from Formula 3. As an example, we consider stacking objects in a traffic lights color order, which results in two recognized concepts (which then have to be translated to PDDL goals):

$$\forall x \; \exists y \; color(x, yellow) \wedge color(y, green) \quad \Rightarrow on(x, y)$$
$$\forall x \; \exists y \; color(x, red) \wedge color(y, yellow) \quad \Rightarrow on(x, y)$$

The corresponding generated PDDL code would be:

```
(FORALL (?X) (EXISTS (?Y)
   (IMPLY (COLOR ?X YELLOW)
   (AND (COLOR ?Y GREEN) (ON ?X ?Y)))))
(FORALL (?X) (EXISTS (?Y)
   (IMPLY (COLOR ?X RED)
   (AND (COLOR ?Y YELLOW) (ON ?X ?Y)))))
```

Often, concepts are demonstrated concerning concrete objects, not objects of certain properties. In such cases, concepts can be learned over a property $instance\_of$, which describes an object of a specific, unique object class. Apart from that, quantifying over all target objects which fulfill $P_t(y)$ is not possible. Not every source object can form a spatial relation with every target object. If there is more than one target object, there would be no solution, therefore the existential quantifier.

In general, the presented method allows to learn concepts, as they usually are instructed in natural language, e.g. "put cuboid objects into box *A* and cylindric objects into box *B*".

Special Cases: However, Formula 3 does still not cover all constellations. Each target object usually has a *capacity* regarding its possible amount of relations with source objects. If there are more source objects than the sum of target objects capacities, there is no solution. For example, assuming a scene with ten red objects and two green pallets, each taking four objects. If the robot should put red objects into green pallets, using Formula 3 the robot will not find a solution. But a human would expect the robot to put at least eight objects into the pallets. Simply exchanging the quantifiers of x and y will cause the robot to put only one *red* object into each green pallet. In fact, the solution has to deal with target object capacities. We invent the functions $capacity(x)$ and $amount(x)$, $capacity(x)$ returns the number of relations a target object x can form with source objects. The number of relations a target object x already has formed with source objects in a current state is returned by $amount(x)$. PDDL2.1 allows the definition of such functions. In a PDDL problem file, a capacity can be set in the initial state. In preconditions of actions, which cause the forming of a relation the relations amount of the target object has to be smaller than its capacity. In the action effects, the relations amount has to be increased. A further predicate $equals(x, y)$ is used, it returns $true$, if $x = y$. Now, the problem concerning the pallets example can be solved. Using the terms from Formula 3 again, the alternative concept description is:

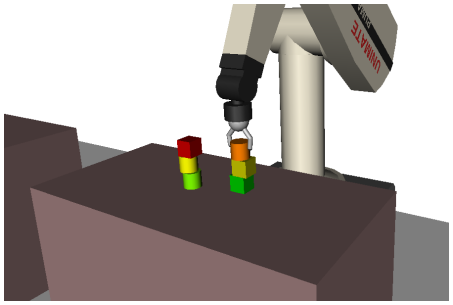

Figure 6: Demonstration: objects of arbitrary shapes are stacked as traffic lights.

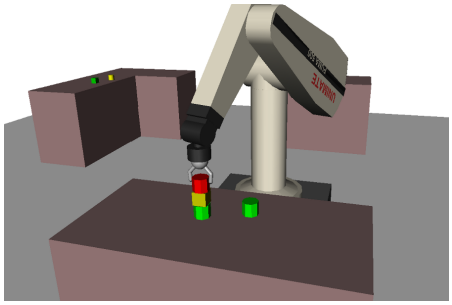

Figure 7: The learning robot applies the concept of stacking by colors on arbitrary objects.

$$\forall x\ \forall y\ P_t(y) \Rightarrow [equals(capacity(y), amount(y)) \quad (4)$$
$$\wedge\ (R(x,y) \Rightarrow P_s(x))]$$

The Formula is explained with regard to the pallets example: If a pallet is green then its capacity must be exhausted, and if there is an object that is in the pallet, it must be red. That will cause the robot to put eight red objects into the two green pallets. The overall concept formalization can now be defined as a disjunction of the formulas 3 and 4. This can be set as overall goal in the PDDL problem file.

Experiments with *Metric-FF* show that often this takes very long to find a plan. Probably, the planner remains searching in one of the two branches of the disjunction. This practical problem can be solved, if it is first determined, which formula has to be applied to find a plan.

A last special case is, when P_s and P_t are not distinguishable. In such a case, Formula 3 has to be extended by $R(x,y) \Leftrightarrow R(y,x)$.

5 EXPERIMENTS

5.1 Experiment 1

Demonstration: In the first experiment, the demonstrating robot is doing a demonstration of stacking objects according to traffic lights. It starts from the setup shown in Figure 2. The objects are stacked independent to their shape, only the color is considered. The last step of the demonstration is shown in Figure 6.

Learning and Reproduction: Regarding colors, the framework provides an own conceptual HSV color space of the three dimensions *hue*, *saturation* and *value* (brightness). Color symbols (color constants in PDDL) of observed objects are defined by the nearest color prototype in the color space. In this space only *red*, *green*, *blue* and *yellow* are defined as prototypes so far. That is why the rather orange cylinder from the demonstration is treated as red. The fact, that humans refer to colors more by the hue value than by saturation and brightness is considered in the weight vector of $[1, 0.2, 0.2]$. In the main conceptual space, source and target objects each have one discrete color dimension describing discretized colors obtained in the HSV color space. The properties $shape$ and $instance\_of$ are stored beforehand for each known object. While searching for concepts, under a projection which considers only source and target object colors (and of course relative positions) three clusters were found. Two of them fitted to the concept prototype of the relation on. Two ones from the projection matrix activated the predicate $color$ for the source and the target object. The corresponding color constants where obtained as already explained. From each knoxel cluster, one of the following concepts was derived:

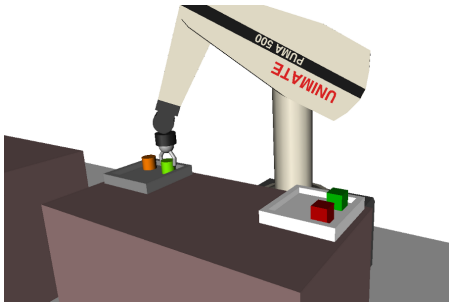

Figure 8: Demonstration: cubes are put into the bright pallet, cylinders into the dark.

$$\forall x\ \exists y\ color(x, yellow) \wedge color(y, green) \quad \Rightarrow on(x,y)$$
$$\forall x\ \exists y\ color(x, red) \wedge color(y, yellow) \quad \Rightarrow on(x,y)$$

The corresponding concept from Formula 4 is also considered. Applied in the virtual environment with new objects of prismatic shapes, the learning robot stacks them according to the learned concept (Figure 7).

5.2 Experiment 2

Demonstration: For the second experiment, two objects a and b being instances of the object classes $bright\_pallet$ and $dark\_pallet$, each with a capacity of four relations are added to the environment. Since we know object extents, three further spatial relation dimensions are added to the conceptual space, describing the spatial intersection of source and target object in each dimension. This data can simply be derived from the relative positions and the extents. A new relational concept prototype in is defined using the intersections. The last step of the demonstration is shown in Figure 8.

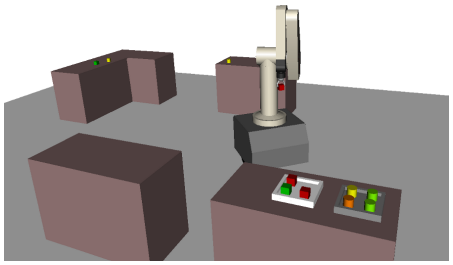

Figure 9: The learning robot applies the concept of sorting objects in specific pallets by shapes. Hereby, the robot also gets objects from other workdesks.

Learning and Reproduction: Under a projection, which considers the shape of the source object, the instance of the target object and the spatial intersections, two clusters were found. The clusters fit to the relational concept in. From each knoxel cluster, one of the following concepts was derived:

$$\begin{aligned}
&\forall x\ \exists y\ shape(x, cube) \\
&\qquad \wedge is_instance(y, bright_pallet) \quad \Rightarrow in(x, y) \\
&\forall x\ \exists y\ shape(x, cylinder) \\
&\qquad \wedge is_instance(y, dark_pallet) \quad \Rightarrow in(x, y)
\end{aligned}$$

Again, the corresponding concept from Formula 4 is also considered. Applying the concepts to new situations, the robot fills the specific pallets with cubes and cylinders correctly. Hereby, the robot also gets objects from other workdesks, if necessary (Figure 9).

6 CONCLUSIONS

The presented work reflects upon three insights. First, conceptual spaces are a proper mean for representing and learning abstract concepts. Furthermore, they are a proper solution to the symbolic grounding problem. Second, skills can be learned from demonstration at an abstraction level, that is similar to concepts as being described in natural language. This is in terms of objects, their properties and relations among them. Third, PDDL is a proper language to represent abstract concepts, while corresponding performant planners can be used to plan at the symbolic level.

7 ACKNOWLEDGMENTS

This work was supported by the Collaborative Center of Applied Research on Service Robotics (ZAFH Servicerobotik, `http://www.zafh-servicerobotik.de`).

References

Adams, B., and Raubal, M. 2009. A metric conceptual space algebra. In *Proceedings of the 9th international conference on Spatial information theory*, COSIT'09, 51–68. Berlin, Heidelberg: Springer-Verlag.

Arkin, R. C. 1998. *Behavior-Based Robotics*. MIT Press.

Atkeson, C. G.; Moore, A. W.; and Schaal, S. 1996. Locally weighted learning. *Artificial Intelligence Review* submitted.

Billard, A.; Calinon, S.; Dillmann, R.; and Schaal, S. 2008. Robot programming by demonstration. In Siciliano, B., and Khatib, O., eds., *Handbook of Robotics*. Springer. In press.

Bonasso, R. P. 1991. Integrating reaction plans and layered competences through synchronous control. In *Proceedings of the 12th international joint conference on Artificial intelligence - Volume 2*, 1225–1231. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Brooks, R. A. 1985. A robust layered control system for a mobile robot. Technical report, Cambridge, MA, USA.

Chella, A.; Coradeschi, S.; Frixione, M.; and Saffiotti, A. 2004. Perceptual anchoring via conceptual spaces. In *Proceedings of the AAAI-04 Workshop on Anchoring Symbols to Sensor Data, AAAI*. AAAI Press.

Chella, A.; Dindo, H.; and Infantino, I. 2006. Learning high-level tasks through imitation. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 9-15, 2006, Beijing, China*, 3648–3654.

Diankov, R. 2010. *Automated Construction of Robotic Manipulation Programs*. Ph.D. Dissertation, Carnegie Mellon University, Robotics Institute.

Ekvall, S., and Kragic, D. 2008. Robot learning from demonstration: A task-level planning approach. *International Journal on Advanced Robotics Systems* 5(3):223–234.

Fikes, R. E., and Nilsson, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3-4):189–208.

Firby, R. J. 1989. *Adaptive execution in complex dynamic worlds*. Ph.D. Dissertation, New Haven, CT, USA. AAI9010653.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)* 20:61–124.

Gärdenfors, P. 2000. *Conceptual Spaces: The Geometry of Thought*. Cambridge, MA, USA: MIT Press.

Harnad, S. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena* 42:335–346.

Hoffmann, J. 2003. The metric-ff planning system: translating 'ignoring delete lists' to numeric state variables. *Journal of Artificial Intelligence Research (JAIR)* 20:291–341.

Knoop, S.; Pardowitz, M.; and Dillmann, R. 2007. Automatic robot programming from learned abstract task knowledge. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 29 - November 2*, 1651–1657. IEEE.

Nilsson, N. J. 1984. Shakey the robot. Technical Report 323, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025.

Pednault, E. P. D. 1989. ADL: exploring the middle ground between strips and the situation calculus. In *Proceedings of the first international conference on Principles of knowledge representation and reasoning*, 324–332. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.