Learning Manipulation Actions from a Few Demonstrations

Nichola Abdo, Henrik Kretzschmar, Luciano Spinello, and Cyrill Stachniss

***Abstract*— To efficiently plan complex manipulation tasks, robots need to reason on a high level. Symbolic planning, however, requires knowledge about the preconditions and effects of the individual actions. In this work, we present a practical approach to learn manipulation skills, including preconditions and effects, based on teacher demonstrations. We believe that requiring only a small number of demonstrations is essential for robots operating in the real world. Therefore, our main focus and contribution is the ability to infer the preconditions and effects of actions based on a small number of demonstrations. Our system furthermore expresses the acquired manipulation actions as planning operators and is therefore able to use symbolic planners to solve new tasks. We implemented our approach on a PR2 robot and present real world manipulation experiments that illustrate that our system allows non-experts to transfer knowledge to robots.**

I. Introduction

Future service robots should have the ability to continuously adapt to our changing needs by easily acquiring new skills or generalizing their knowledge to new situations. However, it is infeasible to preprogram a robot to handle all possible scenarios in the real world before deployment. Instead, we need means that allow non-experts to interact with robots and teach them new actions efficiently—with as few demonstrations as possible. In addition to that, it is difficult to solve most real world manipulation tasks by reasoning purely in terms of low-level motions due to the high-dimensionality of the problems. Instead, robots should reason on a symbolic level and appropriately chain the learned actions to solve new tasks. Such a planning step, however, requires knowledge of the important preconditions and effects of the actions.

In this paper, we present an approach based on teaching by demonstration that allows a robot to learn the preconditions and effects of manipulation actions from a human teacher. A key focus is the ability to learn from a few demonstrations—generating large sets of training data is a nuisance to end-users. However, following the approach of requiring a few demonstrations only is unlikely to result in learning perfect models straightaway. Therefore, we complement the initial learning step with the concept of requesting additional inputs from the teacher. This is done whenever the current model is insufficient or the current situation is conflicting with previously-acquired knowledge. By following this practice, we greatly reduce the teaching overhead and eliminate the need for extensive simulations. Essentially, we give a practical solution to the problem of teaching manipulation actions to a robot in the absence of large amounts of training data.

Our method furthermore enables the robot to combine the learned actions by means of planning to solve new tasks that are more complicated than the learned individual actions. This work aims to bring real robotic systems and symbolic planning closer together by describing a way to learn actions that allows real robots to exploit the planners developed in the AI world. In this paper, we focus on a class of manipulation actions where the preconditions and goals depend on spatial or geometrical constraints between the involved objects. This includes a variety of pick-and-place actions, operating and opening doors, tidying up, etc. We implemented our approach and carried out real-world experiments using a PR2 robot to illustrate the capabilities and flexibility of our system. In brief, our system (i) estimates the important preconditions and effects of the actions by analyzing the state at the beginning and end of the demonstrations in a training phase, (ii) interacts with the teacher to correct potential false positive conditions that have been learned previously, and (iii) generates a symbolic representation of the actions based on their preconditions and effects and uses it to plan for solving new tasks.

All authors are with the University of Freiburg, Germany. This work has partly been supported by the EC under grant FP7-ICT-248258-First-MM and by the DFG through the Hybris-C1 project. We thank Malte Helmert for providing his implementation of the FD planner and Peter Pastor for making his DMP implementation available.

II. Related Work

Recently, imitation learning methods have become increasingly popular as means for transferring task knowledge to robots [4]. The key idea is to speed up the learning process by exploiting demonstrations given by a teacher. For instance, Bentivegna *et al.* [3] show a humanoid robot how to play air hockey by learning primitives that the robot can use in new situations. Asfour *et al.* [2] as well as Kulic *et al.* [14] use hidden Markov models to encode and reproduce demonstrated actions. Calinon and Billard [5] propose Gaussian mixture models to represent the variance over time in the demonstrated trajectories of a manipulator and exploit this information in the reproduction step. Similarly, Eppner *et al.* [7] and Mühlig *et al.* [17] consider the variance in the demonstrations to determine less relevant parts of the tasks. Our approach also inspects the variance among the demonstrations to determine relevance. However, as opposed to analyzing the trajectory of the manipulator, our method analyzes the variations in the state at the beginning and end of the demonstrations to identify the preconditions and effects of the individual actions.

Jäkel *et al.* [9] use demonstrations to generate a so-called strategy graph that segments tasks into sub-goals. The system generates temporal and spacial constraints for the transitions between the sub-goals, and an evolutionary algorithm is used to eliminate irrelevant constraints using a motion planner in simulation. Instead of simulation, our approach allows the robot to eliminate irrelevant conditions incrementally by requesting feedback from the teacher in case it encounters problems when executing a learned action. Exploiting teacher feedback has also been explored in the context of learning motion primitives from demonstrations, for example [15].

There are also a number of approaches that aim at teaching robots skills on a symbolic level for task planning based on teacher demonstrations. Veeraraghavan and Veloso [24] demonstrate sequences of actions to teach a robot a plan for solving sequential tasks that involve repetitions. They instantiate preprogrammed behaviors and then learn the corresponding preconditions and effects. Pardowitz *et al.* [20] extract task-specific knowledge from sequences of actions. The robot extracts the relevant elementary actions and task constraints from teacher demonstrations of pick-and-place actions when setting a table.

Ekvall and Kragic [6] provide a robot with demonstrations of tasks related to setting a table. The robot incrementally learns the constraints for each task with respect to the order of executing the actions. This knowledge is then used to choose the best strategy for solving a new task. To identify the different states observed during the demonstrations, they cluster the relative positions and orientations of the objects and inspect the cluster variances. Related to Ekvall and Kragic, our system applies k-means clustering to features to identify preconditions and effects of actions.

Many researchers adopt object action complexes (OACs), as presented by Krüge *et al.* [13], as a representation that combines low-level robot control and high-level planning. OACs consider objects as important to the robot in terms of the actions that can be applied to them. Pastor *et al.* [21] suggested adding a symbolic meaning to dynamic movement primitives (DMPs) learned from demonstrations, such that they can be used for high level planning in the context of OACs. However, this has not been realized in their work so far.

Kroemer and Peters [12] propose a framework for manipulation tasks that comprises two planning layers: a mid-level layer that optimizes the parameters of the motor primitives and a high-level layer that relies on preconditions and effects of the actions. However, the preconditions and effects are provided to the system and not learned. Chaining basic actions to perform more complex tasks has also been investigated in the reinforcement learning context [18], [11]. Recently, Niekum *et al.* [19] proposed an approach that segments unstructured demonstrations using the Beta Process Autoregressive Hidden Markov Model to identify and learn repeated skills. In contrast, our approach assumes a given segmentation and addresses the problem of extracting preconditions and effects for task planning. Also related to our work, Rudolph *et al.* [22] monitor the change in feature values to identify the effect of actions. Although the overall idea of monitoring changes is similar to our work, they apply a Bayesian network and concentrate on identifying the effects. This paper is an extension of [1], as this work allows for removing false positive constraints through interacting with the teacher.

III. Overview

This work aims at allowing a robot to learn different manipulation actions from a few teacher demonstrations and to then combine them in order to solve more complex tasks than the individual actions. To use actions in a planning system, their preconditions and effects need to be learned, which is the key focus of this work. An example for a precondition is the fact that an object must be grasped by the robot before placing it on top of another object. An example for an effect is the relative position of the two objects at the end of the motion. Identifying preconditions and effects is done by estimating the distribution of world states to recognize patterns while the teacher demonstrates the same action multiple times. These patterns are represented as logical predicates, allowing the robot to translate low-level sensory data into a high-level logical representation and verify when the preconditions or effects are satisfied. This enables the robot to use existing symbolic planners to solve more complex manipulation tasks. Furthermore, we introduce the possibility to incorporate additional teacher feedback to refine the learned conditions. Note that the segmentation of demonstrations into actions is not addressed in this work and is assumed to be provided by the teacher.

IV. State Representation and Kinesthetic Demonstrations

To compactly encode the state of the world, our system uses a set of features. This includes features describing properties of the objects such as their color and dimensions as well as their positions, distances, and orientations relative to each other. We also encode robot-specific features such as the opening of the endeffector and its pose with respect to objects in the scene. Furthermore, we use boolean or discrete features to capture a high-level, simplified, description of a complicated world state. For example, to indicate if an object is occluded (e.g., when another object is placed on it), we used a boolean feature that describes if the object is visible or not. When demonstrating how to open a door, we defined a discrete feature to describe the state of the door (closed, partially open, completely open) based on the measured position of the door edge. New features that may be needed when teaching other tasks can be added easily without affecting already learned actions.

When observing the demonstrations, our system considers only features related to objects that are close to the robot during the demonstrations (e.g. on a table in front of it). This allows us, for example, to ignore the state of a window, potentially in a different room, while the teacher shows the robot how to operate the door.

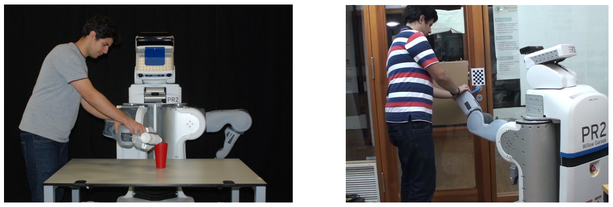

Fig. 1. Examples of kinesthetic training showing how to position a bottle above a cup and how to operate a door handle.

Note that we do not address the perception problem in this paper. We assume that the robot can identify relevant objects in the scene along with their poses. In our experiments, we attached checkerboard markers to the relevant objects and used an out-of-the-box detector available in the Robot Operating System (ROS). As an alternative to markers, we also used depth imaging from a Kinect camera for matching learned pointclouds of objects in the surrounding environment. For perception, we rely on existing technology and do not claim a novel contribution.

To teach the robot basic actions, we use kinesthetic training, i.e., the teacher moves the manipulators of the robot, as illustrated in Fig. 1. This method is rather accurate and does not require extra sensors since the robot can directly record the movements using its own encoders. We assume that the teacher always demonstrates the actions successfully. The trajectory of the end-effector in task space during the motion is encoded using the DMP formulation as proposed by Pastor *et al.* [21].

V. Identifying Preconditions and Effects

To identify the preconditions and effects of actions based on a set of demonstrations, we inspect the recorded feature values at the beginning and at the end of each demonstration. For each feature, we look for patterns in its values to decide whether or not it is important for an action.

In our approach, three questions have to be answered: First, which features are relevant for an action as a precondition or as an effect? Second, if a feature is regarded as relevant, which values are typically observed and how to derive a logical predicate that encodes the decision whether the precondition or effect is fulfilled? Third, given two logical predicates, how to decide whether both represent the same condition? The last question is essential for a planning system to chain actions together by verifying whether the effects of an action match the preconditions of another. The answers are provided in the remainder of this section.

A. *Expressing Preconditions and Effects Through Features*

In the most general case, the robot cannot be sure that an action can be carried out unless the current state of the world is identical to a state observed in one of the demonstrations. Otherwise, a precondition might not be satisfied and executing the action might fail. Such a requirement, however, is far too restrictive, and a smoothness assumption is needed to generalize to new but similar situations.

When deciding which features are relevant for the execution of an action, we make the assumption that the individual features are independent in terms of their relevance to the action (not in their values). Let $\mathcal{M}_f$ be a model for feature f that is learned using the observed values of this feature during the demonstrations. We do not pose any constraints on $\mathcal{M}_f$, except that we can compute the probability of a feature value v, i.e., $P(v \mid \mathcal{M}_f)$. In order to make the decision if a feature value v is *consistent with the model*, we use a gating threshold P_{min}, which is the minimal probability of v that we want to accept:

$$P(v \mid \mathcal{M}_f) > P_{min}. \tag{1}$$

One question here is how to compute the model $\mathcal{M}_f$ for each feature f and how to make the decision given in Eq. (1).

A central problem for learning a model in practice is that the robot only observes positive examples, i.e. successful demonstrations of the actions. Consider now a feature that is irrelevant for an action, i.e., the execution of the action is not affected no matter what value the feature takes. This in turn would require Eq. (1) to be always $true$, which can only be achieved if the full feature space is populated by samples. This is obviously infeasible as this would require a huge number of demonstrations. Therefore, we need a criterion in addition to Eq. (1) in order to determine if a feature is relevant or not.

B. *Estimating Preconditions and Effects by Analyzing the Variations in the Demonstrations*

We regard features as relevant only if they exhibit small variations in their values over the demonstrations. As preconditions, we consider features that take the same or similar values at the beginning of all demonstrations of an action. The same holds for the effects: features that always take similar values after having executed an action are considered to be an effect of that action. Informally speaking, for each action independently, we estimate for each feature the region in space that includes the samples corresponding to the demonstrations. By analyzing the size of such regions, we decide whether the feature is relevant.

For doing that, we require a measure of "data dispersion" or "variance" for $\mathcal{M}_f$. Let this measure be $H(\mathcal{M}_f)$; the smaller it is, the less spread the feature values are. Then, we impose a maximum value, H_{max}, so that we regard the feature f as relevant, i.e.

$$f \text{ is relevant} = \begin{cases} true & \text{if } H(\mathcal{M}_f) < H_{max} \\ false & \text{otherwise.} \end{cases} \tag{2}$$

Note that it is up to the designer to specify how $H(\cdot)$ is implemented. Possible realizations are the entropy or the variance in the feature values.

1) Features Taking Continuous Values: We have to define how to model $\mathcal{M}_f$ and $H(\cdot)$ for features taking continuous values. There exist multiple ways for estimating the boundaries of regions in a potentially high-dimensional space that are populated by samples. Approaches to one-class classification belong to this class of algorithms, e.g. single-class SVMs [23], k-means, PCA, and k-nearest neighbor [10] are all possible solutions.

In contrast to many other learning setups, we suffer from scarcity of training examples. A teacher provides a small number of demonstrations of an action, leading to a few sample points in feature space. Given a few training examples, applying techniques such as SVMs is likely to provide results that do not generalize well. Since we consider training sets in the order of ten sample points, we propose to apply a k-means clustering method.

As our function $H(\cdot)$, we consider the average squared intra-cluster distances considering each cluster c with mean μ_c, i.e.,

$$H(\mathcal{M}_f) = \frac{1}{KN} \sum_{k=1}^{K} \sum_{i=1}^{N_c} dist(v_i^c, \mu_c)^2, \tag{3}$$

where N is the overall number of demonstrations and K is the number of clusters. Furthermore, N_c is the number of data points assigned to cluster c and v_i^c is the value of the i^{th} data point in the cluster. The function $dist(\cdot)$ is a distance measure for the feature under consideration. This can either be the Euclidean distance between objects or the angular difference between rotations. Since the teacher demonstrates an unknown number of ways of performing an action, the system typically does not know the number of clusters K to look for in advance. We therefore perform multiple iterations of k-means with increasing values for K from 1 up to $\sqrt{N/2}$, where N is the number of data points. This upper limit is a heuristic suggested by Mardia *et al.* [16].

If the condition in Eq. (2) using this definition of H holds, then our system has identified a multi-modal pattern in the input data and considers this pattern as a precondition or effect. No further increase of K is then needed. However, if this criterion fails for all values of K, the system determines that the feature is irrelevant to the action since no pattern could be found.

For each relevant feature cluster related to an action, we then define a logical predicate $\mathcal{P}_f$, which is used by the symbolic planner later on to express when the condition represented by a cluster is satisfied. For a continuous feature, the predicate is defined as

$$\mathcal{P}_{f,c}(v) = \begin{cases} true & \text{if } dist(v, \mu_c) \leq d_{max} \\ false & \text{otherwise,} \end{cases} \tag{4}$$

where d_{max} is a threshold defining the maximum allowed distance to the centroid. The value of d_{max} is related to the variation that is allowed during the execution, the accuracy of the perception system, and the calibration of the robot's arms. In our current implementation, we set $d_{max} = 3\,\text{cm}$ for distances and $d_{max} = 20°$ for angles.

2) Features Taking Discrete Values: Besides features taking continuous values, we also consider features taking discrete values. An example of such a feature is object-is-visible, which can be *true* or *false*. To decide whether a discrete-valued feature is relevant for an action, we compute the entropy of the distribution of the feature values during the demonstrations. The entropy is a measure of uncertainty and is defined as

$$H(\mathcal{M}_f) = -\sum_{l=1}^{L} P(f = v_l) \log_2 P(f = v_l), \tag{5}$$

where $P(f = v_l)$ is the probability that the feature f takes the value v_l (out of L possible values), i.e., $\mathcal{M}_f$ is a histogram with L bins. The distribution over the values is computed based on the observations. A low entropy indicates that the probability mass is concentrated in a few feature values and thus indicates a low variation over the demonstrations.

The value that the feature has to take to satisfy the precondition or effect is then given by

$$v^f = \underset{v_l \in \{v_1 \ldots v_L\}}{\text{argmax}} \; P(f = v_l). \tag{6}$$

Similar to the continuous case, we can derive a boolean predicate $\mathcal{P}_f(v)$ that is later on used in the planning process to test whether a discrete precondition is fulfilled as:

$$\mathcal{P}_f(v) = \begin{cases} true & \text{if } v = v^f \\ false & \text{otherwise.} \end{cases} \tag{7}$$

C. Identifying Identical Predicates

To allow a planner to chain actions, we need to identify which predicates learned from one action are the same as the predicates from other actions. Consider two predicates $\mathcal{P}_f^{a_1}$ and $\mathcal{P}_f^{a_2}$ generated from two different actions a_1 and a_2 but from the same feature f. To decide if they represent the same condition, we consider the feature values from the demonstrations of a_1 and a_2 as a merged sample set. The predicates $\mathcal{P}_f^{a_1}$ and $\mathcal{P}_f^{a_2}$ are merged into one predicate if the merged sample set still fulfills the criterion given in Eq. (2). Otherwise, $\mathcal{P}_f^{a_1}$ and $\mathcal{P}_f^{a_2}$ remain individual predicates.

VI. Eliminating False Positives Through Interaction with the Teacher

The technique described above allows for identifying preconditions and effects based on the variations in demonstrations given by the teacher. If the teacher, however, does not provide enough variations in the demonstrations, the robot may learn false positive constraints that do not need to be fulfilled in reality when executing the action. Moreover, the larger the set of predefined features, the more demonstrations may be needed to obtain enough variations in order to eliminate all irrelevant features. For example, if the color of an object never varies during the demonstrations, the robot may learn the color as a relevant condition.

In this work, we explicitly incorporate the concept of teacher feedback to eliminate false positive conditions. Whenever the robot observes a starting state that contradicts the already-learned preconditions, it asks for a confirmation by the teacher that the action is in fact executable given the current state. The preconditions whose corresponding feature values are contradicted by that state are consequently removed from the learned action description.

Another problem can occur due to wrongly learned effects. To execute an action, the robot attempts to compute a pose of its end-effector that satisfies all the effects it learned. In new situations, the presence of false positive conditions can result in the robot's inability to compute a pose satisfying all constraints because in reality only a subset of them must be satisfied. For example, the robot may not be able to compute a pose of its end-effector that satisfies conflicting distance constraints to two objects where only the distance to one object is relevant. In such cases, the robot requests that the teacher illustrates how that case can be solved by providing a new demonstration. The system then records the feature values at the end of the new demonstration and evaluates which of the effect predicates are violated. Those are identified as false positive conditions and removed from the learned action description.

Note that our system distinguishes between the robot's inability to execute the action due to conflicting goal constraints and its inability to do so due to the failure of the motion planner to find a collision-free trajectory to the goal.

VII. Planning using the Acquired Actions

To outsource the planning task to an existing symbolic planner, our systems transforms the actions and learned predicates into the Planning Domain Definition Language (PDDL). For expressing each action in terms of its preconditions and effects, we consider the different possible cases for each relevant feature f. Since f could be relevant as a precondition, an effect, or both, our system adds the appropriate predicate, $\mathcal{P}_f$, or its negation in the preconditions or effects part of the PDDL operator. An example of a generated PDDL operator for approaching a block from the top to grasp it is:

```
(:action reachBlockTop
 :parameters (?b-block ?g-gripper)
 :precondition (and (visible ?b)
         (gripperOpen ?g))
 :effect (and (not (visible ?b))
         (gripperAroundBlockTop ?g ?b)))
```

The operator has been learned from demonstrating the reaching motion to the robot. The parameters part represents the typed variables `?b` and `?g` that are involved in the predicates. The types of objects are not learned but are provided by the perception system during the demonstrations. The terms in the precondition and effect parts correspond to learned predicates. Note that we replaced the automatically generated names such as predicate123 by meaningful ones. Furthermore, a few general physical constraints such as information about the robot's workspace need to be provided in PDDL. For more details, we refer the reader to [1].

For a new task, the system generates a PDDL problem description based on the current and goal states and uses the fast downward planner by Helmert [8] to generate a sequence of actions for the robot to execute. Each step in the plan corresponds to executing a learned action. To physically execute an action, our system first chooses one of the recorded movement primitives corresponding to it. The DMP is propagated using the new endpoints to produce a trajectory that is similar to the one demonstrated by the teacher, as is described in [1]. Before executing it, we simulate the arm trajectory and perform a collision check with any of the obstacles in the scene. In case no safe trajectory can be found, we use a geometric motion planner (from the arm navigation stack of ROS) as our fallback solution.

In the real world, a robot may not carry out all actions as expected or the environment may change. Therefore, we implemented a module that monitors plan execution, updates the truth values of the predicates from the observed feature values, and compares the current state to the expected effects of the actions. In case of a contradiction, the execution monitor triggers a replanning command using the current state of the world as a starting state. The fast downward planner is efficient enough to compute a new plan online so that the robot can proceed without significant interruptions.

VIII. Experiments

The goal of this evaluation is to show the capabilities and flexibility of our approach. We learn different actions such as: grasping blocks and bottles, pouring from bottles, and fairly complicated door opening actions. All components of the system have been implemented as ROS modules and our experiments are carried out with a real PR2 robot. Videos of the experiments can be found at: `http://www.informatik.uni-freiburg.de/%7Eabdon/videos/icra13/`

A. Training Phase

The first set of experiments is designed to illustrate how our system can reliably identify the important preconditions and effects of the actions after the initial training phase. The teaching is performed kinesthetically as illustrated in Fig. 1 and the initial training set consisted of 10 demonstrations per action. The robot recorded all features at the beginning and end of each demonstration as described in Sec. IV. The computations involved in learning the preconditions and effects from the recorded feature values varied between 0.1 s (15 features) and 0.8 s (64 features), depending on the number of features used in the indivdiual experiments.

1) Door Opening Scenario: In this set of experiments, we demonstrated to the robot how it can open a door using 5 simple actions: reaching the handle, turning the handle to unlatch the door, pulling the door from its handle, moving the gripper to the edge of the door, and pushing the door to open it completely. The position of the handle is detected via a checkerboard pattern and the opening angle of the edge of the door was measured using the robot's laser sensor.

As an example, the preconditions learned for pulling the door are: the door is partially open; the handle is visible; the gripper is closed and in a fixed (grasping) pose relative to it. For the effects: the handle is not visible after pulling the door; the door edge and the gripper relative to the robot are in a consistent position; the door is still partially open. For the sake of brevity, only a small subset of the results can be presented here.

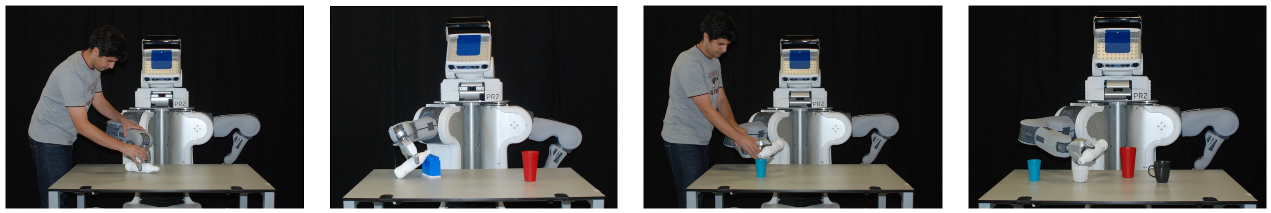

Fig. 2. Left: The robot requested a new demonstration after failing to grasp the bottle, which was lying on the table, due to a false-positive upright orientation condition. Second left: After learning that the angle of the bottle to the robot is irrelevant, the robot was able to grasp it from arbitrary initial orientations. Third left: The robot requested a demonstration for achieving a pouring pose with a cup of a different color and height than the one in the initial training. Right: After removing the false positive color and height conditions, it is able to repeat the action for different cup colors and sizes.

TABLE I

EXAMPLES OF RELEVANT CONDITIONS LEARNED FOR GRASPING A BOTTLE

feature	**bottle-robot position**	**bottle-robot orientation**	**gripper-bottle position**	**gripper-bottle orientation**	**gripper-robot position**	**gripper-robot orientation**	**bottle color**	**...**
preconditions	-	upright orientation	-	-	-	-	white	
effects	-	upright orientation	grasping pose	perpendicular	-	perpendicular	white	

TABLE II

LEARNING THE PRECONDITIONS AND EFFECTS FOR GRASPING A BLOCK

# demonstrations	5	6	9	10	> 10
success rate	17/20	19/20	19/20	20/20	20/20

2) Tabletop Scenario: We considered a set of actions involving multiple blocks, a bottle, and several cups placed on a table in front of the robot. The teacher demonstrated how to reach for an object to grasp it. The system correctly inferred that the pose of the object relative to the robot or the gripper are irrelevant as preconditions, whereas the pose of the gripper relative to the object is a relevant effect. For example, after reaching the bottle, the gripper is perpendicular to it and at a consistent distance from its center and top (see Tab. I).

The teacher also demonstrated how to place a grasped block on top of another one, and how to move a grasped bottle to a pouring pose. The pose of the gripper relative to the grasped object was learned as a relevant precondition and the corresponding object-to-object and gripper-to-object poses and distances at the goal were learned as effects. For the pouring example, this included the pouring angle between the bottle and the cup, and the relative translation and distance of the bottle to the top and center of the cup. Note that we learn the goal state of the pouring action and not the dynamics of pouring itself. Moreover, the robot learned consistent gripper openings in all experiments.

To provide a quantitative evaluation, we recorded 20 demonstrations for the action of reaching a block to grasp it. Instead of using all of them for learning, we sampled only a set and learned the preconditions and effects. We repeated this process 20 times and obtained the results shown in Tab. II. When using 10 or more demonstrations, the robot learned the action correctly. With less than 10 demonstrations, the system fails on average 1 to 3 out of 20 times, i.e., it learns at least one false positive precondition or effect. This is due to insufficient variations in the feature values in the demonstrations. For example, the robot learns a specific starting pose of the gripper as relevant to the action. Such conditions restrict the application of the action to scenes where this condition holds. However, this issue can be fixed through additional teacher feedback. This becomes also important if significantly larger feature sets are used.

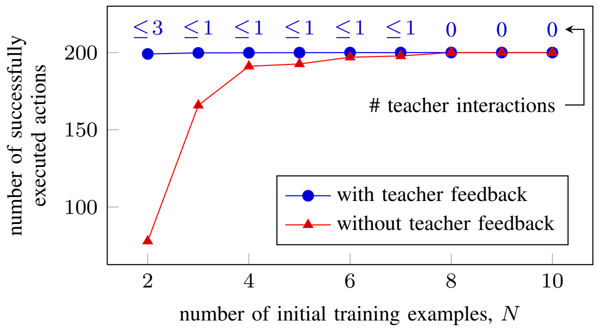

Fig. 4. Effect of teacher feedback evaluated on a task where the robot has to place a ball in a cup after learning this action from N initial demonstrations. In this experiment, we placed a second, irrelevant cup on the table, causing the system to learn false positive conditions related to its position in some situations. The red curve shows the number of successfully executed place-ball-in-cup tasks in 200 randomly sampled configurations of the scene. In contrast to that, the blue curve shows the number of successfully executed tasks when the robot was allowed to ask for feedback immediately after a failure. The number of such teacher interactions is given by the blue numbers. As the robot explicitly obtains feedback in situations it cannot solve, it can quickly remove the false positives and then solve the remaining test scenarios most of the time. Note that this experiment was conducted in simulation and the values are averaged over 50 independent runs.

B. Eliminating False Positive Conditions

This experiment is designed to illustrate the possibility to eliminate false positive conditions that resulted from too little variation in the training phase. For example, the teacher demonstrated to the robot how to grasp a bottle and how to move it to pour something into a cup. The bottle was always placed in an upright standing pose on the table, causing the robot to learn conditions describing the orientation of the bottle and the gripper when grasping it (see Tab. I).

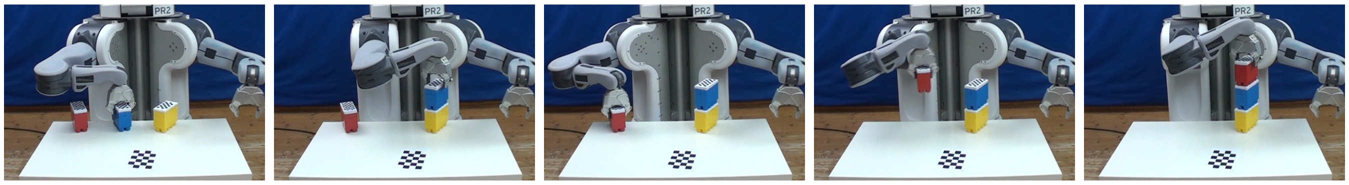

Fig. 3. The robot computes a plan to build a tower using the basic actions learned from the demonstrations.

When asked to execute the action while the bottle is lying on the table, the robot indicated that the learned preconditions are violated. The teacher confirmed that this is a valid starting state, hence removing the false positive orientation precondition. However, due to the conflicting effects describing the gripper-to-bottle and gripper-to-robot angles at the goal, the robot could not proceed, and requested a demonstration of how to perform this task. After showing the grasping pose for that case, the robot successfully eliminated the false positive angular constraints, and was able to repeatedly reach the bottle when lying down or placed at arbitrary angles on the table (Fig. 2, first two images).

A similar situation occurs when learning the pouring pose using the same cup in all demonstrations. After requesting a new demonstration for a different cup, the robot was able to eliminate false positive conditions related to the color and height features of the cup (see Fig. 2, last two images).

To further evaluate the effect of teacher interactions, we considered a task where the robot has to place a grasped ball in a cup. To induce the system to learn false positive conditions, we placed a second irrelevant cup on the table so that the relative positions to this cup were falsely identified as relevant in some situations. Training was done using 2-10 initial demonstrations with random placements of the two cups in a uniformly discretized rectangular region of $0.5\,\mathrm{m}$ by $0.9\,\mathrm{m}$. We then evaluated the robot's ability to repeat the action in 200 new configurations. The number of successfully executed actions is depicted by the red curve in Fig. 4. We then repeated the experiment and allowed the robot to request teacher feedback in case it cannot solve a given scenario, thereby incrementally eliminating false positive conditions it had learned initially. The results are given by the blue curve in Fig. 4. Both plots show the results averaged over 50 simulation runs. As can be seen from the plots, false positive conditions significantly limit the number of new scenarios the robot can solve without teacher feedback. In contrast to that, by allowing the robot to ask for help after a failure, it was able to eliminate the wrong conditions related to the second cup after at most three additional demonstrations.

C. Computing Plans to Solve New Tasks

This set of experiments is designed to show how the robot can use the learned actions to solve tasks that have not been demonstrated beforehand. In one experiment, we instructed the robot to stack three blocks in a specific order using the learned pick-and-place actions. The robot was always able to come up with a valid plan to the goal state. To execute each step, it selects a DMP learned during the demonstrations and used it to get to a desired gripper goal pose. Fig. 3 depicts the plan execution, and a video of this experiment is available under the URL above.

In another experiment, we placed the robot in front of the door and instructed it to keep it fully open while a human repeatedly closed it to different configurations. The robot came up with three different plans depending on the starting state. If the door is completely closed, our robot needs to carry out the following actions: reachHandle; graspHandle; turnHandle; pullDoor; releaseHandle; moveArmToInnerSide; pushDoor. If the door latch was not locked, it is sufficient to execute: reachHandle; graspHandle; pullDoor; releaseHandle; moveArmToInnerSide; pushDoor. If the door is already partially open but the robot does not see the handle, the plan was: moveArmToInnerSide; pushDoor. Some of these actions are visible in parts of Fig. 5. Further images had to be omitted due to limited space but the reader may consider the video showing parts of this experiment. We conducted this experiment for more than $20\,\mathrm{min}$. Due to errors in estimating the handle or door edge position, the gripper sometimes failed to grasp the handle or push the door successfully. However, the system was always able to detect unsatisfied action effects and recover by computing a new plan.

D. Reacting to Unexpected Changes in the Environment

The last experiment was designed to illustrate how the robot can deal with unexpected changes in the environment while executing plans. For the purposes of this experiment, motion commands for navigating between a door and a table were manually provided. The goal was to take two blocks and bring them to the corridor outside the room. Initially, both blocks lay on a table in the room and the door is open. While the robot picks up the two blocks with its manipulators, a person closes the door. After detecting the change, the robot computes a new plan and decides to bring one block back to the table to free one gripper. It then opens the door with the free hand, moves back to the table, picks up the block again, and finally brings both blocks outside the room. Pictures from this experiment are shown in Fig. 5.

IX. Conclusion

We addressed the problem of learning manipulation actions based on a small number of teacher demonstrations. Our method infers the preconditions and effects that need to be satisfied when applying an action directly from the demonstrations and represents them symbolically using logical predicates. Our system furthermore allows the robot to incrementally eliminate false positive conditions it may have learned by requesting teacher feedback as soon as it

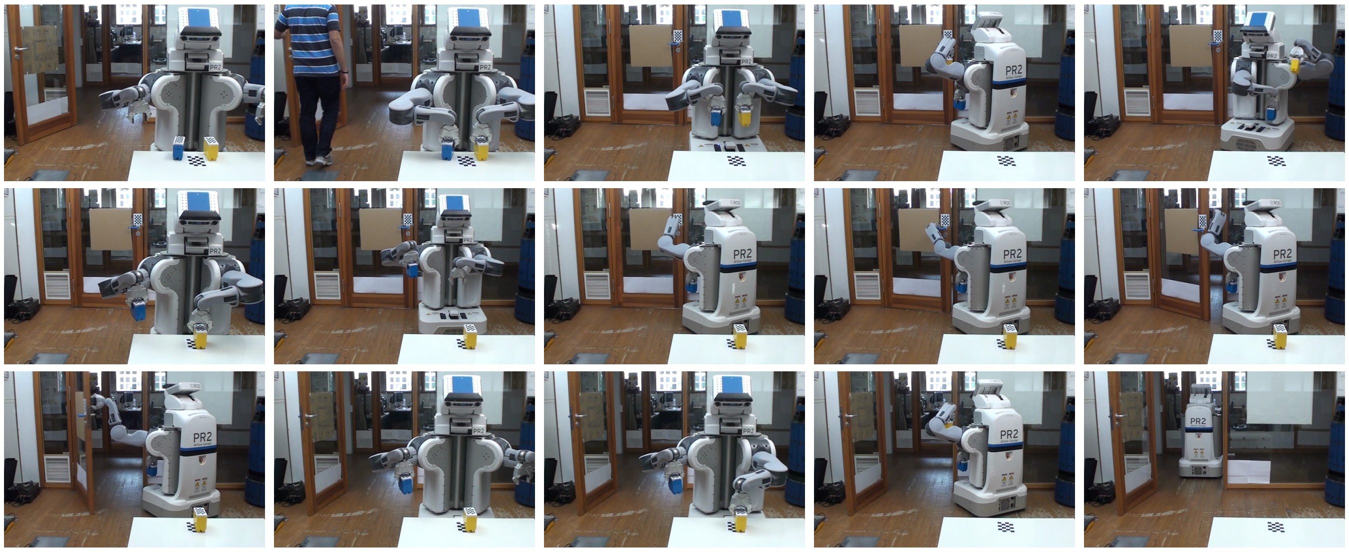

Fig. 5. The robot computes a plan to grasp the two blocks and to bring them outside. Once the robot has grasped the blocks, a person closes the door. After having detected that, the robot computes a new plan and clears its left gripper by first going back to the table and placing the yellow block there. The robot then moves back to open the door and then to the table to regrasp the yellow block. Finally, the robot leaves the room with both blocks and reaches the goal state. See also `http://www.informatik.uni-freiburg.de/%7Eabdon/videos/icra13/` for a video of this experiment.

encounters situations in cannot solve. Moreover, the symbolic representation allows the robot to use state-of-the-art planners to combine the learned actions to solve new tasks. We implemented our approach and presented experiments using a real PR2 robot to illustrate the capabilities and flexibility of our system.

Despite these encouraging results, there is space for further improvements. We are currently investigating learning probabilistic models of the actions to cope with perception noise, imperfect demonstrations, and manually set thresholds. Moreover, it would be interesting to extend the list of considered features and investigate learning more complex actions and incorporating more informative teacher feedback.

References

[1] N. Abdo, H. Kretzschmar, and C. Stachniss. From low-level trajectory demonstrations to symbolic actions for planning. In *ICAPS Workshop on Combining Task and Motion Planning for Real-World App.*, 2012.

[2] T. Asfour, F. Gyarfas, P. Azad, and R. Dillmann. Imitation learning of dual-arm manipulation tasks in humanoid robots. In *Int. Conf. on Humanoid Robots*, 2006.

[3] D.C. Bentivegna, C.G. Atkeson, A. Ude, and G. Cheng. Learning to act from observation and practice. *Int. J. of Humanoid Robotics*, 2004.

[4] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Robot programming by demonstration. In B. Siciliano and O. Khatib, editors, *Handbook of Robotics*. Springer, 2008.

[5] S. Calinon and A. Billard. A probabilistic programming by demonstration framework handling skill constraints in joint space and task space. In *Int. Conf. on Intelligent Robots and Systems*, 2008.

[6] S. Ekvall and D. Kragic. Learning task models from multiple human demonstrations. In *Intl. Symposium on Robot and Human Interactive Communication*, pages 358–363, 2006.

[7] C. Eppner, J. Sturm, M. Bennewitz, C. Stachniss, and W. Burgard. Imitation learning with generalized task descriptions. In *Int. Conf. on Robotics & Automation*, 2009.

[8] M. Helmert. The fast downward planning system. *Journal on AI Research*, 26, 2006.

[9] R. Jäkel, P. Meissner, S. R. Schmidt-Rohr, and R. Dillmann. Distributed generalization of learned planning models in robot programming by demonstration. In *Int. Conf. on Intelligent Robots and Systems*, 2011.

[10] K. Kennedy, B. Mac Namee, and S.J. Delany. Learning without default: A study of one-class classification and the low-default portfolio problem. In *Conf. on AI and Cognitive Science*, 2009.

[11] G. Konidaris and A. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Conf. on Neural Information Processing Systems*, 2009.

[12] O. Kroemer and J. Peters. A flexible hybrid framework for modeling complex manipulation tasks. In *Int. Conf. on Robotics & Automation*, 2011.

[13] N. Krüger, J. Piater, F. Wörgötter, C. Geib, R. Petrick, M. Steedman, A. Ude, T. Asfour, D. Kraft, D. Omrcen, B. Hommel, A. Agostino, D. Kragic, J. Eklundh, V. Krüger, and R. Dillmann. Formal definition of object action complexes and examples at different levels of the process hierarchy. Technical report, 2009.

[14] D. Kulic, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *Int. J. of Robotics Research*, 31(3), 2012.

[15] D. Lee and C. Ott. Incremental kinesthetic teaching of motion primitives using the motion refinement tube. *Autonomous Robots*, 31(2-3), 2011.

[16] K.V. Mardia, J.T. Kent, and J.M. Bibby. *Multivariate Analysis*. Academic press, 1979.

[17] M. Mühlig, M. Gienger, S. Hellbach, J.J. Steil, and C. Goerik. Task-level imitation learning using variance-based movement optimization. In *Int. Conf. on Robotics & Automation*, 2009.

[18] Gerhard Neumann, Wolfgang Maass, and Jan Peters. Learning complex motions by sequencing simpler motion templates. In *Int. Conf. on Machine Learning*, 2009.

[19] S. Niekum, S. Osentoski, G. Konidaris, and A. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *Int. Conf. on Intelligent Robots and Systems*, 2012.

[20] M. Pardowitz, R. Zöllner, and R. Dillmann. Incremental acquisition of task knowledge applying heuristic relevance estimation. In *Int. Conf. on Robotics & Automation*, 2006.

[21] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *Int. Conf. on Robotics & Automation*, 2009.

[22] M. Rudolph, M. Mühlig, M. Gienger, and H.-J. Böhme. Learning the consequences of actions: Representing effects as feature changes. In *Int. Conf. on Emerging Security Technologies (EST)*, 2010.

[23] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson. Estimating the support of a high-dimensional distribution. Technical report, Microsoft Research, TR87, 2000.

[24] H. Veeraraghavan and M. Veloso. Teaching sequential tasks with repetition through demonstration. In *Int. Conf. on Autonomous Agents and Multiagent Systems*, 2008.