Project Acronym:	Xperience
Project Type:	IP
Project Title:	Robots Bootstrapped through Learning from Experience
Contract Number:	270273
Starting Date:	01-01-2011
Ending Date:	31-12-2015

Deliverable Number:	D1.2.2
Deliverable Title:	Validation and execution of defined benchmarks
Type (Internal, Restricted, Public):	PU
Authors:	G. Metta, S. Fichtl, D. Kraft, N. Krüger, A. Gams, T. Petrič, B. Nemec, A. Ude, E. Ugur, S. Szedmak, J. Piater, T. Asfour
Contributing Partners:	ALL

Contractual Date of Delivery to the EC:	31.12.2015
Actual Date of Delivery to the EC:	12.02.2016

Contents

Chapter 1

Executive Summary

Xperience project focuses on the performance of the learning process. In this context we need to benchmark by how much and how quickly the task knowledge and robot skills improve as the amount of data available to the learning agent increases. This document focused on the analysis of numerical methods and their implementation as numerical software in order to measure the impact of structural bootstrapping applied across all levels of the system architecture. Being able to test the correctness of the implementations as well as the performances of the methods is of significant importance. Furthermore, the validation and benchmark is one of the major steps in the construction quality of open source software, in particular if they are be used to transfer our findings in the community and employed in practical applications.

In summary, this deliverable places a great deal of importance in benchmarking, in terms of metrics, that demonstrate and measure the impact of structural bootstrapping applied across all levels of the architecture while also focusing on key performance indicators as benchmarks for learning, defined in D1.1.1, such as the following, to name a few:

- **Quality of learning**: How good is the acquired task knowledge? How much better is the task knowledge after coaching compared to standard imitation learning? (this is task specific)
- **Efficiency of learning**: How many training examples are needed to achieve satisfactory performance? while also looking at the reduction of the number of training examples.
- **Stability of learning**: Standard deviation of the number of training examples for learning of a specific task.
- **Speed of acquisition**: How many trials do we need to add information about new objects to the library of known objects.
- **Efficiency of generalization**: How many specific objects belonging to a particular class do we need before we can apply the acquired knowledge to all objects of this class?

In particular we show throughout this document how bootstrapping greatly influences the learning speed, increases the prediction and motor accuracy and remarkably reduces the need for large training samples. This increase of performance is seen in action generation, action coding, visual recognition and Object-Action Complexes and we are able to obtain better generalization, and engender complex behaviors as those needed for interacting with other cognitive agents.

Chapter 2

Validation and execution of defined benchmarks

2.1 Sensorimotor Level

2.1.1 Transfer of manipulation predictions between actions

This section will discuss the bootstrapping results we were able to achieve in learning visual success predicting classifiers (see Section 2.1.1.1). Section 2.1.1.2 will give our learning approach without bootstrapping and our experimental setup. The following section (2.1.1.3) will show the results we are able to achieve without bootstrapping. Section 2.1.1.4 describes the approach used to bootstrap and the results of bootstrapping are shown and discussed in Sections 2.1.1.5 and 2.1.1.6. The work summarised in this section has lead to a more detailed description which can be found in [FKKG15].

2.1.1.1 Problem description

The goal of the visual success classifiers trained in this work is to accurately predict — based on visual data — whether an action can be executed successfully in a given scene. The actions are means-end actions where one action (means) is executed on one object in order to facilitate the successful execution of another action (end) on another object. It is, therefore, the spatial relationship between the objects involved, that is determining whether the means action can succeed to facilitate the end action.

An example of an action used in this work is the ‘slide’ action. The goal of the ‘slide’ action is to grasp and tilt one object with another object on-top of it, so that the object on-top slides off the bottom object. In case there is no object on-top (it might e.g., be beside) the action will fail. An successful ‘slide’ action execution is illustrated in Figure 2.1.

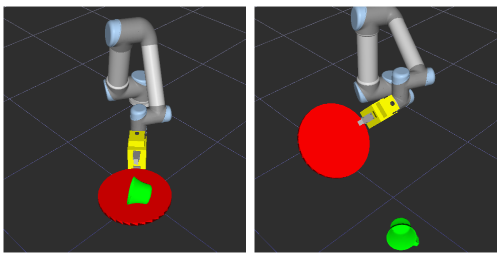


Figure 2.1: Before/after action execution snapshots to Illustrate the ‘slide’ action.

2.1.1.2 General learning approach

The robot experiments used in this work were performed using a six degrees of freedom robot arm in a physically realistic simulated robotics setup including a simulated Kinect sensor. Using colour based segmentation we extracted a point cloud for every object in the scene. Using Eigen decomposition over the segmented point clouds we extract nine variables as approximations of an object's position (X, Y, Z), orientation (Roll, Pitch, Yaw) and size (elongation along each object axis).

In our experiments, we used an overall set of 29 objects, which can be split into four different groups: (1) Toys, (2) Bases, (3) Obstacles and (4) Rakes. For every experiment exactly one toy object and one object of a different group is used. The objects are randomly distributed within the workspace area that approximately forms a semi-circle in front of the robot with a radius of 1.8 m. The robot is equipped with nine actions it can perform. Depending on the two objects in the workspace, one of which always is a toy object, different actions are available for execution.

Based on the visually derived variables (nine per object) describing each of the objects, a random forest classifier is trained to predict the success of an action. This approach is illustrated by Figure 2.2.

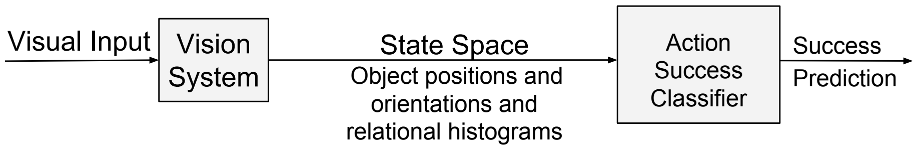

Visual Input
Vision System
State Space
Object positions and orientations and relational histograms
Action Success Classifier
Success Prediction

Figure 2.2: Illustration of the inputs for learning action success Predictors.

2.1.1.3 Results without bootstrapping

Figure 2.3 shows the learning curve for learning nine action success predictors without bootstrapping.

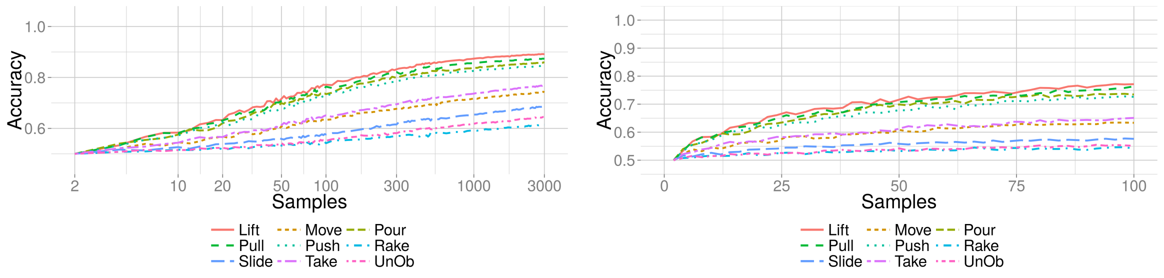


Figure 2.3: Illustration of the learning curve of action success predictor learning for nine actions without bootstrapping. On the left, the X-axis is logarithmic scaled to highlight that the learning continues even after several thousand training samples. On the right, the plot highlights the early stage of learning with up to 100 samples.

The X-axes in Figure 2.3 show the number of samples used for training and the Y-axes show the accuracy of the action success classifiers. The training data is always 50% positive and 50% negative samples. It can be seen by the left part of Figure 2.3, that learning without bootstrapping continues even after several thousand training samples are used. The right part of Figure 2.3 highlights the early stage of learning with up to 100 samples.

2.1.1.4 Bootstrapping approach

We investigated how additional knowledge can be used to bootstrap the learning of these action success classifiers. The agent stores its Knowledge in form of random forest based predictors:

- **Action Success Predictor**: The classifiers that predict the success of actions.

- **Category Predictor**: Classifiers that recognise specific categorical patterns in the environment (see [FKKG15]).

Bootstrapping is achieved by adding the output of existing predictors to the state space of a new action success classifier that is learnt to predict the success of a new action. Figure 2.4 illustrates our general structure of learning. We investigate the following approaches to bootstrapping of action success classifiers:

A) using previously learned action success predictor(s) for other actions as additional input

C) using previously learned category predictor(s) as additional input

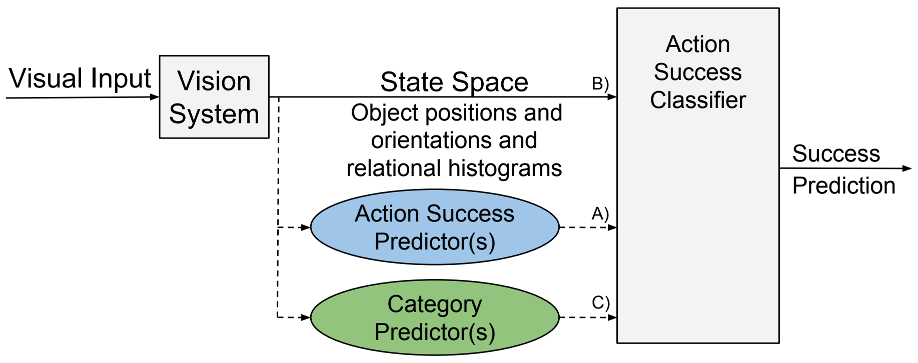

Visual Input
Vision System
State Space
Object positions and orientations and relational histograms
B)
Action Success Predictor(s)
A)
Category Predictor(s)
C)
Action Success Classifier
Success Prediction

Figure 2.4: Illustration of the inputs for learning action success Predictors. The solid lines represent the basic case of learning with no additional information. To bootstrap the learning additional information can be added. The blue ellipse illustrates the addition of one or all other existing actions' success predictions. The green ellipse illustrates the addition of one or all previously learned category predictor outputs.

Firstly, we analyse the bootstrapping effect by comparing the learning curves of learning with and without bootstrapping. Secondly, we benchmark the bootstrapping effect by using a novel metric that we introduced in [FKKG15] to quantify the bootstrapping effect.

2.1.1.5 Results with bootstrapping

The following bootstrapping result figures show on the left the learning curve of the action success predictors (comparable with Figure 2.3 left) and on the right the bootstrap factor that quantifies the bootstrapping performance.

A) Figure 2.5 shows the learning curve for learning the nine action success predictors bootstrapped with the prediction of another action success predictor that leads to the best learning results[1]. Figure 2.6 shows the learning curve for learning the nine action success predictors bootstrapped with the predictions of all other action success predictors at once.

C) Figure 2.7 shows the learning curve for learning the nine action success predictors bootstrapped with the prediction of the one category predictor that leads to the best learning results. Figure 2.8 shows the learning curve for learning the nine action success predictors bootstrapped with the predictions of all category predictors at once.

2.1.1.6 Discussion

By comparing the results of learning without bootstrapping in Figure 2.3 with the results of learning with bootstrapping in Figures 2.5 to 2.8, the increase in learning speed achievable through bootstrapping becomes evident. The Bootstrap factors that accompany the learning curve results of learning with

[1]This best result is the one with the highest average accuracy learning curve from two to 100 samples (see [FKKG15] for details).

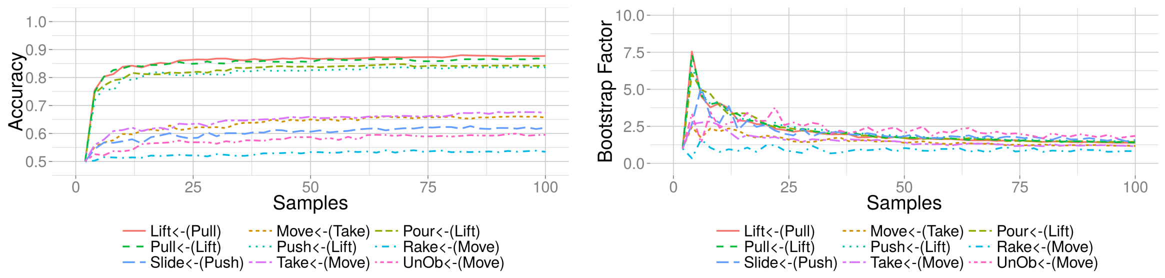


Figure 2.5: Illustration of action success predictor learning for nine actions using another action success predictor output for bootstrapping. Of all the potential action success prediction sources for bootstrapping, for each action, only the best result is presented.

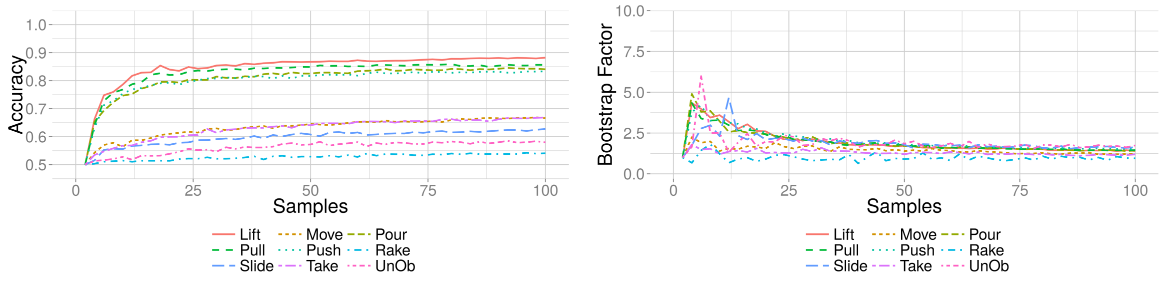


Figure 2.6: Illustration of action success predictor learning for nine actions using all other action success predictor outputs at once.

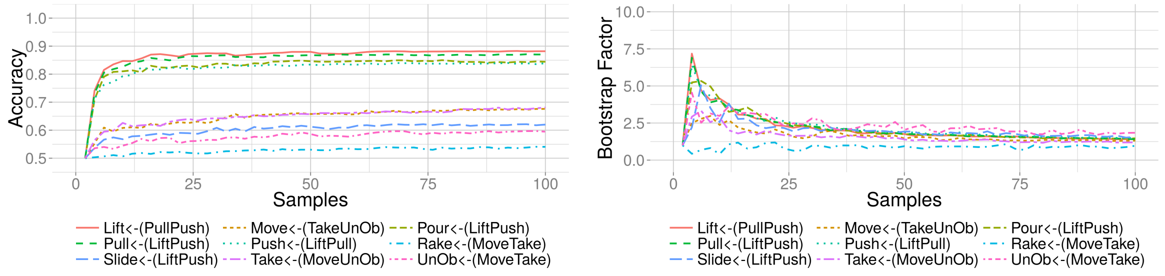


Figure 2.7: Illustration of action success predictor learning for nine actions using a single category predictor output for bootstrapping. Of all the potential category predictor sources for bootstrapping, for each action, only the best result is presented.

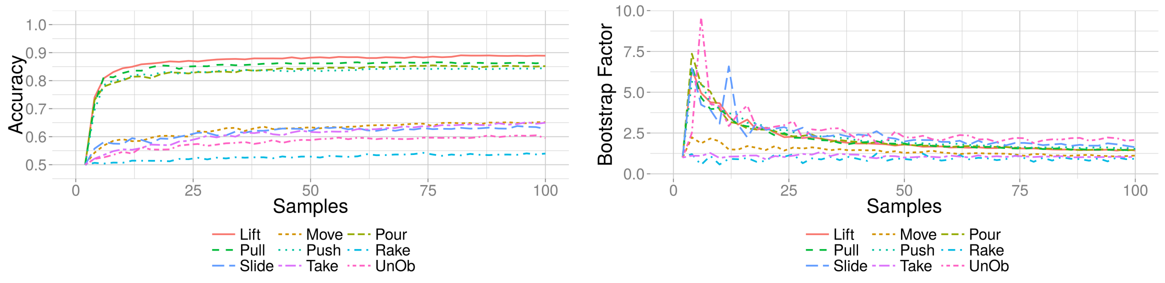


Figure 2.8: Illustration of action success predictor learning for nine actions using all other category predictor outputs at once.

bootstrapping highlight the bootstrapping performance. With bootstrapping, the action success classifiers often reach up to 7.5 times the accuracy of not-bootstrapped classifiers when using the same amount of training samples at early stages of learning (up to 9.5 in the best case) It can also be seen that not bootstrapped classifiers can need several hundred training samples to reach the same performance that is reached by a bootstrapped classifiers after as few as twelve training samples.

2.1.2 Learning to grasp objects based on feature relations

Our work within learning of grasp affordances for use on previously unseen objects was initially motivated by the work performed in [17], see also Figure 2.9. In this work, the idea was to learn a predefined set of grasp primitives, called elementary grasping action (EGA), that where triggered by either co-planar pairs of contours (Figure 2.9a-c) or by a single surface feature (Figure 2.9d-f). Three different grasp types where hand defined for each of the heuristic based feature constellations and this knowledge was applied on real scenes of objects. By utilising the set of EGA's in a combined way (selecting one of them in turns) a rather high rate for performing a successful grasp was achieved of around 20-30%.

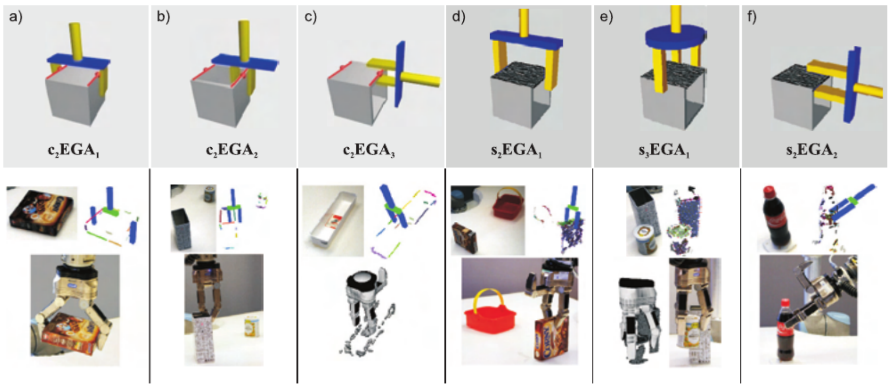


Figure 2.9: Overview of the elementary grasping actions EGA, presented in [16]. a) to c) shows hand defined EGA's given co-planar contours and e)-f) show hand defined EGA's with respect to surface features. Figure is taken from [17]

In the following two subsections, we first describe the learning of grasp affordances directly from surface features (section 2.1.2.1). In section 2.1.2.2, we then introduce an additional in-between level to the very same learning algorithm, in which a discrete set of prototypes of visual structures are acquired by unsupervised learning. We show that we achieve better performance on an even harder problem by exploiting the visual structures provided by this inbetween level. In addition we show, that the extracted discrete set of features can be associated to semantically meaningful structures such as "walls" and "borders" to which grasp affordances can be statistically linked.

2.1.2.1 Learning of grasping affordances without prior learned visual structures

In our work in Xperience, we proposed a framework for learning constellations of pairs of surface patch features of different sizes and learning associated grasping affordances in a probabilistic way [34] instead of hand defined such as in [17]. The aim of the work was to learn some of the underlying structure that exists in a visio-action space spanned by a combinations of features, see also Figure 2.10. In this context, we introduced a semantic property of a local feature to be a border feature, see Figure 2.10b. This feature type was found by means heuristics and gave the additional information for a feature to be at a border and have a direction towards the border. Given this, we spanned a space of feature pairs by means of spatial relationships and two finger pinch grasps, allowing for learning structures in a feature grasp domain and apply it on novel objects. The approach was applied to an object set consisting of three different object classes, respectively open- (container), round- and box objects in a simulated environment. The results of the investigation, see in Table 2.1, show success-rates in the range of 68% to 83% depending on the object set. A particular interesting results is the impact that the border feature have for the open object set, where the structure that it contributes with improves the results significantly. This lead us to believe that we need to add/learn additional structure at the lower level features to achieve better predictions. This work has lead to a publication as a journal paper [34].

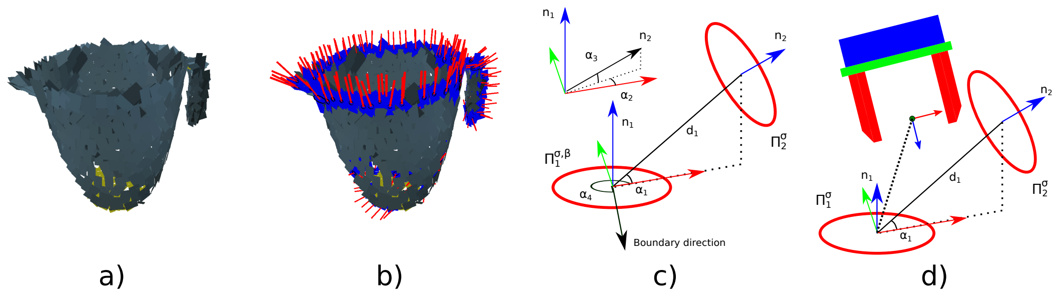


Figure 2.10: Work of [34], a) Shows a basis surfling feature representation. b) basis surfling representation where borders are found using heuristics. c) Utilised 2nd order combinations of features. d) Linking visual features to actions. Figures are taken from [34]

Table 2.1: Comparison of results acquired without prior visual learned structures [*] from [34] and with prior learned visual structures [**] from [35] where 20 prior visual features were learned. The results without prior visual knowledge were only performed object class wise (therefore the empty fields in the table). For the results with prior visual knowledge, the results were performed for the all the object classes making the problem more difficult. Random depict the chance for randomly selecting a successful grasp in the set.

	Containers [%]			Boxes [%]			Curved [%]			All [%]		
	NPG	WPG	CG	NPG	WPG	CG	NPG	WPG	CG	NPG	WPG	CG
[*]	68	-	-	-	84	-	-	84	-	-	-	-
[**]	78.0	40.5	74.0	13.6	87.9	87.9	6.3	92.5	92.5	37.0	70.4	83.8
random	10.6	11.5	-	4.8	46.6	-	4.4	51.3	-	7.0	34.1	-

2.1.2.2 Learning of grasping affordances with prior learned visual structures

Instead of learning from local surface structure directly, we now introduce a discrete set of semi-local visual prototypes acquired by unsupervised learning as an in-between level to the very same learning scheme as used in [34].

Given the aforementioned results, we proposed the SPGF (Sliced Pineapple Grid Feature) in [35], a semi-local surface based feature, see also Figure 2.11. The feature is based on radially distributed pairwise spatial relations between a central features and its neighbours which are organized into a circular grid structure (Figure 2.11a). In this grid structure parametrization, we extract a finite set of descriptors by means of k-means clustering (Figure 2.11b) allowing us to infer features on a novel situation (Figure 2.11c). The features are learned on the full object dataset used in [34], resulting in a shared visual representation. Given this representation we apply a similar probabilistic grasp affordance learning approach as in [34] with the difference that we learn affordances for the discrete set of visual features. In addition, we learn two types of pinch grasps a wide pinch grasp (WPG) and a narrow pinch grasp (NPG). Given this system, we show that we are able to learn reusable visual structures for a variety of objects that can be used for predicting successful action affordances with a reasonable probability. We show that the learned visual representation can be utilised for predicting two different grasp types with rates for a successful grasp in the range of 74% to 93% depending on object class (Table 2.1).

When assessing the learned visual feature, we have identified the border feature (previously derived from heuristics in [34]) as well as other type specific surface structures such as ”walls” and ”borders” extracted from the object data, hence we could replace manual design settings by learning.

In particular, we have shown how a finite set of semi-local visual features derived by k-means clustering allows for learning grasping affordances. In addition, the property of utilising the visual structures for multiple different grasp types is observed in the visio-action visualizations in Figure 2.13. These figures show the grasp success probability for the two different grasp types with respect to the previously derived visual structures. When comparing the figures it becomes obvious that only one structure, number eight, is a reasonable predictor for the narrow pinch grasp, whereas a number of different structures show prediction potential for the wide pinch grasp (2,4,6 and 10). This work has been published in [35].

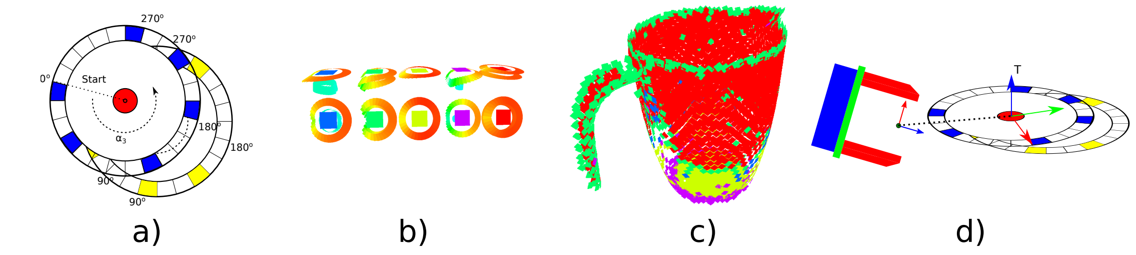


Figure 2.11: Overview of the SPGF feature presented in [35]. a) The circular grid structure of the SPGF features describing the spatial relations of the neighbourhood radially. b) A finite set of features derived from the circular grid by means of k-means clustering. c) Inferring of the clustered visual types on a novel object. d) Linking of action with visual types. Taken from [35]

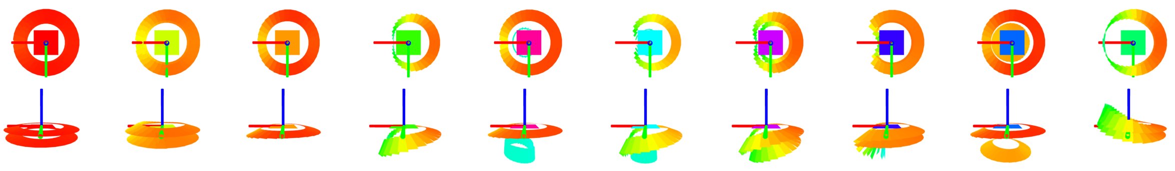


Figure 2.12: Visualisation of a learned set of visual features with K=10. The features are denoted 1 to 10 from left to right. The bottom and top rows show the same features from different angles. In addition to the actual inclination of the outer ring feature, the colour also denote the angle difference to the normal of the centre feature, green/cyan depict strong curvature whereas red depict none or little curvature. The orientation of the features are described by the inlaid frames (red, green and blue sticks). Taken from [35]

2.1.3 Accelerated motor learning in constrained domains

In this section we demonstrate the effectiveness of structural bootstrapping for motor learning. We considered the problem of learning compliant movement primitives, where feedforward torque commands need to be learned to enable compliant control. The approach uses a parametric trajectory representation that includes both position and joint torque trajectories. This representation, called Compliant Movement Primitive (CMP), is based on dynamic motion primitives (DMPs) for the position part and a weighted sum of radial basis functions for the feedforward torques, i. e. torque primitives (TP). The details are described in [22]. The learned torque components are used in a feed-forward manner in order to reduce the necessary feedback gains while maintaining the same level of accuracy but with compliant behavior of the robot. The method can be applied for any task or task variation, but the torque trajectories have to be learned separately for each of these. This representation was developed in the FP7 project ACAT.

Since torque profiles have to be available for each task and even each task variation, their practical implementation requires that they can be learned also without human demonstration. For this purpose we first note that the parametric nature of the trajectory representation allows us to employ real-time statistical generalization, i. e. Gaussian process regression, in order to generalize between a smaller subset of learned behaviors [8]. In [6] we showed that generalization achieves similar levels of accuracy as the directly demonstrated movements when generalizing from a database of learned CMPs. However, the problem of building a comprehensive database remains demanding and time consuming. In the Xperience project, we developed a new approach to reduce the learning time by having a robot to autonomously expand its database through learning using the mechanisms of *structural bootstrapping* [PCG$^+$15, FNU15].

Autonomous learning of CMPs simplifies the execution of dynamically versatile tasks while ensuring accurate and compliant execution of the motion. However, since torques are not linearly scalable, TP part of CMPs have to be learned for every variation of the task. These variations can include different speeds, payloads, goals, etc. The process of learning TPs can be significantly accelerated by using statistical generalization techniques, which can generate first approximations of the TPs based on a given query point. In case the generalized TP achieves the desired performance, it can be immediately added to the database of motion. If not, a recursive regression method [GPD$^+$16] can be applied using the generalized TP for the initial approximation, vastly reducing the number of needed iterations of motion.

Instead of learning all task variations by demonstration, we developed an algorithm where we bootstrap

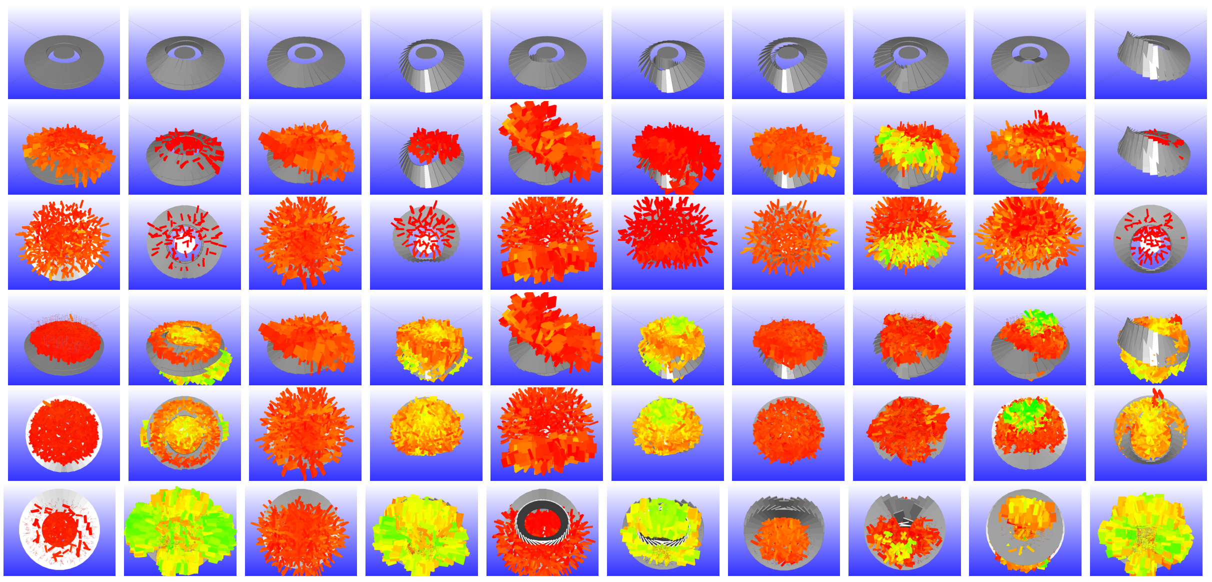


Figure 2.13: Visualisation of the visual features with the learned grasp affordances for the NPG grasp (rows two and three) and the WPG grasp (rows four, five and six). The features are denoted 1 to 10 from left to right. Red areas depict low probability (0.0) of success. The colour changes towards green that depict a success probability of 1.0. First row show the features, second and fourth row show the features with associated grasp from a perspective view and the third and fifth row show the features from a top view. The sixth row show features from below view. Taken from [35]

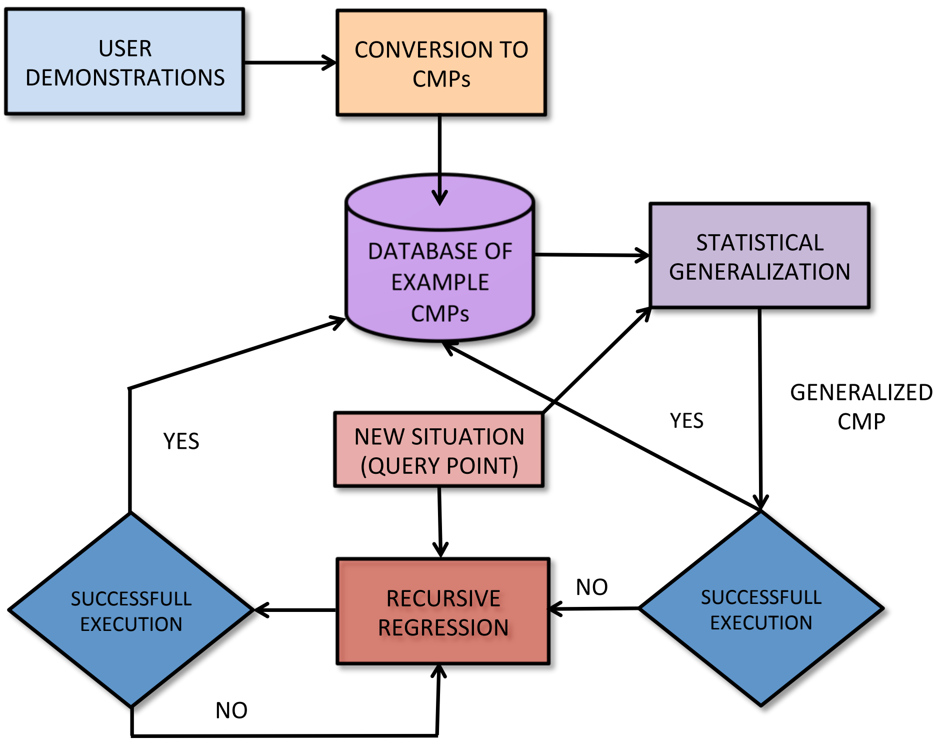


Figure 2.14: Learning of compliant movement primitives using bootstrapping in the motor space

autonomous learning of a new trajectory (a CMP includes both position and torque trajectory) with an initial approximation provided by Gaussian process regression. Fig. 2.14 shows the basic schema of the approach. Thus the learning starts from a good initial estimate and therefore needs less iterations to accurately execute the motion. The number of required iterations is significantly reduced with each new database entry. It should be noted that the learning can be either unsupervised or supervised.

The method was evaluated on a humanoid robot performing a reaching task and on Kuka LWR-4 robot performing a peg-in-hole task. We compared the speed of learning when bootstrapping for initial approximation and without the bootstrapping. The learning results for eight reaching examples of both cases are shown in Fig. 2.15, where we can see that the convergence was faster with bootstrapping for the initial estimate. Similar learning performance can also be observed for the peg-in-hole task. The learning results are shown in Fig. 2.16, where we can see that the learning with bootstrapping is significantly faster compared to learning without any prevision experience.

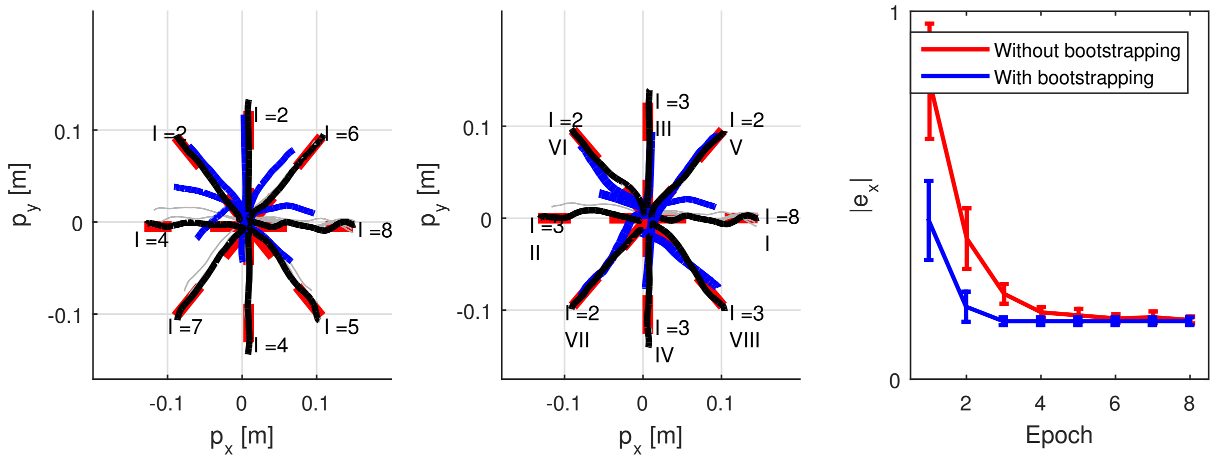


Figure 2.15: The left plot show the results without bootstrapping and the middle plot shows the results with bootstrapping, where the sequence of learning is determined with the Roman number. The desired trajectory is shown with doted red line, initial trial is shown with blue line, gray thin lines show the intermediate learning iterations and the black thick line shows the learned behavior. The I indicates the number of epoch needed for successful learning of CMP, i. e. once the error was smaller than the desired threshold. The right plot shows the error and the standard deviation for both cases.

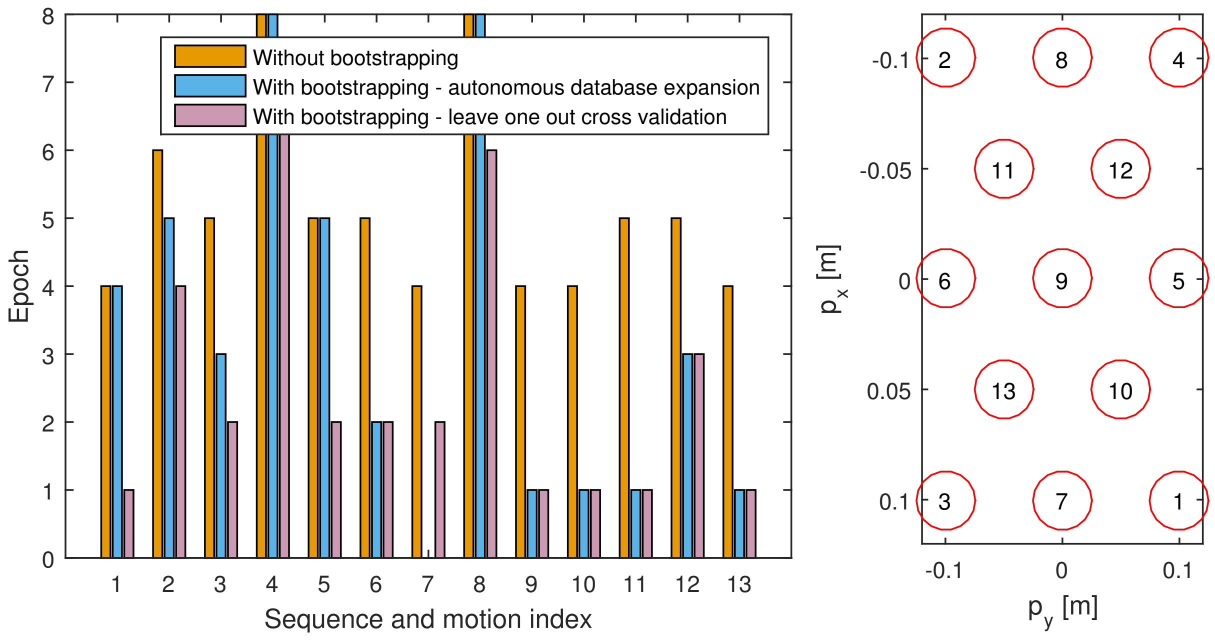


Figure 2.16: The left plot show the number of epochs needed to accomplish desired accuracy while performing peg in a hole task. The right plot shows the sequence of learning.

2.1.4 Action replacement versus unconstrained learning

This section relates to the Xperience deliverable D.1.2.1, chapter 5, bootstrapping at sensorimotor level. As explained in D.1.2.1, some actions appear to be similar at higher-level information layers. It was suggested that we could bootstrap the learning of a new action by the parameters of an action, which is similar at the semantic level [40].

Our hypothesis is that knowing that two behaviors are similar can result in faster learning. Here we demonstrate the benefits of bootstrapping for learning of a stirring behavior, which is bootstrapped by the parameters of a previously learned wiping policy for pot stirring. While initially we planned to use Reinforcement learning (RL) for the purpose of benchmarking, it turns out that in many cases Iterative Learning Control (ILC) and Repetitive Control (RC) [39] can be applied to achieve faster results [38]. Therefore, we benchmark the learning of the pot stirring action with action replacement using repetitive control.

Our experimental setup was composed of two KUKA LWR robots, equipped with Barred hands and controlled using Fast Research Interface (FRI) [29]. First, wiping policy was obtained using LbD, see Figure 2.18. This was accomplished using kinesthetic guiding in gravity compensation mode and recursive regression [GPD$^+$16]. Next, our goal was to learn how to stir in a metal pad of diameter of 21 cm using wooden spoon (see Figure 2.17). The position, size and shape of the pad was not known. However, we assumed that the initial pose of the robot tool (the spoon) was inside the pad and that the robot orientation and z coordinate were also known. They remained unchanged during the stirring.

As a learning algorithm, we applied Repetitive Control (RC). The RC algorithm repetitively modifies the movement to achieve the desired behavior, which in this case was defined as moving in contact with the edge of the pot (this criterion is specified by the teacher). In order to determine the benefits of bootstrapping similar activities, we considered two cases:

1. **Without bootstrapping:** learning without any previous knowledge about the stirring trajectory and
2. **With bootstrapping:** the initial trajectory was obtained using LbD for wiping and the result was encoded as a periodic DMP.

The task of the robot was to acquire the stirring motion as quickly as possible.

In our experiments we initially placed the robot's end-effector at 2-3 cm away from the center of the pot. Figure 2.19 shows the evolution of the stirring motion in $x - y$ plane. The evolution of motion as a function of time is shown in Figure 2.20. Note that the robot learned the desired policy in approximately 15 cycles without any prior knowledge and in approximately 7 cycles with the prior knowledge of the trajectory taken from wiping. Figure 2.19 also shows that the tracking was not perfect, but note that the tracking of the robot hand was captured, which was different from the motion of the spoon due to the high compliance in robot joints. The actual trajectory of the spoon was following the pad shape.

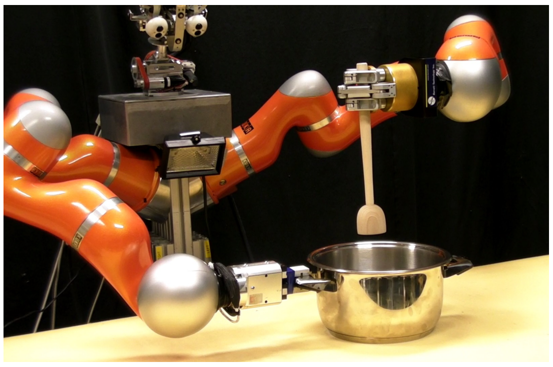


Figure 2.17: Experimental setup

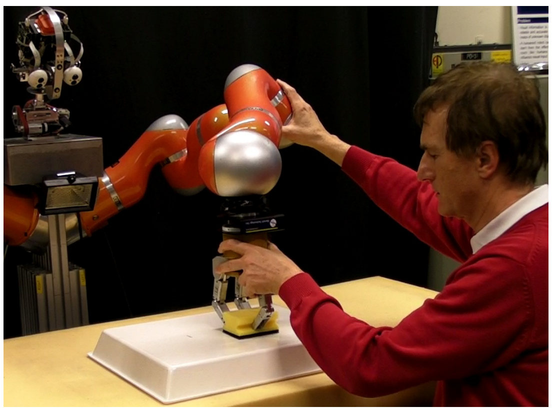


Figure 2.18: Demonstration of wiping

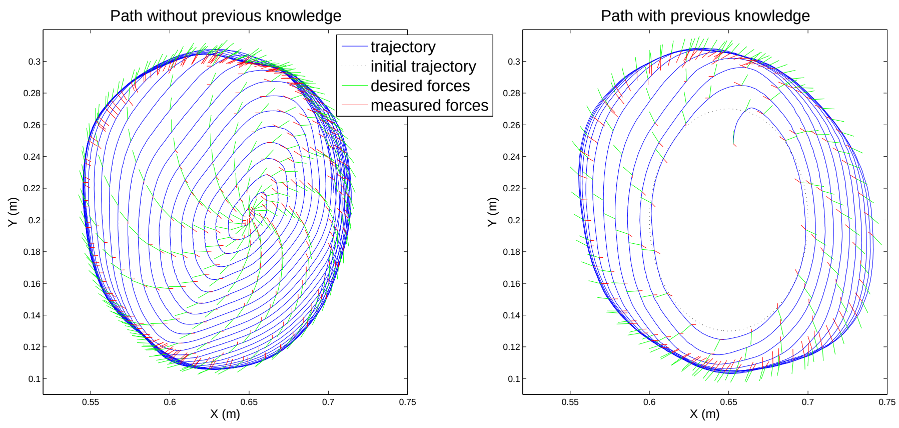


Figure 2.19: Learned path with desired and actual forces: left) without previous knowledge, right) initial wiping trajectory was provided

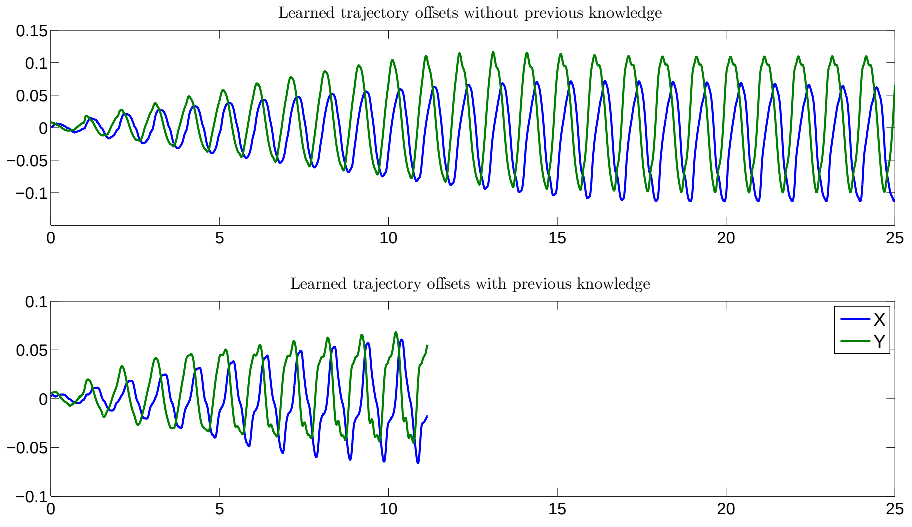


Figure 2.20: Time plot of the learned path: upper) without previous knowledge, lower) initial wiping trajectory was provided

2.1.5 Bootstrapping reactive grasping by combining perception and action

Although many papers on image segmentation contain statements such as "segmentation is an important pre-processing step for object recognition", the practical usefulness of low-level segmentation algorithms for the purpose of object recognition has been questionable up to now. This is due to many ambiguities in natural images that can lead to different segmentation results. We developed an interactive approach that resolves such ambiguities and makes low-level segmentation more reliable. Our hypothesis is that the integration of visual perception and physical interaction for autonomous segmentation in cluttered environments can *bootstrap visual object learning* and lead to a significant increase in robustness when recognizing and grasping previously unknown objects (see [28], [27], [25], [26]), without needing to acquire many thousands of annotated images as statistical methods do. In this section we describe how we benchmarked the proposed interactive system with respect to the number of pushes that are needed to reliably segment unknown objects from the background for the purpose of model learning, recognition and grasping.

2.1.5.1 Assessing the Segmentation Quality

We examined the performance of our interactive segmentation approach by testing it with 30 objects of different shape, size and visual appearance type, which have been segmented twice each. To measure the quality of the obtained segmentations, two metrics are determined: First, the object should be segmented as completely as possible, i.e. in an optimal case the point cloud forming the object hypothesis should fully cover the object. The second metric is the size of the falsely segmented area, i.e. the part of the scene that is segmented but does not belong to the object. This happens when the object hypothesis includes points that belong to the background or other objects.

Figure 2.21 shows these two metrics depending on the number of pushes executed. As can be seen, after the first push the object is usually not covered completely, but already large part of it. After two to three pushes, the object hypothesis contains almost the complete object, with the exception of small patches that newly appeared due to object rotation or that were discarded from the hypothesis due to a change in their appearance (e.g. reflections or bad depth estimation). After four pushes, the coverage does not improve further, but different parts of the object may become visible, thus more information can still be gained. The ratio of falsely segmented image regions compared to the whole object is always small.

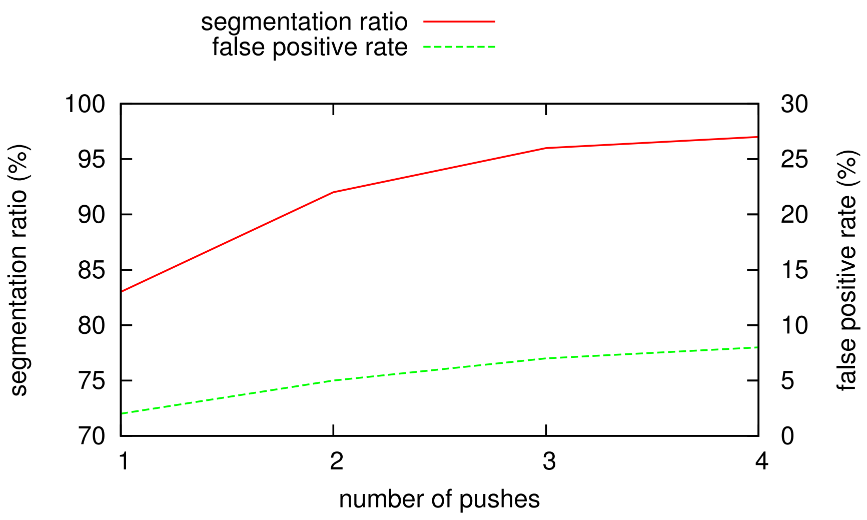


Figure 2.21: The average segmentation quality depending on the number of pushes that were executed. The red line shows the segmentation ratio, i.e. the percentage of the object that is included in the segmentation. The dashed green line depicts the false positive rate, i.e. the fraction of the segmentation that does not belong to the actual object.

In summary, the interactive segmentation approach leads to very good segmentation results in complex scenes which in general are hard or even impossible to obtain by otehr segmentation methods.

2.1.5.2 Assessing the Visual Object Learning based on the Segmentation

To benchmark the segmentation method, we tested how many pushes are necessary to achieve a good multiview coverage of different objects. To this end, we performed a training process with 30 pushes for 5 of our test objects. We took test images of each of them from 8 different view directions, where the object was turned on the table in steps of 45°. We then tried to recognize the object using classifiers learned from subsets of the acquired descriptors. As expected, when only the first few training images were used, the object recognition succeeded only from some of the 8 viewpoints, but with an increasing number of pushes and consequently more training images from different viewpoints, the recognition rate improved (see Fig. 2.22). When looking at the individual objects, it seemed that a certain saturation was usually reached between 15 and 25 pushes.

In summary, the interactive segmentation approach provides an efficient way for learning multiview visual object representations autonomously.

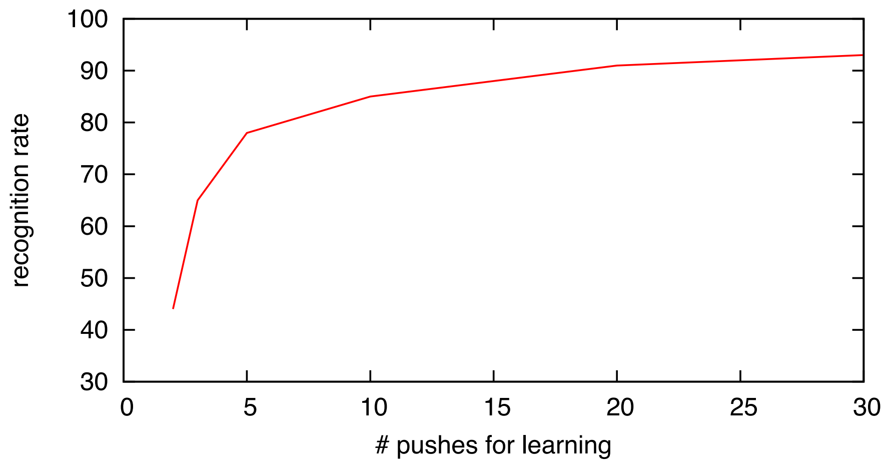


Figure 2.22: The performance of multiview object recognition with respect to the number of training images acquired from different viewpoints.

2.2 Objects and Actions

2.2.1 Robot object manipulation database for recognition

The ability to visually recognize objects is a fundamental skill for robotics systems. Indeed, a large variety of tasks involving manipulation, navigation or interaction with other agents, deeply depends on the accurate understanding of the visual scene. Yet, at the time being, robots are lacking good visual perceptual systems, which often become the main bottleneck preventing the use of autonomous agents for real-world applications. Lately in computer vision, systems that learn suitable visual representations and based on multi-layer deep convolutional networks are showing remarkable performance in tasks such as large-scale visual recognition and image retrieval. To this regard, it is natural to ask whether such remarkable performance would generalize also to the robotic setting. In this section we investigate such possibility, while taking further steps in developing a computational vision system to be embedded on the iCub humanoid robot. In particular, we released a new data-set (ICUBWORLD28 [12]) that we use as a benchmark to address the question: how many objects can iCub recognize? Our study is developed in a learning framework which reflects the typical visual experience of a humanoid robot. Last year, we started collecting and making available a data-set (ICUBWORLD1 [12]) that reflects the typical visual experience of iCub and testing different solutions for visual recognition. Our preliminary results confirmed on the one hand the potential of recently proposed systems, and on the other, highlighted the challenges posed by the specific robotics context in particular the lack of accurate supervision. This section builds on our previous work (presented in Y4) to take a further step in the development of a computational vision system for the iCub. In particular, in this section we conduct an empirical study focusing at key performance indicators as benchmarks in order to eventually answering the question: ***How many objects can iCub recognize today?***

We consider this problem within the Human-Robot Interaction scenario proposed in [2] and [3]for the acquisition of the ICUBWORLD data-set. In the current work, a human teacher shows 28 different objects to the iCub, verbally annotating them using a speech recognition system to provide labeling. The same procedure is repeated for four consecutive days, leading to the acquisition of a new data-set, dubbed ICUBWORLD28

An ideal robotic visual recognition system

- **Reliability**: In order to be reproducible, our analysis will be performed off-line on a visual recognition data-set directly acquired from the robot cameras, ICUBWORLD28. However, in order to generalize the recognition performance observed on such a benchmark, we will need a measure able to quantify the confidence with which we can expect such results to hold also in the real-world application.

- **Contextual Information**: The robotic setting offers a great deal of contextual information that could be incorporated in the learning system to improve recognition performance. For instance, by observing an object from different points of view, the robot could be able to better disambiguate between different classes. Typically, contextual information is not available in standard computer vision settings and therefore is unclear in general how to employ it in recognition

- **Learning incrementally**: A human-like artificial system should be able to learn a richer model of the world as new observations become available. Specifically, it is natural to expect that the visual recognition system of a humanoid robot should benefit from the incorporation of visual data acquired on multiple occasions, such as training sessions across multiple days.

- **Self-Supervision**: Ideally, the interaction between a human and a robot should take place along natural communication channels (for the human), such as speech or vision. Clearly, such a scenario limits the amount of supervision that a human teacher can provide to the robot. For instance, in the human-robot application considered in this work, images cannot be manually segmented around the object of interest and therefore the system has to rely on so-called weak or self- supervised strategies, such as motion segmentation, to eliminate, at least partially, the visual distractors (e.g. background or other objects).

Setup and Acquisition

Setup: The application setup we employed in this work is analogous to the one described in [3] and we briefly outlined it in the introduction of this paper: a human supervisor is standing in front of the iCub robot and shows it different objects while verbally providing the class annotation. Exploiting independent motion detection routines [2], the robot tracks the novel object while acquiring images at 33hz. The independent motion detection algorithm allows to perform an approximate localization of the object, effectively reducing the image size from 320 240 pixels to a mean of   120 120 pixels (See Fig. 2 for an example). Cropped images are then processed by a representation module that encodes the visual information into a single vector or descriptor that will then be used for classification 2.23.

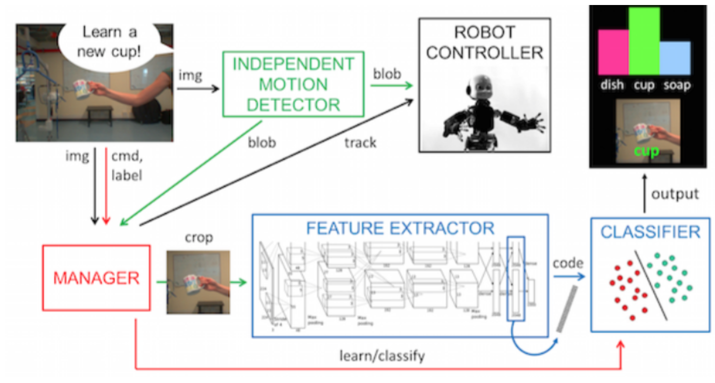


Figure 2.23: The visual recognition system adopted in this work and currently implemented on iCub.

Acquisition: Within the setting described above, we collected the ICUBWORLD28 dataset which comprises images of 28 distinct objects evenly organized into 7 categories 2.24. For each object in the dataset, we acquired a separate train and test sets during sessions of 20 seconds each. We reduced the acquisition frequency by a factor of 3 (i.e. acquiring one image around every 0.09 seconds) to lower the computational costs of the learning process. Thus, after each session, we collected 220 train and 220 test images for each of the 28 objects. To assess the incremental learning performance of the iCub visual recognition system we repeated this same acquisition protocol for 4 consecutive days, ending up with four datasets (Day 1, to 4) of more than 12k images each and 50k images in total. This release is available at [12].

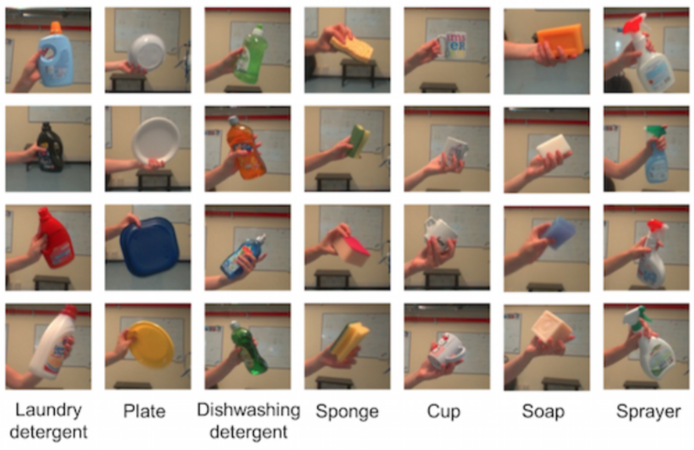


Figure 2.24: Example images from one of the 4 datasets comprising ICUBWORLD28. As can be seen in the figure, each data-set is composed by 28 objects organized into 7 categories.

Extracting visual representation. To extract visual representations of images acquired from iCubs cameras, in this work we employed a CNN originally trained on the ImageNet dataset [24]. Specifically we employed a model provided in Caffes library [14], BVLC Reference CaffeNet, which is available online and is based on the well-established network proposed in [18]. Following the strategy proposed in [32] and [7], we employed the CNN as a black-box module that takes images in input and returns their corresponding vector representations in output.

Learning. In visual recognition settings, the typical approach to classification is to employ so-called supervised learning methods such as Support Vector Machines or Regularized Least Squares (RSL). In this work we rely on the GURLS [33] machine learning library to perform RLS. Indeed, as empirically observed from previous work on the iCub, RLS exhibited comparable or even better results than Support Vector Machines. Moreover, the rank-one update rule for matrix inversion provides a natural variant of the classic RLS algorithm to the setting in which training data is provided incrementally to the system (also the incremental RLS algorithm is implemented in the GURLS library).

Reliability and Scalability

Ideally, a reliable recognition system should be robust with respect to set of objects it has to discriminate.In other words, we would like the classification performance of a predictor to not vary dramatically when we change the set of classes on which it is trained/tested. Therefore, to quantitatively measure the reliability of the visual system currently available on the iCub, we performed multiple classification experiments for different subsets of classes in ICUBWORLD28 for the data-set corresponding to Day 1. More precisely, for any t = 2, . . . , 26 we randomly selected 400 different combinations of t object classes among the available 28 (to avoid the combinatorial explosion of 28 t experiments) and trained/tested the learning system described previously on the corresponding reduced data-sets. As a measure of performance for the resulting predictor we computed its average accuracy, namely the ratio of correct guesses with respect to the cardinality of the whole test set.

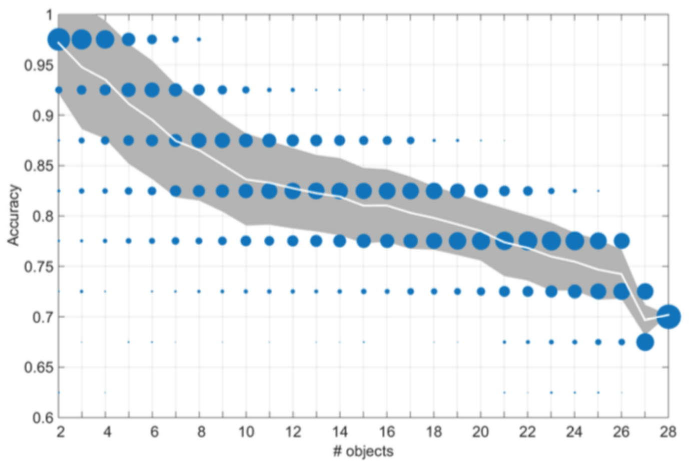


Figure 2.25: Empirical estimation of the probability distribution P(acc = A—t) for a predictor trained on a random set of t objects to have accuracy A.

Apart from the expected drop in accuracy that we observe when the cardinality of the multi-class problem increases, this analysis provides us with useful insights: first notice that the slope of the mean accuracy reported in 2.25 (white curve) experiences a remarkable decrease as the number of classes increases (e.g. after t = 10), suggesting that such a negative effect should become less and less disrupting as we learn new objects. Second, notice that for each fixed cardinality t, the distribution of accuracies P (acc = A—t), measured across the multiple trials, is clearly concentrated around its mean. More specifically, this means that in general we can expect with high confidence that a predictor trained on a randomly selected set of t objects would have accuracy between 5% of the mean of P (acc = A—t). This offers a useful perspective on what recognition performance we should expect during a typical run of the human-robot interaction application. To better quantify the expected capability of the system to generalize its performance, in 2.26 we report the minimum accuracy that we are guaranteed to achieve within specified levels of confidence. To better understand the implications of this analysis, let us consider for instance the Blue curve in 2.26, related to 95% and passing by t = 15 and A = 0.75: with high probability (95%) and for a random choice of 15 objects, the resulting predictor is guaranteed to achieve at least 0.75 classification accuracy. This result, and its corresponding visualization in 2.26, is of particular use from a practical perspective since it can be employed as a reference data-sheet to train the iCub. Indeed, depending on the desired confidence C and the number t of objects we want the robot to discriminate, 2.26 informs us what is the approximate level of accuracy that we can expect to achieve with the classifier that we will train.

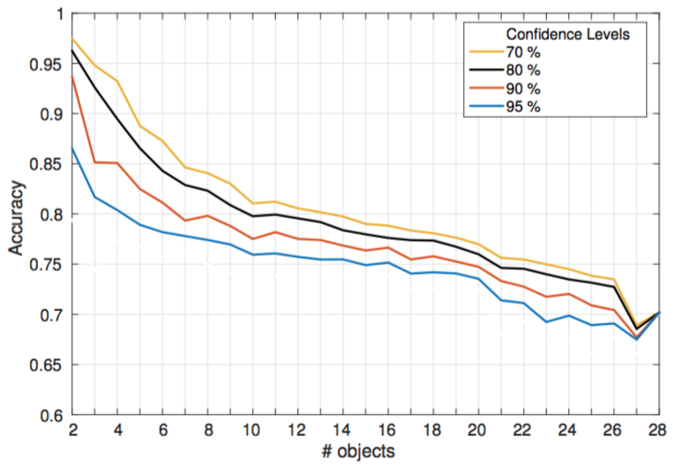


Figure 2.26: Confidence intervals for predictors trained on a randomly sampled set of objects. For a fixed number of objects t, the value on a curve C represents the minimum accuracy that we are guaranteed to achieve with the trained predictor, with confidence C.

Exploiting Contextual Information

The classification performances reported in 2.26 are clearly not comparable to the human-level accuracy that we would expect on the problem considered. Indeed, even for relatively low confidence values such as 80% (Black curve), we observe a fast decay of the guaranteed accuracy, which falls under the 0.9 threshold just after 4 objects. A viable approach to mitigate this problem relies on noticing that the robotic setting offers a great deal of prior and contextual information that could remarkably improve performance. To this regard, let us consider the natural assumption that the class of an object does not change while the robot observes it from multiple points of view. In such a setting, given a set of w images (acquired from different viewpoints around the object of interest) and a trained classifier with a (per-frame) accuracy A, we can consider a new classification rule that combines the individual predictions on the set of w frames into a global label. For instance, if we assume that the w images are sampled i.i.d., we have that the rule returning the label that occurred at least 50% + 1 times would correctly classify the object with probability (or accuracy). In principle, this strategy could be extremely beneficial: suppose for instance that the trained predictor has a per-frame accuracy of A = 0.7. Then, even for small sets (or windows) of just 3 images we would have improved classification accuracy of 0.78, while for a larger w = 21 we would achieve an impressive 0.97. We evaluated the approach described above on ICUBWORLD28. In particular, since in our setting the samples are acquired as a stream of consecutive images and we are interested in on-line recognition, we chose to classify windows selecting the current frame together with the previous w 1 ones. This approach could be interpreted as a sort of label-filtering process that suppresses flickering one-frame misclassification.

2.27 reports the effect of the label-filtering approach on the confidence curve associated to C = 80% introduced in 2.26. We varied the size of the temporal win- dows from 0 (instantaneous) to 4 seconds, corresponding to a range of w between 1 and 50 frames. Notice that even in this non i.i.d. scenario, the system performance clearly benefits from smoothing, in particular when several classes are considered. Probably this is due to the fact that, as the number of object to discriminate grows, the chance of short-lived one-frame misclassification increases proportionally.

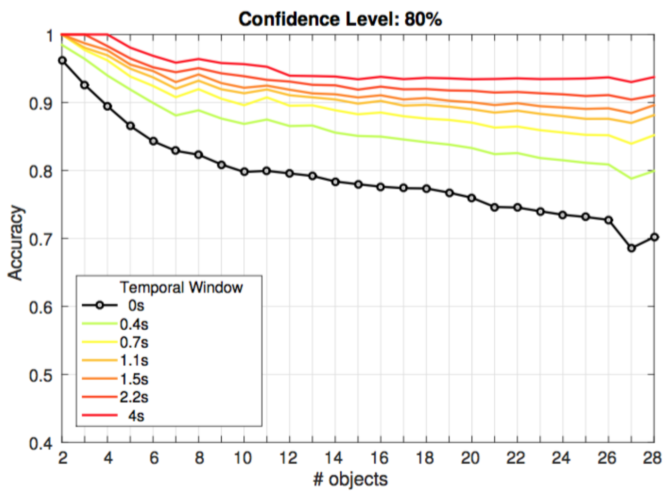


Figure 2.27: Improvement of the classification accuracy with respect to an increasingly large temporal filtering window. Results are shown for fixed confidence level C = 80% (see 2.26).

Incremental Learning

The temporal filtering strategy considered previously leads to an impressive boost in recognition accuracy. However, if we consider the original goal of achieving human-level performance on ICUBWORLD (say, for reference, 0.98 accuracy), we notice from 2.27 that even for a relatively low confidence value of 80% the system is still lacking a significant accuracy gap. To this regard, in this section we take into account another aspect of robotics settings that could in principle improve the recognition capabilities of the system, namely the ability to learn incrementally. Indeed, the robotic scenario is naturally suited to life-long learning applications. Specifically, in visual recognition settings, novel training evidence could be provided to the robot incrementally (and in principle, indefinitely) in order to update its knowledge as the task requires. A first result, that empirically quantifies the importance of learning incrementally and motivates the experimental analysis of this section, is reported in 2.28. We consider the experimental setting previously introduced and report the curve associated to 80% confidence for classifiers trained on an incremental number of examples per class. As can be noticed, the incremental growth of the training data has a remarkable impact on the overall classification performance and opens the question of what would be the long-term effects of such a learning process on the system's recognition capabilities.

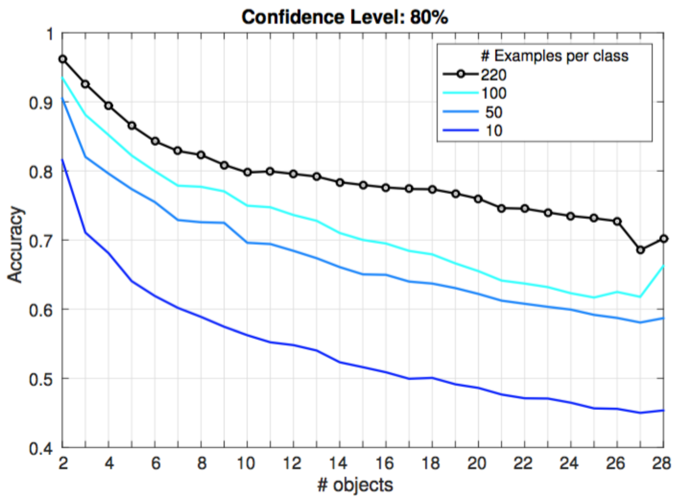


Figure 2.28: Classification accuracy for fixed level of confidence C = 80%) and an incremental number of training examples per class.

	Day 1	Day 2	Day 3	Day 4	Average
Day 1	67.7	41.9	37.2	67.2	53.5
Day 2	40.1	67.8	35.4	66.8	57.5
Day 3	62.0	63.5	66.4	64.9	64.2
Day 4	62.9	64.1	65.3	67.1	64.8
All days	73.4	71.0	68.1	68.9	70.3

Table 2.2: Accuracy of predictors trained on single days compared with a predictor trained on all days together. For a fair comparison, the training data-set have same size (100 examples per class).

We recall that ICUBWORLD28 is a data-set collected during 4 separate days and that for each day both a training and test set were acquired. We further recall that all experiments discussed so far were performed on a single day of ICUBWORLD28, say Day 1. To the purpose of studying the impact of incremental learning on visual recognition, in the following we will take into account also to the remaining 3 days. In particular, we considered the learning setting in which we trained a classifier incrementally on the training sets of the first three days of ICUBWORLD28 and then evaluated it on the tests set of the fourth unseen day. To reduce the amount of computations we focused only on the problem of correctly classifying the 28 objects in the data-set and report the measured accuracy in Fig. 8 for classifiers trained starting respectively from Day 1 (Blue), Day 2 (Orange) and Day 3 (Yellow). On one hand, we notice that when provided only with training data acquired from a single day, the incremental learning accuracy exhibited by predictors follows a remarkably similar pattern for all days, suggesting the the three data-sets contain a similar amount of information. On the other hand, we observe that while all these curves seem to saturate around 0.65 accuracy, adding data from a new day allows to overcome such limitation, improving the over- all system performance (here we refer to the jumps observed for both the Blue and Orange curves as they switch between days). The results reported in 2.29 seem to suggest that training across multiple days is more beneficial than training during a single session because it exposes the system to less redundant information. To confirm this observation we considered a further experimental scenario where we compared the performance of a predictor trained on data acquired from all days with the accuracy achieved by other four classifiers, each trained on a different day of ICUBWORLD28 taking the first 100 examples per class. In order to compare problems of identical dimension, the mixed dataset was created by taking the first 25 samples (per class) from the training set of each day. 2.2 reports the resulting classification accuracy tested separately on each day. In line with the original intuition, we notice that predictor trained on the mixed dataset clearly outperforms the others on average. However, it is of particular interest to observe that even on a single day basis, the predictor trained on all days (and thus less exposed to redundant information) outperforms predictors trained and tested on the same day.

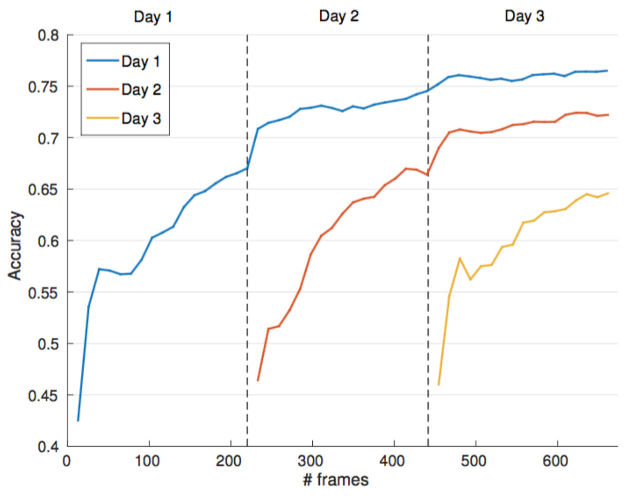


Figure 2.29: Incremental learning on ICUBWORLD28. Blue, Orange and Yellow curves identify the classification accuracy of predictors trained incrementally starting from, respectively, Day 1, Day 2 and Day 3. We used the test set from Day 4 to assess the generalization performance of the classifiers.

	Day 1	Day 2	Day 3	Day 4	Average
PHOW [1]	42.5	39.0	34.6	39.0	44.1
BoW [5]	44.9	40.8	35.3	38.8	41.1
Sparse Coding [41]	29.2	24.1	21.9	23.7	30.6
HMAX [31]	30.5	27.3	25.4	23.7	32.8
Fisher Vectors [19]	47.3	44.7	41.5	44.3	48.6
VLAD [13]	44.2	40.0	35.0	38.1	44.5
CaffeNet [14]	75.9	70.9	71.9	73.9	80.8
OverFeat [30]	66.8	57.5	57.7	60.0	68.3

Table 2.3: Comparison of several architectures for visual representation learning applied to the visual classification problem of ICUBWORLD28. Modern Convolutional Neural Networks (the CaffeNet used in this work and Overfeat) clearly outperform previous methods.

For completeness, we close this work by providing a brief comparison with other methods for visual recognition. In 2.3 we report the classification accuracy of systems trained/tested on the different days of ICUBWORLD28. The following architectures for visual representation learning were evaluated: Bag of Words (BOW) [5], Sparse Coding [41], Fisher Vector [19], VLAD [13], PHOW [1] and the Overfeat implementation [30] of a Convolutional Neural Network. Due to space limitation we refer the reader to the original papers for more informations about these methods. However, we point out that these approaches can be divided in two groups: pre-trained deep architectures (the CNNs CaffeNet and OverFeat) and single layer shallow representations (the remaining methods), where the dictionary learning stage was carried out on a subset of the training set of ICUBWORLD28. As can be noticed pre-trained CNNs clearly outperform the others and this was the main reason for the choice of CaffeNet for our experiments.

Discussion

We identified a natural human-robot interaction application as a possible test-bed for our investigation of the visual recognition problem. In order to foster the re-producibility of our experiments, we collected a novel data-set within this scenario, ICUBWORLD28, comprising images depicting 28 object classes and acquired over the course of 4 days. We approached the problem by first defining a measure performance that would allow us to operatively quantify our confidence that results observed off-line on ICUBWORLD28 would then generalize to the real application. We then identified multiple aspects of the robotics context that could be leveraged to improve the overall recognition capabilities of the otherwise purely-visual system. In particular we empirically observed that exploiting the temporal consistency of subsequent frames in the visual stream or adopting weakly-supervised strategies to reduce the amount of distractors in the image can be extremely beneficial. Following these principles we were able to provide a preliminary answer to the original question *How many objects can iCub recognize today?* . Our results show on one hand that modern visual representation architectures such as CNN are finally able to address visual recognition in robotic settings but on the other hand they point out that the problem is extremely challenging and far from being solved.

2.2.2 Locally convex connected patches

The LCCP algorithm was benchmarked on the Object Segmentation Database (OSD-v0.2) which was proposed by [23]. It consists of 111 cluttered scenes of objects on a table, taken with close proximity to the pictured objects. The scenes contain multiple objects, which have mostly box-like or cylindrical shape, with partial and full occlusions and heavy clutter in 2D as well as 3D.

The qualitative examples (2.30) show that our algorithm performs very well in the segmentation of these cluttered scenes. The object separation can be intuitively understood: all objects present in the scenes are separated by concave boundaries, i.e. a line connecting neighboring surfaces of two different objects always travels through air. This is also true for the boundary between an object and the supporting surface. As a consequence, objects that have a convex shape are correctly captured as one segment and separated from the other objects. Hollow objects (bowls, cups etc.) can be observed to show multiple segments inside, because the orientation of surface normals changes strongly on these concave surfaces.

The quantitative results 2.31 demonstrate that our approach is able to compete with state-of-the-art

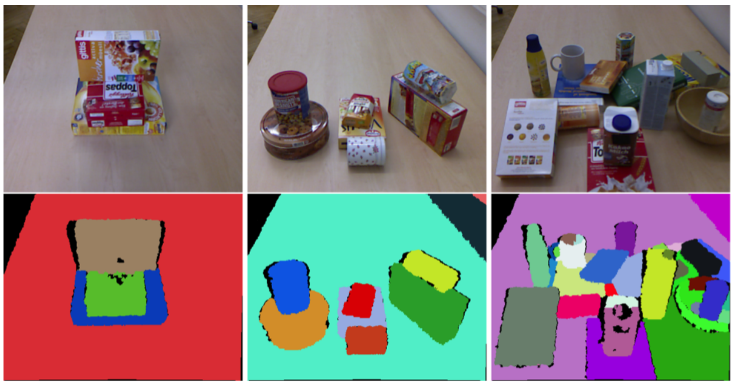


Figure 2.30: Example results for the OSD dataset. Points beyond a distance of 2m were cropped for visualization.

Method	Learned Features	WOv	fp		fn		F_{os}	F_{us}
		Mean	Mean	SD	Mean	SD	Mean	Mean
LCCP	NO LEARNING	88.7%	4.8%	2.6%	8.3%	8.7%	7.4%	4.7%
Richtsfeld et al.	RGB-D + Texture + Geometry	-	-	-	-	-	4.5%	7.9%
Ückermann et al.	NO LEARNING	-	1.9%	3.3%	7.8%	7.3%	-	-

Figure 2.31: Comparison of different segmentation methods on the OSD dataset using weighted overlap WOv (the higher, the better), false positives fp , false negatives fn , as well as over- and under-segmentation Fos and Fus (the lower, the better).

methods in the task of segmenting cluttered scenes into objects. Compared to the learning-based method from [23] we achieve better object separation, but higher oversegmentation error. The latter is because we sometimes detect object parts (handles) and we do not utilize model fitting, which helps against noise. Comparing to the learning-free method of [4] we obtain results within a standard deviation. (2.32) shows results for the LCCP implementation at UGOE applied on different tools. By tuning LCCP parameters it is also possible to retrieve parts of objects (lower path).

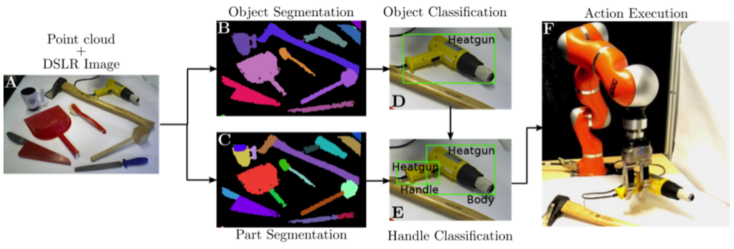


Figure 2.32: LCCP with consecutive classification and execution: A) Original image of the scene, B, C) object/part segmentation, D, E) object/part classification and F) action execution robot grasping a heatgun by its handle.

2.2.3 Inferring Object Affordances By Generalizing From Experience

Here we extend and adopt the ROAR (*repository of objects and attributes with roles*) benchmark framework proposed in D1.2.1. to the final demonstrations. The ROAR implements the learning infrastructure of object-action relation and the replacement of the objects or actions with a similar suitable ones. The system relies on data sources collected by the method proposed by [15]. In that paper a text mining technique is introduced to discover the relationships between different word pairs. The possible connection between the words is described by frequencies of co-occurrences as a raw score; based on those frequencies a probabilistic score is also presented. We refer to this data as *web-relations*.

The full test procedure is built upon a scenario where the ternary relation between a pair of objects and a common action is to be predicted. That kind of relation can answer this type of question: *Given an object and a related action, what are the objects that are most suitable as a substitute for this object under that action?* The "applicability" of those objects is measured by a predicted score derived on the base of the scores taken from the *web-relations*.

The scenario can be summarized in these points:

With bootstrapping (ROAR): The robot checks the ROAR to verify the hypothesis that an object, e.g. an "apple", might be substituted by another one, "orange", for a given action, after observing some affordances of the first object, e.g. "soft" or "cut-able", to establish relations between those objects.

Then it checks the ROAR for those objects in its workspace, and observes that a third object is located in the workspace known by the ROAR, and that it is useful for the action in question.

The ROAR's performance can be quantified without an end-to-end robotic implementation in the following manner:

1. Apply the annotated, realistic, ground-truth data set G, the *web-relations*, of objects with attributes; see some examples of the data set in Table 2.4. Those examples represent the relationships between objects and actions. The relations are characterized by two scores, Pointwise Mutual Information and the absolute frequency of co-occurrence in the world wide web data source.
2. Do K times:
   (a) Create ROAR R_k by populating it with G, erasing a random subset of attributes.
   (b) For each object in the ROAR, check whether its utility for mixing is correctly predicted.
3. The ROARs performance is the average of the correct-prediction rates over all objects in the ROAR.

To test the stability of the ROAR with respect to the population of objects, the above procedure can be repeated for various subsets of G.

Without bootstrapping (no ROAR) : The robot attempts to execute the action with the first object observed, but generally fails to achieve the desired result. All it can do is proceed trying the other objects in its workspace, one after the other, until it finds an object useful for the action required.

The system performance without ROAR is then simply given by the proportion of objects in the robots workspace that are in fact useful for mixing.

The ROAR can provide to a given object all other relating pairs with respect to a given action, and computes the confidences of the corresponding object-object-action ternary relation. An extract of the prediction result is shown in Table 2.5. The source of this prediction is the data set of the *web-relations* mentioned above.

Experimental setting

The input data, the annotated source, contains 19189 object-action pairs with the frequency of the co-occurrence in the world wide web data base, and an additional statistical measure, Pointwise Mutual Information to express the relationship within each of the pairs. Table 2.4 displays a slice of the full dataset. The input data is filtered only to achieve sufficient consistency otherwise it comprises a broad range of possible connections and some random noise as well. To improve the input data set requires further filtering, but that task is not part of the functionality of the ROAR, since it ought to work on any formally correct data base. Therefore the predicted results might reflect the noisiness of the input sources.

First the learning system of the ROAR establishes a binary relation between any two objects by comparing their connections to the related actions. The level of the connections is measured by the correlation between the scores of the shared actions. Since the input dataset is not complete, not all possible object-action pairs are included; therefore all missing action related object-object pairs are predicted, and finally the scores predicting the ternary, object-object-action, relations are derived.

The scores are based on the assumption that the values of the predicted scores are taken from a Gaussian distribution with given expected value and variance. Based on this conjecture a p value can be computed for each ternary relation, which can express the confidence in the corresponding prediction. These confidence values are returned as scores. Some examples with the highest scores of the output relations are shown in Table 2.5.

The entire table of the ternary relations contains 138967 items which can be stored in an SQL database. This database can then be queried for an object that most strongly relates to a given object with respect an action, or for the best applicable action connecting two given objects.

The test procedure outlined above has been repeated 5 times by applying 5-fold cross-validation. The known examples of the data were randomly cut into 5 folds, and in each run one fold is selected as test and all remaining as training. The average accuracy is measured by

$$\mathrm{ERR} = \frac{\mathrm{RMSE}}{\text{Score Range}} = \frac{\left(\sum_i (y_i - \hat{y}_i)^2\right)^{1/2}}{\max_j y_j - \min_j y_j}, \tag{2.1}$$

where index i runs over the test instances, index j runs over the training examples, y_i, y_j are the original values of the scores in testing and training respectively, and $\hat{y}_i$ denotes the predicted score on the test examples. The results are shown in the table below.

Score	Mean Accuracy (Standard Deviation)
Pointwise Mutual Information	0.064 (0.003)
Frequency of co-occurrence	0.083 (0.005)

Object	Action	Pointwise Mutual Inf.	Frequency
chair	take	1.3263	192977
table	set	1.0251	115054
table	leave	1.0945	104349
cup	take	0.4530	102246
cup	drink	1.7899	100161
cup	pour	1.9432	82714
table	create	1.4062	76001
fruit	eat	1.8377	71068
knife	take	0.5074	64253
cup	give	0.6992	61376
knife	have	0.2783	57321
table	clear	1.5715	56098
chair	occupy	2.3486	51208
table	give	-0.0842	50293
chair	hold	0.8307	49533
fruit	bear	2.1986	45625
cup	hold	0.5095	45585
tray	carry	2.6249	43562
knife	hold	0.9540	42312
table	construct	1.5507	40033
cup	bring	0.6302	39449
table	find	0.4126	39383
chair	pull	1.7299	37364
chair	have	-0.4491	36671
cup	fill	0.9582	36570
apple	eat	2.5208	35804
cup	hand	1.4147	34861
table	approach	1.4038	34207
table	contain	0.8769	30937
table	prepare	1.0308	29944
milk	add	2.1591	29121
milk	drink	1.8468	29118
chair	draw	1.4699	28742
chair	fill	0.9139	27571
juice	add	2.6489	27090
table	complete	1.5869	26586
plate	place	0.8138	25709
can	open	3.6008	25686
table	lay	1.1276	24155
knife	put	0.7094	23720
chair	leave	0.4246	23462
table	stand	0.9793	23334
knife	carry	0.8350	22494
chair	place	0.4177	22443
cup	put	0.1288	22303

Table 2.4: Source data set of benchmark (selection)

Object 1	Object 2	Action	Predicted score
slicer	cup	call	0.9963
cup	cornflakes	eat	0.9955
sifter	plate	take	0.9951
chair	squeezer	give	0.9947
table	squeezer	call	0.9946
table	squeezer	NONE	0.9946
colander	chair	have	0.9931
corn	fruit	reap	0.9930
sifter	table	give	0.9926
fruit	corn	reap	0.9924
⋮	⋮	⋮	⋮
table	stirrer	put	0.8999
tray	opener	take	0.8999
sieve	knife	have	0.8999
fork	spoon	take	0.8999
bowl	carrot	put	0.8999
can	corkscrew	take	0.8998
plate	blender	have	0.8998
table	fridge	buy	0.8997
pot	mixer	put	0.8997
knife	tomato	hold	0.8997
⋮	⋮	⋮	⋮
bowl	spoon	place	0.7999
stove	fridge	take	0.7999
tongs	plate	hand	0.7999
microwave	juice	take	0.7999
cleaver	table	NONE	0.7998
blender	cup	wash	0.7998
cup	nuts	put	0.7997
juice	whisk	give	0.7997
ladle	cleaver	have	0.7997
can	tray	carry	0.7997
⋮	⋮	⋮	⋮
plate	tray	contact	0.6999
fruit	bowl	add	0.6999
tongs	cup	snatch	0.6999
skimmer	pot	shout	0.6999
poacher	cereal	have	0.6999
scissors	table	NONE	0.6999
cup	tongs	bang	0.6999
mixer	cup	open	0.6999
ladle	juice	carry	0.6999
table	cleaver	grasp	0.6999
⋮	⋮	⋮	⋮
skimmer	apple	pull	0.5999
corn	spoon	put	0.5999
fork	cup	drop	0.5999
pan	cutter	move	0.5999
bowl	corn	order	0.5999
nuts	table	knock	0.5999
scissors	pan	grab	0.5999
dredge	pot	raise	0.5999
chair	scissors	prevent	0.5999
cutter	table	reciprocate	0.5999

Table 2.5: Combined “object-object-action-score” prediction results (selection)

2.2.4 Learning Pairwise Affordances

Previously we proposed that acquired knowledge about how objects behave under single-object actions help us more efficiently learn how they behave under paired-object actions. In other words, use of high-level object attributes learned for single-object actions should allow faster learning and generalization in predicting paired-object actions compared to use of low-level attributes only. Consider an example where the robot learns stackability affordances, i.e. learns to detect if two given objects would stably stack on top of each other. In this case, the robot needs to explore a high-dimensional search space if it learns from low-level shape features of these objects such as curvatures of different sides. On the other hand, if the robot uses previously learned high-level abstractions, such as rollability, it would learn which objects can be stacked on top of each other by associating rollability and stackability from fewer examples. This is achieved because the robot already learns and encodes part of object-robot-environment dynamics in the higher-level attribute of rollability, and can re-use this attribute to bootstrap other related learning problems that share similar characteristics.

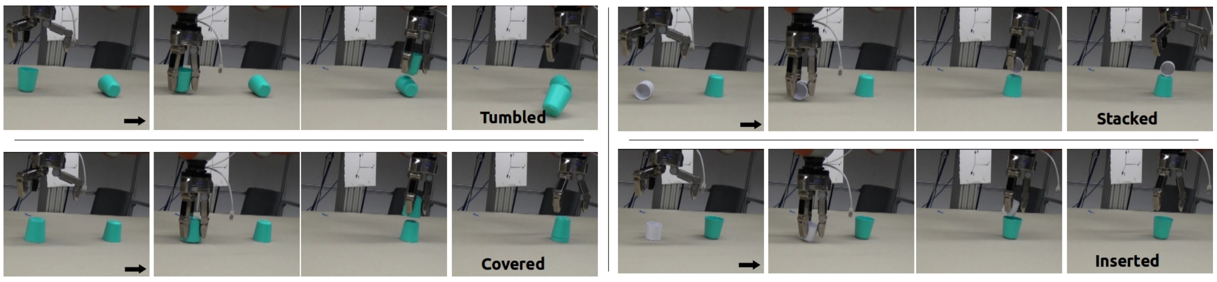


Figure 2.33: Sample snapshots from stacking interactions on various objects with the hand-arm robot used in UIBK.

To quantify the speed-up in learning, the hand-arm robot in UIBK robot (Figure 2.33) performed exploratory actions on single objects and pairs of objects of different shapes and sizes. The following actions are applied to objects with the following observed effects:

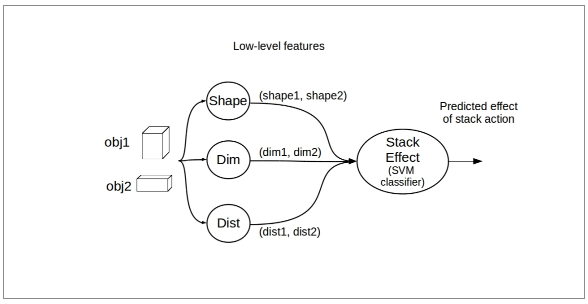
- Single-object actions: poke from top, poke from front, poke from side, grasp, release on table
- Effects of single-object actions: pushed, rolled, toppled, resisted, no-change
- Paired-object action: stack
- Effects of paired-object action: stacked, inserted, covered, tumbled

From interaction experience, the robot learns predicting effects of actions on objects given their visual features. The visual features of an object include dimensions of the object (dim), distribution of normal vectors obtained from local surfaces on the point cloud of the object (shape), and distribution of local distances of all pixels to the neighouring pixels (dist). Support Vector Machines are used to learn the mapping from object attributes to effect categories for each action. This deliverable analyzes the bootstrapping effect on pairwise affordances, therefore we will skip details of learning single-object actions, and focus on learning of paired-object actions, i.e. learning of effects of stack actions.

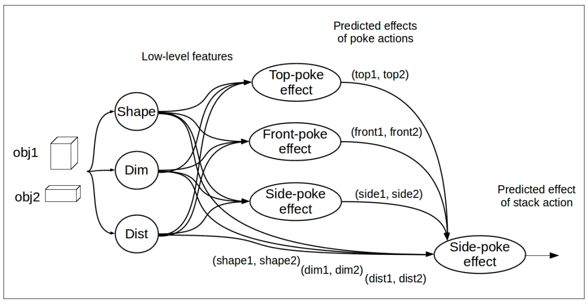
In order to test our hypothesis, we realized two effect prediction systems for stack action as follows:

- **No bootstrapping (NO-BOOT):** Effect of stack action on a pair of objects is directly predicted from visual features ((dims, shape, dist)) of both objects. Figure 2.34(a) shows this prediction scheme where visual features are used as inputs to SVM classifier/predictor.
- **With Bootstrapping (WITH-BOOT):** Effect of stack action on a pair of objects is predicted from visual features of both objects and the predicted effects of poke actions on these objects (Figure 2.34(b)).

We compared the learning speed of stack effect predictor in NO-BOOT and WITH-BOOT conditions using two different datasets. The first dataset, which includes small and diverse set of objects, is collected



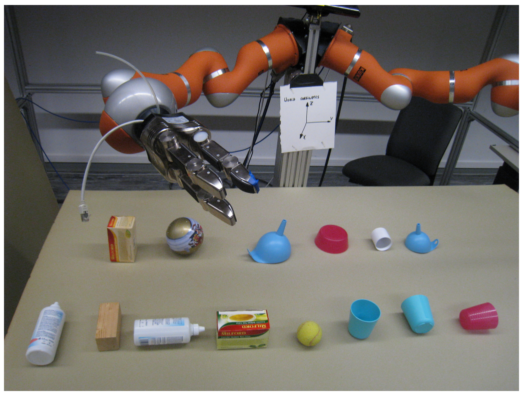
(a) No bootstrapping (NO-BOOT)



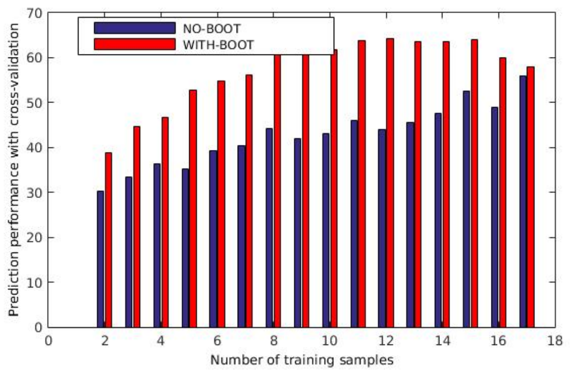
(b) With bootstrapping (WITH-BOOT)

Figure 2.34: Effect/affordance prediction systems. Each circle with an action corresponds to an SVM classifier that predicts the effect of the corresponding action. Note that this prediction structure, i.e. the links from poke action predictors to stack predictor, can emerge autonomously with an active learning approach using intrinsic motivation and feature selection [36].

through robot's physical exploration (Figure 2.35(a)). The robot executed stack action with 18 pairs of random objects. We evaluated the performance of these classifiers by systematically changing the size of the training set. For each training set size, we trained 10 classifiers using randomly selected samples. We tested each classifier using the remaining sample interactions. The bootstrapping effect is visible in the initial phases of learning where it is important to generalize to novel situations from small number of training samples. As visual features are real valued large sized vectors that encode object shape properties independent of robot-object dynamics, they require more training data for learning. On the other hand, effect predictions for poke action encodes more abstract information about robot-object dynamics, therefore provides more generalizable prediction performance with less data. With increasing number of training samples, performance of NO-BOOT approaches to performance of WITH-BOOT as abstract effect predictions are already computed from visual features and do not contain additional information.



(a) Small diverse set of objects



(b) Performance difference between NO-BOOT and WITH-BOOT

Figure 2.35: (b) The effect prediction performance of stack action that involves two objects with the small dataset provided in (a). The training of classifiers are done with the indicated number of samples (interactions) with either shape features or affordance features. The initial high performance of WITH-BOOT demonstrates the advantage of using bootstrapping. This figure is taken from [36].

In order to more rigorously validate our bootstrapping hypothesis, next, we used a dataset that includes 83 objects (Figure 2.36 and 83×83 effects created on these pairs of objects by the stack action. In order to collect such a dataset, the robot, for example, was required to make 6889 interactions for an action that involves two objects, which is not feasible in the real world. Thus, we used a human expert to fill-up the effect field of the complete table. We compared the learning speed with (WITH-BOOT) and without (NO-BOOT) use of inputs from other effect predictions in predicting stack effects. We performed 50 learning runs both WITH-BOOT and NO-BOOT in learning stack effect predictions. In this setting, we can see a significant bootstrapping effect in learning as shown in Figs. 2.37(a) and 2.37(b). Mean and standard deviation of the group of trained predictors of WITH-BOOT and NO-BOOT are shown with the bold line and the filled areas. As shown in the figures, initial and final mean accuracies and variances of two different cases are same. Still, the expected bootstrapping effect in the beginning of the learning steps are clearly visible in both figures. At the bottom of each figure, the difference in accuracies between WITH-BOOT and NO-BOOT is given, which is confirmed with double-side t-test with $p < 0.01$. The boxes at the bottom in Figure 2.37(a) show that the accuracy of WITH-BOOT becomes 10% higher with 40 training samples, compared to NO-BOOT case with significance level of $p < 0.01$. As shown, the bootstrapping effect quickly increases in the beginning and remains same for some time. The bootstrapping effect is more significant when the accuracy is computed with novel pairs. Novel pairs of objects (o1, o2) for stack action corresponds to objects, where o1 and o2 have never been experienced in bottom and top roles during stacking, respectively. The effect becomes visible after 18 pairs instead of 22 pairs; and increases to a maximum value of 13% compared to 10%. This also shows that use of affordances as inputs in learning and predicting other affordances provides significant generalization capabilities. Preliminary version of these results were previously presented in [37] and has been recently submitted to a journal [UP15].

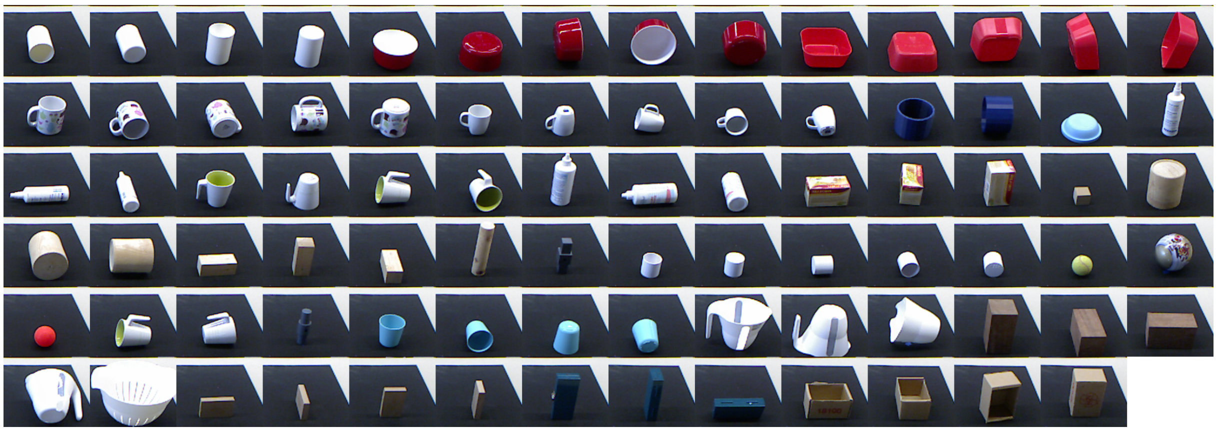
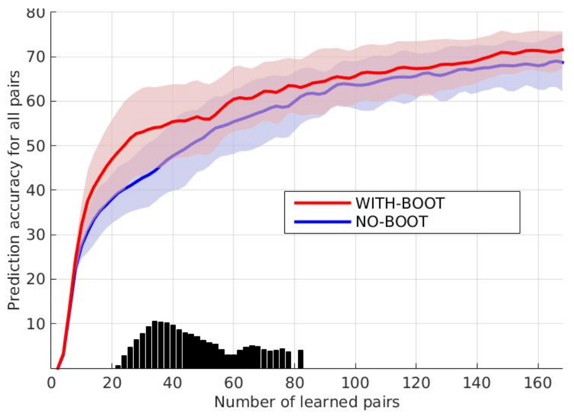
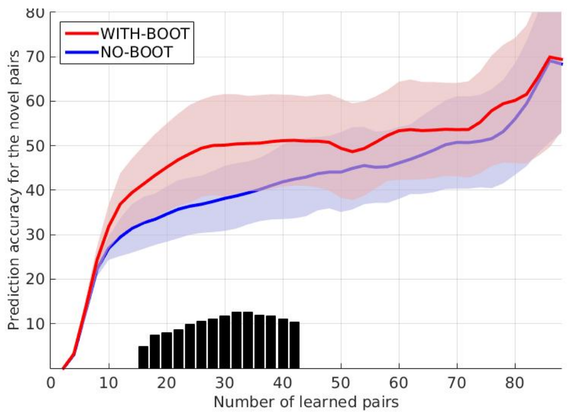


Figure 2.36: Objects from the large dataset.



(a) Test with all pairs.



(b) Test with novel pairs.

Figure 2.37: The evolution in prediction accuracy of stack effect predictor. The accuracy is computed by using all object pairs in (a) and by using only the novel object pairs in (b). The black bars show the amount of difference in prediction accuracies between WITH-BOOT and NO-BOOT, in significance level of $p < 0.01$.

2.3 Planning

2.3.1 high-level symbolic planning

Benchmarking of high-level symbolic planning components usually follows the standard approach taken by the International Planning Competition [2] (see, e.g., [9]), where the two main metrics for evaluation are planning time and plan quality. Planning time is a measure of the raw speed of the planning process applied to a particular planning domain and problem, and usually includes preprocessing, plan generation, and postprocessing time. Plan quality attempts to measure how close the generated plan is to the "ideal" plan that could be (theoretically) generated. Plan quality is usually assessed against a secondary metric (e.g., length, cost) which may have a large effect on the planning process (especially the planning time). (A third metric which combines planning time and plan quality in a weighted average is also sometimes used.) In both cases, planners are usually compared against other available planners on the same set of planning domains.

When a planner is included as a component in a larger integrated system, the traditional planning metrics may have to be considered in the larger integration context. For instance, longer planning times may be acceptable provided such behaviour doesn't negatively affect the overall response time of the integrated system. Additional factors, such as the syntactic form of planning actions or the specification of domain-dependent knowledge, also affect the planning process. In particular, the tradeoff between the expressiveness of the planning representation and the time a planner takes to generate plans, is often a significant factor in the results produced by the standard metrics. Thus, benchmarking also involves analysing the scalability of particular aspects of a planning domain using a given planner, and the identification of techniques to reduce the overall complexity of the planning domain (e.g., introducing domain control knowledge, restricting action models, etc.).

As part of UEDIN's work in WP3.2, benchmarking of symbolic planning using the PKS planner [20] [21] has mostly involved assessing the quality of the plans that are generated in the project's domains, given acceptable bounds on the planning time, and understanding the scalability of the main planning domains used on the project. Since the work on PKS has typically focused on issues related to planning problem representation, rather than raw computational speed, the role of factors like commonsense domain control knowledge is particularly important.

For instance, Figure 2.39 shows the results of planning in an open-ended version of the table setting domain from Scenario 2 (WP5), where the goal is to set a number of place settings (#PS).

					Planning time (s)		
#PS	#L	#O	≈#States	#Actions	SGPLAN	FF	PKS
1	5	3	125	9	0.02	0.00	0.00
2	5	6	15625	18	0.03	0.01	0.13
3	5	9	1 953 125	28	12.18	20.37	24.31
3 (+1)	5	10	9 765 625	31	2999.86	3543.73	3976.35
3 (+2)	5	11	48 828 125	35	—	—	—
4	5	12	244 140 625	37	—	—	—

Figure 2.38: results of planning in an open-ended version of the table setting domain from Scenario 2 (WP5).

Three objects must be set for each place settings (e.g., a plate, a knife, and a fork) from a given number of objects available (#O). Any object can be moved to any location within a given number of discrete locations in the domain (#L). The domain includes three basic actions (grasp, putdown, and move). (Details about the planning domain can be found in deliverable D3.2.4.) The number of planning states (#States) and the minimum number of actions required in a successful plan (#Actions) are shown for a given planning problem (i.e., a given tuple (#PS,#L,#O)). The planning time (in seconds) is also shown for three planners, including SGPLAN ([11]), FF ([10]), and PKS. (SGPLAN and FF are provided to give indicative measures of the planning time using other off-the-shelf planners that are optimised for classical planning problems of this form.) These results illustrate the exponential growth of the state space for larger problem instances, and the effect this has on the planning process: moving from 3 place settings to 3 place settings plus 1 extra object results in a jump in planning time from 24.31 seconds to 3976.35 seconds using PKS, with similar results for the other planners. This is mainly due to the massive

[2](IPC: (`http://www.icaps-conference.org/index.php/Main/Competitions`)

parallelism in the structure of the domain (e.g., every object can always be moved to any location in the domain) so there is little structure for the planner to exploit, leading to difficulty for modern planning heuristics.

By contrast, Figure 2.39 shows the results of adding commonsense domain control knowledge (CSK) to the PKS planning domain.

			PKS planning time (s)	
#PS	**#L**	**#O**	**No CSK**	**CSK**
1	5	3	0.00	0.001
2	5	6	0.13	0.005
3	5	9	24.31	0.031
3 (+1)	5	10	3976.35	0.063
3 (+2)	5	11	—	0.092
4	5	12	—	0.121
5	6	15	—	1.496
6	7	18	—	23.830
7	8	21	—	487.240

Figure 2.39: the results of adding commonsense domain control knowledge (CSK) to the PKS planning domain.

In particular, two simple rules are encoded as part of the domain description: (1) once an object is placed, do not remove it, and (2) finish setting all the objects of a particular type before moving to a new type. These rules have the effect of adding structure to the domain by enforcing an ordering on certain placement actions, while prohibiting object removals, leading to a pruning of the underlying search space. This has a marked effect on the planning process, allowing PKS to scale to much larger problem instances with reasonable planning times (e.g., 5 place settings in under 2 seconds, and 6 place settings in under 24 seconds). These results also motivate the need for structural bootstrapping techniques of the kind considered on this project in order to learn appropriate commonsense control knowledge based on prior experience.

Additional analyses in this benchmarking study are expected to be published by UEDIN beyond the end of the project, along with a proposal to submit the table setting domain as a challenge problem for the International Planning Competition.

References

[1] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *IEEE International Conference on Computer Vision*, 2007.

[2] Carlo Ciliberto, Sean Ryan Fanello, Lorenzo Natale, and Giorgio Metta. A heteroscedastic approach to independent motion detection for actuated visual sensors. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3907–3913. IEEE, 2012.

[3] Carlo Ciliberto, Sean Ryan Fanello, Maurizio Santoro, Lorenzo Natale, Giorgio Metta, and Lorenzo Rosasco. On the impact of learning hierarchical representations for visual recognition in robotics. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3759–3764. IEEE, 2013.

[4] Andr ckermann, Robert Haschke, and Helge Ritter. Real-Time 3D Segmentation of Cluttered Scenes for Robot Grasping. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, 2012.

[5] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cdric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[6] M. Deniša, A. Gams, A. Ude, and T. Petrič. Generalization of discrete compliant movement primitives. In *International Conference on Advanced Robotics (ICAR)*, pages 565–572, Istanbul, Turkey, 2015.

[7] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[8] D. Forte, A. Gams, J. Morimoto, and A. Ude. On-line motion synthesis and adaptation using a trajectory database. *Robotics and Autonomous Systems*, 60:1327–1339, 2012.

[9] Jörg Hoffmann, Stefan Edelkamp, Sylvie Thiébaux, Roman Englert, Frederico dos S. Liporace, and Sebastian Trüg. Engineering benchmarks for planning: the domains used in the deterministic part of IPC-4. *J. Artif. Intell. Res. (JAIR)*, 26:453–541, 2006.

[10] Jörg Hoffmann and Bernhard Nebel. The ff planning system: Fast plan generation through heuristic search. *J. Artif. Int. Res.*, 14(1):253–302, May 2001.

[11] C. W. Hsu, Wah R., and Chen. New features in SGPlan for handling soft constraints and goal preferences in PDDL 3.0. In *In Proceedings of the 5th International Planning Competition, International Conference on Automated Planning an Scheduling*, pages 39–41, 2006.

[12] The icub datasets. `http://www.iit.it/en/projects/data-sets.html`.

[13] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge S&#x00E1;nchez, Patrick Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(9):1704–1716, September 2012.

[14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA, 2014. ACM.

[15] Peter Kaiser, Mike Lewis, Ronald Petrick, Tamim Asfour, and Mark Steedman. Extracting common sense knowledge from text for robot planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3749–3756, 2014.

[16] G. Kootstra, J. Ypma, and B. de Boer. Active exploration and keypoint clustering for object recognition. In *icra*, Pasadena, CA, 2008.

[17] Gert Kootstra, Mila Popovic, Jimmy Alison Jørgensen, Norbert Krüger, and Danica Kragic. Enabling grasping of unknown objects through a synergistic use of edge and surface information. *International Journal of Robotics Research*, 2012.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[19] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV*, pages 143–156, 2010.

[20] Ronald P. A. Petrick and Fahiem Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2002)*, pages 212–221, Menlo Park, CA, April 2002. AAAI Press.

[21] Ronald P. A. Petrick and Fahiem Bacchus. Extending the knowledge-based approach to planning with incomplete information and sensing. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 2–11, Menlo Park, CA, June 2004. AAAI Press.

[22] T. Petrič, A. Gams, L. Žlajpah, and A. Ude. Online learning of task-specific dynamics for periodic tasks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1790–1795, San Francisco, CA, 2014.

[23] Andreas Richtsfeld, Thomas Mörwald, Johann Prankl, Michael Zillich, and Markus Vincze. Segmentation of unknown objects in indoor environments. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4791–4796. IEEE, 2012.

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[25] D. Schiebener, J. Morimoto, T. Asfour, and A. Ude. Integrating visual perception and manipulation for autonomous learning of object representations. *Adaptive Behavior*, 21(5):328–345, 2013.

[26] D. Schiebener, A. Ude, and T. Asfour. Physical interaction for segmentation of unknown textured and non-textured rigid objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, 2014.

[27] David Schiebener, Julian Schill, and Tamim Asfour. Discovery, segmentation and reactive grasping of unknown objects. In *Proc. 12th IEEE-RAS International Conference on Humanoid Robots*, Osaka, Japan, 2012.

[28] David Schiebener, Aleš Ude, Jun Morimoto, Tamim Asfour, and Rüdiger Dillmann. Segmentation and learning of unknown objects through physical interaction. In *2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 500–506, Bled, Slovenia, 2011.

[29] G. Schreiber, A. Stemmer, and R. Bischoff. The fast research interface for the KUKA lightweight robot. In *ICRA Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications – How to Modify and Enhance Commercial Controllers*, Anchorage, Alaska, 2010.

[30] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR 2014)*. CBLS, April 2014.

[31] Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3):411–426, March 2007.

[32] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.

[33] Andrea Tacchetti, Pavan Kumar Mallapragada, Matteo Santoro, and Lorenzo Rosasco. GURLS: a least squares library for supervised learning. *CoRR*, abs/1303.0934, 2013.

[34] Mikkel Tang Thomsen, Dirk Kraft, and Norbert Krüger. Identifying relevant feature-action associations for grasping unmodelled objects. *Paladyn. Journal of Behavorial Robotics*, 6(1):85–110, 2015.

[35] M.T. Thomsen, D. Kraft, and N. Kruger. A semi-local surface feature for learning successful grasping affordances. In *VISAPP International Conference on Computer Vision Theory and Applications. 2016.*, 2016.

[36] E. Ugur and J. Piater. Emergent structuring of interdependent affordance learning tasks. In *IEEE Intl. Conf. on Development and Learning and on Epigenetic Robotics (ICDL-Epirob)*, 2014.

[37] E. Ugur, S. Szedmak, and J. Piater. Bootstrapping paired-object affordance learning with learned single-affordance features. In *IEEE Intl. Conf. on Development and Learning and on Epigenetic Robotics (ICDL-Epirob)*, 2014.

[38] R. Vuga, B. Nemec, and A. Ude. Enhanced policy adaptation through directed explorative learning. *International Journal of Humanoid Robotics*, 12(03), 2015.

[39] Y. Wang, F. Gao, and F.J. Doyle. Survey on iterative learning control, repetitive control, and run-to-run control. *Journal of Process Control*, 19(10):1589–1600, 2009.

[40] F. Wörgötter, C. Geib, M. Tamosiunaite, E.E. Aksoy, J. Piater, Hanchen Xiong, A. Ude, B. Nemec, D. Kraft, N. Krüger, M. Wächter, and T. Asfour. Structural bootstrapping; a novel, generative mechanism for faster and more efficient acquisition of action-knowledge. *IEEE Transactions on Autonomous Mental Development*, 7(2):140–154, 2015.

[41] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 1794–1801, 2009.

Attached Articles

[FKKG15] Severin Fichtl, Dirk Kraft, Norbert Krüger, and Frank Guerin. Bootstrapping the learning of means-end action success predictions. Technical Report 2015–1, Cognitive and Applied Robotics Group, The Maersk Mc-Kinney Moller Institute, University of Southern Denmark, 2015.

[FNU15] D. Forte, B. Nemec, and A. Ude. Exploration in structured space of robot movements for autonomous augmentation of action knowledge. In *The 17th International Conference on Advanced Robotics (ICAR)*, pages 237–243, Istanbul, Turkey, 2015.

[GPD$^+$16] A. Gams, T. Petrič, M. Do, B. Nemec, J. Morimoto, T. Asfour, and A. Ude. Adaptation and coaching of periodic motion primitives through physical and visual interaction. *Robotics and Autonomous Systems*, 75, Part B:340 – 351, 2016.

[PCG$^+$15] T. Petrič, L. Colasanto, A. Gams, A. Ude, and A. J. Ijspeert. Bio-inspired learning and database expansion of compliant movement primitives. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 346–351, Seoul, Korea, 2015.

[UP15] Emre Ugur and Justus Piater. Emergent structuring of interdependent affordance learning tasks using intrinsic motivation and empirical feature selection. *IEEE Transactions on Neural Networks and Learning Systems*, 2015. submitted.

Cognitive and Applied Robotics Research Group
The Maersk Mc-Kinney Moller Institute
University of Southern Denmark

Technical Report no. 2015 – 1

Bootstrapping the Learning of Means-End Action Success Predictions

Severin Fichtl, Dirk Kraft, Norbert Krüger and Frank Guerin

Dec 18, 2015

Title	Bootstrapping the Learning of Means-End Action Success Predictions Copyright © 2015 Severin Fichtl, Dirk Kraft, Norbert Krüger and Frank Guerin. All rights reserved.
Author(s)	Severin Fichtl, Dirk Kraft, Norbert Krüger and Frank Guerin
Publication History	Dec 2015. A CARO/MMMI Technical Report

Bootstrapping the Learning of Means-End Action Success Predictions

Severin Fichtl[1,2], Dirk Kraft[2], Norbert Krüger[2] and Frank Guerin[1]

***Abstract*—Robots acting in everyday environments need a good knowledge of how a manipulation action will affect objects in a relationship, such as 'inside' or 'behind' or 'on top'. We investigate how this could be learnt by a robot from its own action experience. A major challenge in this approach is to reduce the number of training samples needed to achieve accuracy, and hence we investigate an approach which can leverage past knowledge to accelerate current learning (which we call bootstrapping). We learn Random Forest based action success predictors from visual inputs and demonstrate two approaches to knowledge transfer for bootstrapping. In the first approach to bootstrapping, the state space for a new predictor is augmented with the output of previously learnt predictors. In second approach to bootstrapping, we learn categories that capture underlying commonalities of a pair of existing predictors and augment the state space with this category classifier's output. In addition, we introduce a novel heuristic, which suggests how a large set of potential categories can be pruned to leave only those categories which are most promising for bootstrapping future actions. Our results show that we can autonomously learn categories, and that bootstrapping learning of actions using these categories outperforms learning without bootstrapping if the learning problem is hard.**

I. Introduction

We are interested in grounding knowledge in a robot's own sensorimotor experience, an accepted principle of developmental robotics, justified e.g. in [1]. A major problem with this approach is that it takes a long time to learn through robot experience. The state of the art in artificial development and learning methods does not permit a robot to learn from experience as rapidly as an infant. This mirrors problems in other areas of artificial intelligence such as speech recognition, where systems need orders of magnitude more data to learn from than a small child is exposed to [2]. The added difficulty in robotics is that we do not have a large data set to run the learning algorithm on, a robot must first go through the slow process of trying actions out in the world. This problem of slow learning has generated interest in methods that can bootstrap learning [3], [4], [5]. The basic idea is that if we have some prior knowledge, we should be able to learn similar things faster, i.e. bootstrap the learning.

In this paper, we focus on bootstrapping the learning of one part of a robot's knowledge that is important for object manipulation, that is the knowledge about spatial relationships that determine the outcome of actions on object pairs. However, the general approach described here is also relevant for other bootstrapping tasks where prior knowledge is existing and rapid learning from few samples is desired.

*This work was supported by UK EPSRC and the EU Cognitive Systems project XPERIENCE (FP7-ICT-270273)

[1]University of Aberdeen, Aberdeen, United Kingdom (f.guerin@abdn.ac.uk)

[2]University of Southern Denmark, Odense, Denmark (fichtl@mmmi.sdu.dk, kraft@mmmi.sdu.dk, norbert@mmmi.sdu.dk)

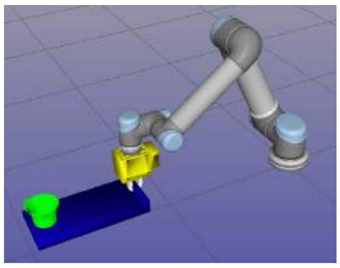


Fig. 1. Illustration of the Robot Simulation environment with the robot attempting to pull a shelf-like object with a cup on top of it.

Spatial relationships are extremely important for robotic manipulation e.g. in service robots, to determine the outcome of actions in everyday home environments; for example, trying to reach an object that is partly obstructed by another object, or pulling or lifting an object when another object is on top or inside of it (see Fig. 1). These are the kinds of situations that children are competent with, and in looking at how infants first acquire this knowledge, we see the close connection with means-ends behaviours involving more than one object (i.e. when one action is used as a means to achieve some other end goal). Infants start to appreciate the importance of spatial relationships when they begin exploring means-end actions at around eight months [6], [7], [8], and means-end behaviour is the start of planning: how to use one action as a step to achieve some more distant goal. Hence learning how spatial relationships determine the outcome of actions is a crucial part of the problem of robots learning planning operators for manipulation. To demonstrate our bootstrapping approach, we therefore use this particular problem.

Our key technique for bootstrapping of this learning is inspired by works in cognitive science, where symbolic knowledge is learnt from interaction and can then be reused where it benefits subsequent learning, for example the 'synthetic item' of Drescher and Chaput [9], [10]. These techniques have not been applied to high dimensional robot manipulation scenarios to the best of our knowledge (Ugur et al. [4] being a very recent exception). In our work, we learn categories from early experience, where these categories correspond to spatial relationships like 'on top' or 'inside'. Note that these categories are not imposed by the human designer, but rather are learnt by a classifier with the task to discriminate situations from visual input, where a certain action will have one outcome or another. These categories can then be used as a binary input to subsequent learning, resulting in a speed up where they have discriminative power. We learn from simulated robot actions as this was the only feasible way to generate the thousands of examples we needed to compare different approaches.

Bootstrapping (in the sense used here) is a kind of transfer learning [11], and while transfer can give a massive performance boost, it is also well known that it can equally well reduce performance via negative transfer [11]. For this reason we believe that it is important to evaluate a bootstrapping technique not just on one carefully chosen example, but on a reasonable range of examples, in order to develop an understanding of where it is likely to work best and where it will not. Therefore we have attempted to learn a variety of categories (36) and used them to bootstrap the learning of a range of action success predictors (9). In this, our work stands out from related work on bootstrapping like [3], [4] where the authors focus on single instances of transfer.

Our exhaustive approach to creating categories from action pairs leads to a large amount of category classifiers, some of which may be good for bootstrapping and some not (a problem which would be worse in, e.g. a realistic service robot system with many more actions). Rather than trying out all categories for bootstrapping there is a need for a method to prune possible categories and arrive at a small set which is most promising for bootstrapping. To achieve this, we introduce a novel heuristic based on how correlated existing action success predictions are. This is based on the intuition that there exist some underlying physical relationships in the world which are implicitly captured by some action success predictors, and are common for some actions, and that these relationships are important to physical causality and are likely to be useful to determine the success of future actions. Our heuristic enables us to prune away $3/4$ of the candidate categories without losing bootstrapping performance.

As a representation for learning, we use Random forests. This allows us to compare category based bootstrapping as described above with what action success prediction based bootstrapping in which an already learned action success predictor is directly fed into the input of a Random Forest classifier for a new action. We show that this action success prediction based approach is nearly as effective as the category based approach, which indicates that category formation – although required for higher level transfer learning involving, e.g., language – might not be necessary for transfer on such a low motor–sensorial level as dealt with in this paper.
To summarize the achievements of this work:

- We realize a system in which autonomous category formation on spatial object relations for action outcome prediction is performed during robot exploration.
- We show that by exploiting these categories, we can efficiently bootstrap the learning of action outcome predictions leading to a significant speed up.
- We show that by applying a heuristic for selecting , we can circumvent the inherent problem of accumulating a large number of highly correlated categories.
- We compare the category based bootstrapping with action success prediction based bootstrapping approach, which leads to similar bootstrapping performance than category based learning.

The remainder of this paper is structured as follows: Section II reviews the literature. Sections III and IV describe in more details the methods and experimental setups used. Section V presents the results of this work. Section VI discusses our work in a broader context.

II. Literature Review

In this section we first sketch the big picture of cognitive development and describe where our work fits within this. We then focus on the specific problem we tackled and review closely related approaches.

This work has been inspired by the cognitive development of human infants. Infants undergo rapid development from simple action schemas such as sucking or banging objects at six months, to solving relatively complex problems such as simple tool use at two years of age [6], [12]. Two-year-old infants are clearly capable of reasoning about objects, spatial relations and actions effects and are able to come up with plans to effectively reach their goals. Their knowledge is at a relatively high level of abstraction because they easily generalise across a wide variety of everyday situations. Psychologists have described their knowledge structures as "schemas" (or various similar terms, see [8]) which are roughly analogous to the "planning operators" of Artificial Intelligence, because there are situations which make them likely to be executed (like the precondition of a planning operator), and expected effects (postcondition), as well as some motor control program describing the behaviour executed. Two-year-olds seem to possess a sizable repertoire of schemas whereas six-month-olds seem to have a relatively impoverished repertoire. One of the essential mysteries of cognitive development is how to account for the acquisition of this large repertoire of relatively abstract knowledge, based on concrete action experiences.

Within this development a particularly interesting stage concerns the acquisition of means-ends behaviours which begins about 8 months (i.e. where one action is used in order to facilitate another [13]), because it is through learning means-ends behaviours that infants begin to learn about relationships between objects [14]. For example in pulling a supporting object (e.g. cloth) to retrieve a more distant object (e.g. toy) that is on top, the precondition that determines success of the action must capture the relationship between the objects. Much of the child's learning up to this point is related to relationships with its own body (subjective), such as something being reachable, or suckable. However the relationship "on top" marks a beginning of learning more objective properties of the world. Probably this is initially learnt in quite a context-bound manner, but gradually it will be generalised. According to Mandler's analysis [15] infants begin with 'perceptual knowledge', which is implicit in individual schemas, and develop towards more abstract 'conceptual knowledge'. Conceptual knowledge is essential for more advanced problem solving. The mechanism may also be the same as, or at least closely related to, "representational redescription" [16]. As infants make this transition they begin to see things at a higher level of abstraction, seeing precisely those relationships which are important in determining what object manipulations are possible (by the infant or other agents). For this reason we have explored the acquisition of chunks of knowledge that are locked in the context of a particular manipulation, as well as more abstract chunks of knowledge that begin to capture a category like "on top". We explore the use of both for bootstrapping subsequent acquisitions.

Although there are works which learn planning operators [17], [18], [19], [20], in this paper we focus only on learning the precondition for a new behaviour (i.e. learn to predict action success in a given state). This is quite close to work on learning relational 'affordances'. There is rather a lot of work on affordances, but it has been noted by Moldovan et al. [21] that there is very little work on relational affordances (i.e. the actions afforded by a pair of objects in a particular spatial relationship). In their work the relational features considered are relative distance between two objects, the relative orientation of one with respect to the other, and whether or not they are touching. We go for a much richer relational description looking at much finer grained details of the objects, which can for example capture such things as one object having elements surrounding another, or in front of another.

Ugur et al. [4] learn 'paired object affordances' for the action of stacking. The features input to the learner are histograms of normal vectors for various points on an object's surface. They learnt to predict the effect the effect of a stacking action, given the visual features of the pair of objects being stacked. A related affordance learning approach is that of Griffith et al. [22] which focused on learning features which could predict if an object could serve as a container, based on the effect of exploratory actions. There is a significant difference with our work in that we are looking at objects already in a relationship, in order to determine the effect of an action, whereas Ugur et al. or Griffith et al. are looking at the features of objects before they are put in a relationship, in order to determine what relationship they might end up in after an action.

Our histogram approach is inspired by the approach of Mustafa et al. [23], which considers relationships between surface patches (distances and angles) in a single object. These histograms characterise the object and are quite robust to variations in viewpoint. We also consider relationships among surface patches; however we look at a pair of objects, considering the relationship between every patch on the first object with every patch on the second. Our histograms characterise the relationship between the objects [24], [25], [26]. We are not aware of any other work which uses a feature computed from relationships among parts of two different objects.

Rosman and Ramamoorthy [27] learn spatial relationships between objects using a support vector machine based approach, where support vectors are picked for their ability to differentiate the point cloud into two objects. This has the effect that the subset of points considered by the classifier are on the edges of the object. Relations are then learnt based upon the relative positions of clusters of the support vectors. Our histogram based approach preserves more information about the relations between objects in the scene, whereas much of this information is discarded by Rosman et al. [27] and also the above approach of Moldovan et al. [21]. We consider not just the border between objects, but all patches on each object; this could be important for example when a small object is near to one edge of a large one, in a containment relationship.

A further work on support relations is by Panda et al. [28]. The work exploits a number of visually derived features regarding the relationship between the objects: proximity, boundary overlap, depth boundary, containment, relative stability. In addition, a rule based method is employed to infer what supports what, when multiple objects are stacked or leaning on each other. In contrast to this work, we have approached the problem more from a developmental robotics perspective; we are attempting to see what the system can learn without significant prior knowledge, and learning from the effects of its actions. The reasons for our preferring the developmental approach have been discussed elsewhere [29], [8].

All of these works which can recognise containment or support relationships (and implied affordances) have importance beyond the task of informing a robot of action possibilities. Affordance work is beginning to be used to help solve classic computer vision problems such as object categorisation [30] which take the approach of imagining an actor exploiting the affordance defining the object. Also it is highly relevant to learning about human activities from observations, for the purposes of imitation or understanding [31], [32]. These works use semantic scene graphs and can benefit from accurate descriptions of spatial relationships between objects within these graphs.

A second aspect of our work which deserves comparison with others is bootstrapping. As stated in Sec. 1, what we mean by 'bootstrapping' is a type of transfer learning: "*transfer learning* aims to extract the knowledge from one or more *source tasks* and applies the knowledge to *a target task*" [11]. In our work our feature space is the same for our source and target domains, but the source and target tasks are different, making this an example of 'inductive transfer' [11]. In terms of the detailed taxonomy on transfer learning by Lazaric [33] ours is 'learning speed improvement'; i.e. a reduction in the amount of experience required to learn the solution, which is distinct from transfer approaches which improve the asymptotic performance, or which 'jumpstart' to give better performance at the first attempt on the new (target) task; i.e. before training.

There are also closely related bootstrapping approaches in the developmental robotics literature. Do et al. [3] present a case study of learning to wipe a table, where they show that 'mixing' experience (e.g. a cake mix) can be reused to bootstrap the learning of 'wiping'. They use sequences of objects getting in contact and losing contact to assess action similarity. While the demonstrated example shows strong positive transfer, we believe it is important to extend such studies to a larger variety of actions, to gain knowledge of where transfer works and where it may not bring any benefit or even reduce performance. In our work we demonstrate both positive and negative bootstrapping effects on a variety of nine different actions.

In another recent 'bootstrapping' approach, Ugur et al. [4] bootstrap the learning of a 'stacking' affordance by first learning a 'rolling' affordance They first learn for a set of individual objects if the object has the 'rollable' affordance by executing preprogrammed 'poking' actions. This affordance knowledge helps the robot to quickly learn whether two objects are stackable. Two 'rollable' objects are less likely to stand on top of each other, than two objects that are not 'rollable'. However, there are also affordances where knowledge of one is unlikely to bootstrap the learning of the other, e.g. 'rollable' and 'graspable' where the former depends mainly on the shape of the object (e.g. sphere versus cube) and the latter mainly on the size (e.g. fits in gripper versus does not fit in gripper). Ugur et al.'s work is in the same spirit as our approach; where they have 'rollable' as an input to the second stage of learning, we have several categories. We have attempted to learn a variety of categories because the robot does not know in advance which, if any, might be useful in later stages of learning.

In fact the work of Ugur et al. [4] above is attempting something of much broader scope than what we tackle in this paper; they attempt is staged development where there are successive developments which build on each other. This is an area of great interest to developmental robotics, and there are several more examples [34], [35], [36], [37]. One area of interest within this is whether the stages need to be pre-scripted [34], or could emerge naturally, e.g. as a consequence of some relatively simple intrinsic motivation mechanism [38]. We see our work as contributing to this area because we believe spatial relationship categories learnt could influence subsequent behaviour and the experiences generated (e.g. the robot creates "on top" and "inside" situations), however this has not been pursued yet.

We do not feel that our work is particularly close to computer vision work in scene understanding (E.g. [39]) because those works typically recognise all objects, and then can use higher level knowledge to assist in understanding. Our work in contrast is at a lower level, and is more concerned with the physical relationships among surfaces without regard for object knowledge. We think of it more like how an infant might recognise simple physical relationships between household objects without any idea of what their names are or what their typical purposes are.

The following section will describe in more detail our approach to bootstrapping the learning of action success predictors.

III. Methods

In this section, we describe our methods for using additional knowledge for bootstrapping the learning of a classifier (Section III-A), how to generate categories used as additional knowledge (Section III-B) and how we measure the effect of additional knowledge on learning performance (Section III-C).

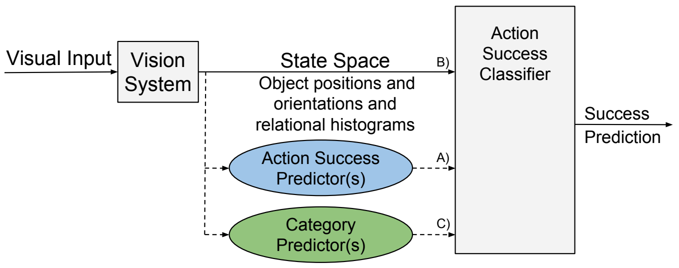

Visual Input
Vision System
State Space
Object positions and orientations and relational histograms
Action Success Predictor(s)
Category Predictor(s)
B)
A)
C)
Action Success Classifier
Success Prediction

Fig. 2. Illustration of the inputs for learning action success classifiers. The solid lines represent the basic case of learning with no additional information. To bootstrap the learning additional information can be added. The blue ellipse illustrates the addition of one, two or all other existing actions' success predictions. The green ellipse illustrates the addition of one or all previously learned category predictor outputs.

A. Learning Action Success Predictors

The goal of the action success classifiers trained in this work is to accurately predict whether the action associated with the classifier can be executed successfully in a given scene. The actions are (with one exeption) means-end actions where one action (means) is executed on one object in order to facilitate the successful execution of another action (end) on another object. It is, therefore, the spatial relationship between the objects involved, that is determining whether the means action can succeed to facilitate the end action. To learn the mapping between the current state of the environment and the means action success, the classifiers are trained using a state space description including the positions of both objects relative to the robot and information describing the spatial relationship between the objects. In this paper, we investigate how additional knowledge can be used to speedup the learning of action success classifiers.
Throughout the paper, we will refer to the different classifiers with the following terminology:
Action Success Classifier: The classifier that is currently learnt with the goal of predicting the success of a new action.
Action Success Predictor: Already existing classifiers that predict the success of other existing actions. Their output is used to bootstrap the learning of a new action classifier.
Category Predictor: Classifiers that recognise specific categorical patterns in the environment. The category predictor's output is used to bootstrap the learning of a new action classifier.

Fig. 2 illustrates our general structure of learning. We investigate the following approaches to learning action success classifiers:

B) basic learning with no additional information (represented by the solid line in Fig. 2)
A) using previously learned action success predictor(s) for other actions as additional input
C) using previously learned category predictor(s) as an additional input

In the following subsections we describe in more detail these different approaches to bootstrapping.

1) Basic Learning Without Bootstrapping (B): Learning without bootstrapping corresponds to the typical isolated learning of an individual action success classifier. Here, for learning to predict the success for different actions, each action success classifier receives as input a set of visually derived descriptors of a scene before an action execution trial and a success label differentiating successful trials from unsuccessful ones. The classifier learns a mapping from these visual inputs to success labels. The combination of all the inputs that go into the classifier, apart from the success label, define the state space (see Fig. 2).

2) Bootstrapping With Already Learnt Action Success Predictors As Knowledge Source (A): Our approach to bootstrapping is based on extending the state space of a new action success classifiers with the output of an action success predictor learnt for already existing other actions. If the new action is related to an already existing action, the state space extention can carry relevant information and hence bootstrap the learning.

We investigate three different variations of action success prediction based bootstrapping: Adding the success prediction(s) of

A1) a single already existing action success predictor,
A2) two already existing action success predictors,
An) all already existing action success predictors

as additional inputs to the action success classifier under training.

3) Bootstrapping With Automatically Created Categories As Knowledge Source (C): The approach to bootstrapping followed here is extending the state space of a new action success classifier with the output of a category predictor. A category is an abstraction from the state space that describes properties of the environment. E.g., the abstract notion of one object being 'ontop' of another could be captured by a category predictor. Further examples are one object 'inside another' or an object being 'in reach' of the robots arm. Such abstraction are useful for general reasoning in a cognitive system. In this paper we show, how such categories can be built autimatically from experience made by exploration.

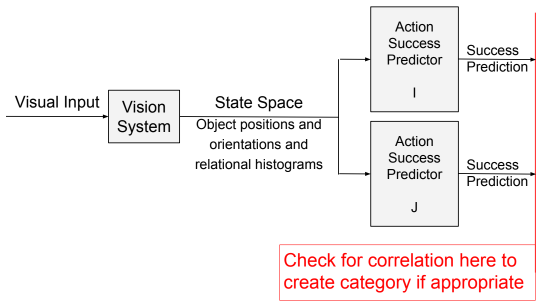
We investigate two different variations of category prediction based bootstrapping:

C1) adding the output of a single existing category predictor,
Cn) adding the output of all existing category predictorss

as additional inputs to the action success classifier under training.

If a category encoded by the added category predictor(s) is relevant for the new action, then adding the output of this category predictor to the new actions' state space can be used by the new predictor to bootstrap its learning.

To create a category predictor, we combine the training data that was used to train two individual action success predictors. This combined training set is then used to train a new classifier that captures properties of the state space, that lead to either success or failure for both of the two different actions. In this initial approach we limit ourselves to creating categories from



Visual Input
Vision System
State Space
Object positions and orientations and relational histograms
Action Success Predictor I
Action Success Predictor J
Success Prediction
Success Prediction
Check for correlation here to create category if appropriate

Fig. 3. Searching for correlations between action success predictions.

two actions while the concept could be extended to combine more actions, e.g., by finding clusters of correlating actions (i.e., actions that are often predicted to succeed in the same states).

B. Heuristic For Category Formation

The number of categories predictors potentially created in our approach increases polynomially with the number of actions available; therefore a heuristic for pruning out category candidates before creating and trying category predictors would be useful. The aim of a heuristic would be to remove the candidates that are unlikely to capture meaningful properties of the agent's environment.

We hypothesise that strong correlations between action predictions indicate that a meaningful category predictor can be learnt. Hence, as heuristic we use the correlation between pairs of action success predictor outputs. After learning at least two action success predictors, the system records the prediction correlation between pairs of action success predictors. To compute the correlation between these, both action success predictors are presented with the same scenes to predict their associated actions' success. The agent then records the predictions of each of the action success predictors for all scenes and calculates the correlation between these action success predictions (see Fig. 3). A category predictor is created only if the correlation exceeds a certain threshold.

C. Bootstrap Factor

To analyse the effect of our approaches to bootstrapping, we calculate a bootstrap factor as

$$bf = \frac{P_{bl} - P_c}{P_{sl} - P_c}$$

where P_{bl} is the performance of bootstrapped learning (e.g., Fig. 8), P_{sl} is the performance of standard learning (e.g., Fig. 7), P_c is the performance of chance and bf is the bootstrap factor as illustrated in Fig. 9.

This gives a value above one if bootstrapping assists learning and a value between zero and one if bootstrapping impedes learning. Note that a small bootstrap factor can stand for significant improvements in learning speed. E.g., jumps from 70% to 80% performance, or from 76.7% to 90% performance have a bootstrap factor of 1.5.

IV. Experimental Setup

The robot experiments used in this work were performed using a physically realistic simulated robotics setup (Section IV-A). Correspondingly a perception system in combination with a simulated Kinect is used (Section IV-B). The objects and actions used in the experiments are described in IV-C and Sections IV-D respectively. In the last section (IV-E) the used classification technology (Random Forests) is described in some detail and used parameters are given.

A. Simulation Environment

To collect data we used a physically realistic simulation environment [40]. This simulator is designed for robot simulations and simulates a Kinect sensor [41] with a realistic noise model [42] to produce data that resembles the depth map results from real Kinect camera setups. We simulated a robot arm with six degrees of freedom (DOF), mounted on a table with a two finger gripper attached as tool (see Fig. 1).

B. The Robotic Perception System

The Kinect camera is positioned in the workspace on the opposite side of the robot at a high position, looking down on the workspace area in front of the robot.

The Kinect records images with VGA resolution (640x480 pixels) and provides one 3D point per pixel (307200 points per scene). For performance reasons, we subsampled this point cloud via voxel grid subsampling with a resolution of 0.0125 m which resulted in point cloud sizes between 40k and 50k points.

We use simple colour based segmentation in this work. All objects are coloured in one of a set of known colours, and every object in the scene has a different colour selected from this set. All points that have the same colour are assigned to a new point cloud, representing one object. This is a common simplification also used in real vision set ups, e.g., by Rosman and Ramamoorthy [27][1]. We acknowledge that highly sophisticated object segmentation algorithms exist, e.g., [43], [44] and we assume they could be employed to work in a more complex environment, e.g. with real objects and clutter.

Using Eigen decomposition (as used for PCA) over the segmented point clouds, our vision system extracts nine variables as approximations of an object's position, orientation and size. The nine variables that describe the segmented object point clouds are:

- X, Y and Z position of the centre of the segmented object point cloud (the Euclidean average of all point positions in the robots coordinate frame).
- Three angles (Roll, Pitch and Yaw) describing the orientation of the segmented object point cloud. Based on the orthogonal set of Eigenvectors of the point cloud, we derive a segmented object point cloud coordinate system (x-axis is in the direction of the biggest Eigenvector, z-axis is in the direction of the smallest Eigenvector). The three angles describe the relative orientation between the segmented object point-cloud coordinate system and the robot coordinate system.
- Three size values for the elongation along the object's three axes (the Eigenvalues are used to indicate elongation along each Eigenvector axis).

This gives reasonable results for most objects in practice, finding sensible directions for not spherical or not symmetrical objects. For symmetrical objects such as spheres and cylinders, the resulting direction is partly arbitrary and a product of noise (i.e. due to noise in the camera sensor the point cloud would not be perfectly symmetrical or spherical), but the 'direction' of a sphere is irrelevant for manipulation and therefore this poses no issue. Likewise for a cylinder the important orientation of the cylinders' main axis is captured correctly. These nine variables per object make up the 18 variable baseline vision state space representation, where the first nine variables belong to the object that is subject to manipulation by the action.

Independently from the Eigen decomposition based object representation, the segmented point clouds are also used to create *relational histogram features* (RHF) [23] to capture the spatial relationships between objects. These RHFs form a relational space into which the absolute geometric information (3D position and orientation) of the segmented object point clouds is transferred. For this, every point of the first object is compared with each point of the second object to calculate a set of distance and angle features. These relational features encode the spatial relationship of the two objects and are captured in form of a histogram with 300 bins. This RHF extraction process is described in more detail in [25], [26] and we use the RHFs described there as 1D histograms as these were found to show the best performance in [25].

The final state space is then made up from the $9 + 9 = 18$ values of the PCA state space representation for the two objects on their own, or combined with the RHF to $18 + 300 = 318$ values. In the remainder of this paper, we will refer to these as the PCA or RHF state spaces respectively.

C. Objects

In our experiments, we used an overall set of 29 objects, which can be split into four different groups (see Fig. 4)[2]:

1.	Toys	(5 Objects)
2.	Bases	(15 Objects)
3.	Obstacles	(5 Objects)
4.	Rakes	(5 Objects)

For every experiment exactly one toy object and one object of a different group is used. The objects are randomly distributed within a workspace area that approximately forms a semi-circle with a radius of 1.8 m. The robot arm is placed in the center of the semi-circle and no object is placed closer than 20 cm away from the robot. The rake objects are an exception as they are attached to the robot arm as a tool. The maximum reach of the robot is approximately 1.2 m.

[1]The mentioned approach uses stereo but this is in the same way applicable to the Kinect sensor since the for Kinects necessary registration between point-clouds and colour is straightforward.

[2]One Object (Cup) is member of two Groups (Toys and Bases)

TABLE I
LIST OF ACTIONS

Action	Motor Program	Goal
Lift	Grasp base object and lift it.	Base and Toy objects are lifted.
Move	Reach to toy object and move it aside.	Toy & base objects have moved aside.
Pull	Grasp base object and pull it.	Base and Toy objects are pulled closer.
Push	Grasp base object and push it.	Base and Toy objects are pushed further away
Rake	Put rake head behind toy object and pull.	Toy object has been brought closer.
Take	Grasp toy object and lift it.	Toy object is lifted.
Pour	Grasp base object and lift & tilt it.	Base and Toy object are lifted.
Slide	Grasp base object and lift & tilt it.	Base object is lifted, Toy object has moved but not been lifted.
Unobstruct	Grasp base/obstacle object and push aside.	The toy object that wasn't reachable before, is now reachable.

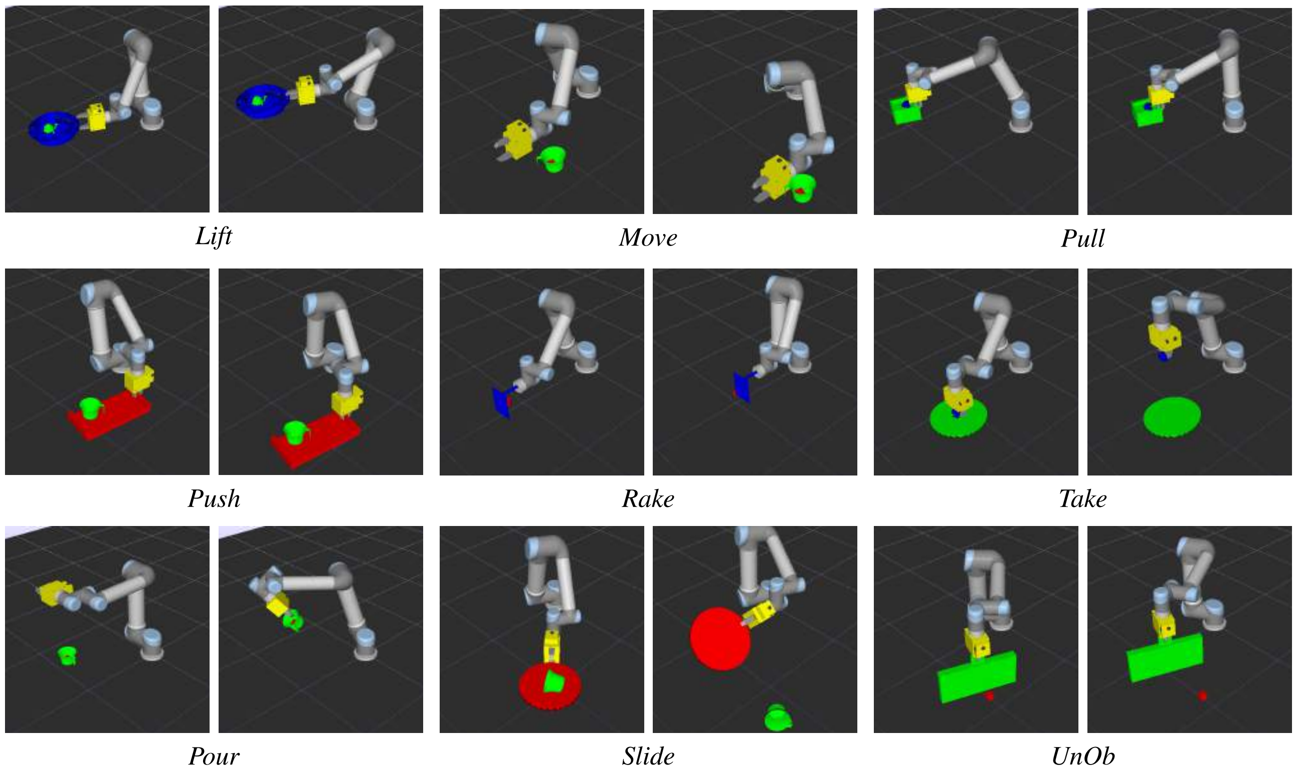


Fig. 5. Action execution effect snapshots of the nine actions. The left figures show a scene before action execution. The right figures show the state afterwards. (UnOb is short for Unobstruct.)

D. Actions

The robot is equipped with nine actions it can perform. Depending on the two objects in the workspace, one of which always is a toy object, different actions are available for execution. Table I briefly describes the actions, their motor programs, and the objects involved and the actions' goals. Fig. 5 illustrates the actions' motor programs.

Note that some actions have identical goals but different motor programs, while other actions have identical motor programs but different goals. The set of actions was contrived to cover an extensive set of both, different requirements for success and also similar/duplicated requirements. With this set of closely-, partly- and un-related preconditions between actions, we can demonstrate both successful and unsuccessful bootstrapping results.

The simulated actions used inverse kinematics without path planning, which leads to occational failures due to impossible paths. We collected approximately 10,000 samples per action, with 50% positive and 50% negative samples. The positive samples are selected uniformly. The negative samples are selected with a bias such that the distribution of distances between the centres of gravity (COG) of the two objects is similar within the groups of positive and negative samples. If we do not select negative examples with smaller distance then a classifier can achieve very high accuracy by simply using the distance between COGs, and not capturing the 'on top'/'inside' relation at all.

E. Classifiers

To predict the success of actions in a particular scene, we use ensemble classifiers based on the Random Forest [45] algorithm. In the following sections we will first describe

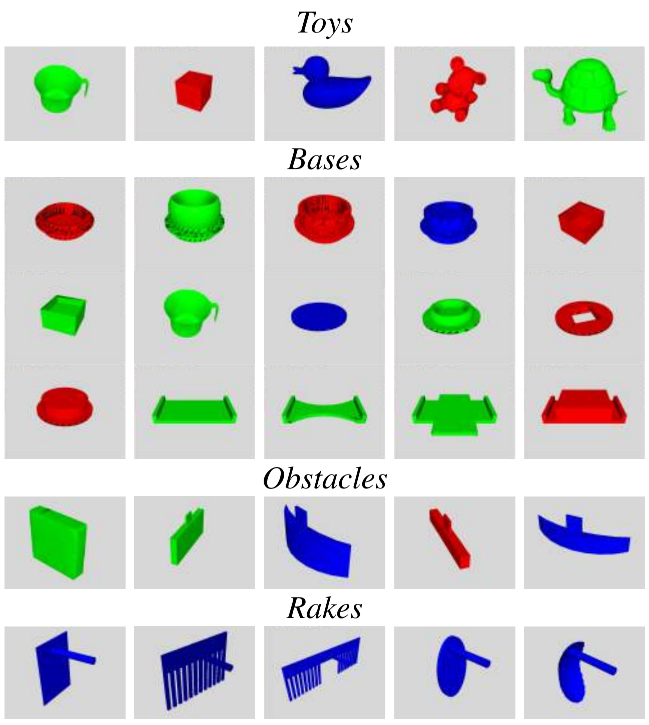


Fig. 4. Illustration of the Objects used for the experiments of this work

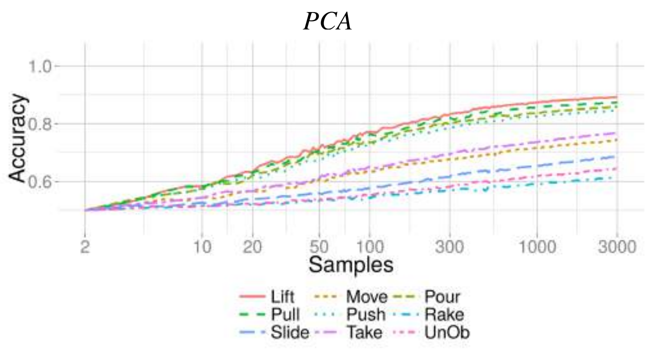


Fig. 6. Illustration of action success classifier learning for nine actions using the PCA state space without bootstrapping. The X-axis is logarithmic scaled to highlight that the learning continues even after several thousand training samples.

Random Forests in general and then give details about the parametrisation of the Random Forests used in this work.

1) Random Forest Classifiers: The idea of ensemble classification methods is to turn a set of 'weak' classifiers into one 'strong' classifier, where the final classification is based on some form of voting scheme. In Random Forests, these 'weak' classifiers are standard *Classification and Regression Trees* (CART) [46], which iteratively split the dataset into more and more subsets where each split maximises the class purity of the created subsets. The number of trees used can vary from ten up to hundreds or even thousands of trees per forest. The final prediction of the Random Forest is the class that is returned by most of the individual trees.

Random Forests have a series of advantageous properties compared to other types of classifiers, e.g. SVMs or ANNs, some of which they inherit from classification trees. Random Forests:

- do not require much data preparation: Linear dependent variables do not have to be removed and data normalisation is not required.
- can handle both, numerical and categorical data.
- are comparably fast.
- look for the best variable to split. They inherently do feature selection and are therefore particularly well suited for high dimensional data.
- tend to not overfit (if forest $\gtrsim$ 10 trees). Therefore no pruning is necessary leading to strong individual trees.

Random Forests (RF) are particularly well suited for our use case, as they inherently do feature selection and hence identify the relevant features from the (potentially) large amount of state space variables (see Section IV-B).

2) Random Forest Parameterisation: The amount of trees (T) in our Random Forests is dependent on the amount of samples available for training (N), where

$$T = 95 + \frac{N}{20}$$

up to a maximum size of 115 trees (the maximum forest size is reached with 400 samples). To parametrise the trees in our Random Forests, we followed the guidelines of Breiman et al. [45], [47]. For each tree in the forest, we use $2/3$ of the samples in the training set to train the tree. We use standard CART trees as base classifiers using the *Gini impurity criterion* [46] to split nodes and allow a minimum size of 1 sample for a partition created by a split. We do not prune the trees, but we do limit growth to a maximum depth of 200 splits. For every split, each tree uses all samples available to it ($2/3$ of the samples available to the forest). But, for every split, at each node, out of M dimensions, only $m << M$ are randomly picked and the best split on these is used to split the node, where $m = \sqrt{M}$.

V. Results

In the following subsections, we present the results of learning action success classifiers based on the different approaches introduced in Section III.

The results of basic learning (B) of action success classifiers without bootstrapping is presented in Subsec. V-A. The results of learning action success classifiers when bootstrapping with already learnt action success predictors following the approaches (A1), (A2) and (An) are shown in Subsecs. V-B, V-C and V-D respectively. In Subcec. V-E the results of bootstrapping from a single category (C1) are presented. The outcome of using the heuristic for category formation is shown in Subsec. V-F. Finally, the results of learning action success classifiers from all created categories (Cn) is presented in Subsec. V-G.

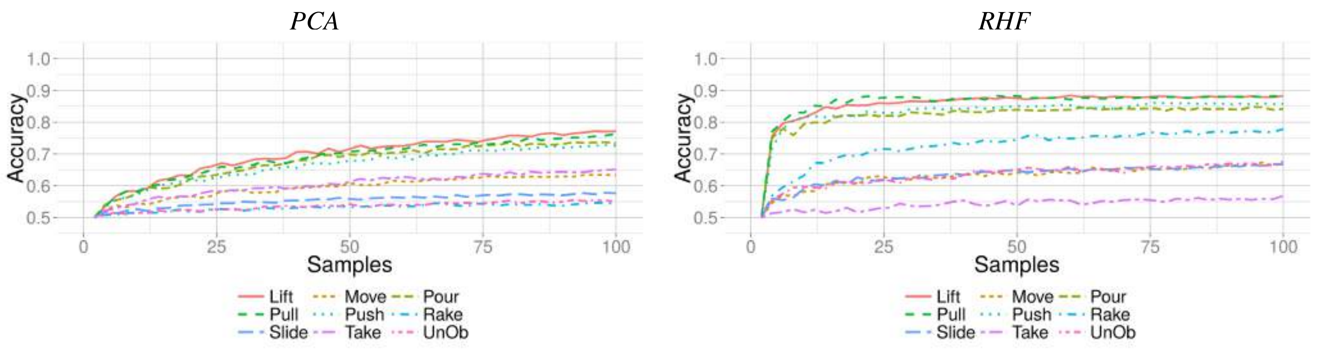


Fig. 7. Illustration of the early stage of action success classifier learning for nine actions in different state spaces without bootstrapping.

A. Basic Learning of Action Success Classifiers (B)

Following the basic approach (B) for learning of classifiers as described in Sec. III-A1, we trained a set of success classifiers for nine actions based on the PCA and RHF state spaces as described in Sec. IV-B.

The long term learning rate for the different action success classifiers using the PCA state space representation is illustrated in Fig. 6. It can be seen that learning continues after learning from several thousand training samples. The early stage of learning success classifiers for the same actions using different state space representations are highlighted in Fig. 7. It can be seen by comparing the PCA and RHF cases, that an appropriate state space representation is of significant importance for learning. The state representation using RHFs in general massively outperforms learning without histograms. This is not surprising as the RHFs were specifically designed with this use case in mind. We will show that we can achieve similar performance in the PCA state space via bootstrapping.

The 'Take' action serves as a good example of the potential shortcomings of such specialised state spaces. The success of 'Take' has a weak negative correlation with an aspect the RHF is able to represent (i.e., 'inside') but has a much stronger correlation with aspects not being represented by the RHF (e.g., object orientation). The increased amount of input variables in the RHF case therefore actually causes a decrease of learning speed (curse of dimensionality).

The basic action success classifier results presented in this section serve three purposes: Firstly, they illustrate the 'standard' learning rate for learning action success classifiers when learning without bootstrapping and will be used as a baseline for comparison in the following sections. Secondly, they constitute part of the 'knowledge' that will be used for bootstrapping in the following sections. Thirdly, the RHF based results serve as performance 'benchmark'. Using bootstrapping in the PCA state space we attempt to achieve similar learning speeds and accuracy as can be achieved using manually optimised state spaces.

B. Bootstrapping With a Single Already Learnt Action Success Predictor as Knowledge Source (A1)

Here we present the results of using a single already learnt action success predictor as knowledge source, as described in Section III-A2. In practice, this means that the prediction of an existing action success predictor is used to extend the state space of the new action success classifier, by adding it's prediction to the 18 PCA or 318 RHF state space variables. These additional inputs in the state space might make the mapping from state space to action success easier to learn, if the added inputs carry relevant information.

We found that bootstrapping works best if the bootstrapping input gets promoted by being added more than once, because by that the likelihood of being chosen in the Random Forest algorithm increases. For that reason we added the prediction of the already existing action success predictor six times to the PCA state space and 30 times to the RHF state space. This creates an extended PCA state space of 24 variables or an extended RHF state space of 348 variables.

Fig. 8 demonstrates the bootstrapping on all nine actions. For each of the nine actions, the 'best' bootstrapping result is presented, highlighting the potential of bootstrapping. This 'best' result is the one with the highest average accuracy learning curve from two to 100 samples. Comparing the PCA cases from Fig. 8 and Fig. 7, the increase in learning speed achievable through bootstrapping becomes evident. Fig. 8 also demonstrates that bootstrapping in simple state spaces *(PCA)* can lead to performances comparable to learning in optimised RHF state spaces as in Fig. 7.

Fig. 9 emphasizes the bootstrapping effect of the results demonstrated in Fig. 8, by presenting the corresponding bootstrap factors (as defined in section III-C). Two observations can be made here: **a)** It can be seen that especially when learning from few samples, the bootstrap factors are considerably large with values between three and eight in the PCA case. In the RHF case, however, an approximate bootstrap factor of one indicates that there is no or only little bootstrapping effect. The bootstrap factors then decrease until they asymptotically approach a value of 1.0 as the performance of the unbootstrapped learner catches up with the bootstrapped learner. **b)** Fig. 9 also highlights the fact that bootstrapping in simple state

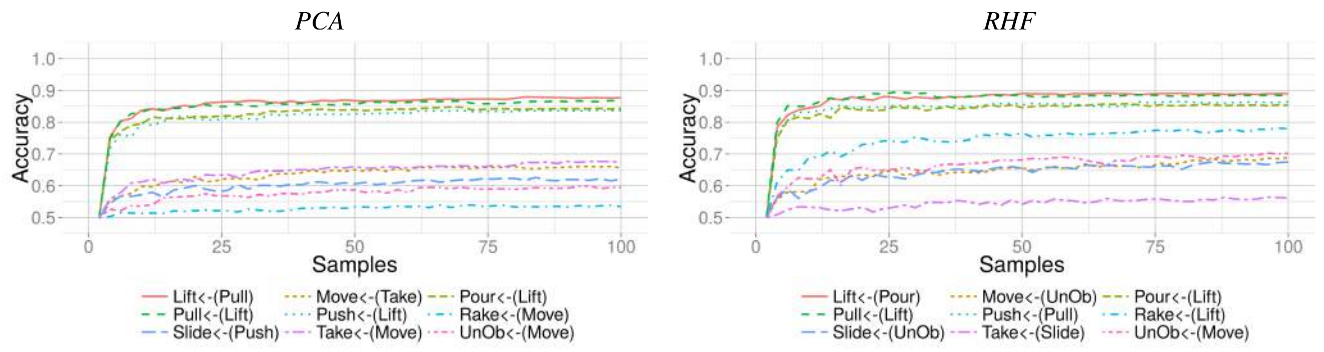


Fig. 8. Illustration of the action success classifier learning speed for nine actions in different state spaces using another action success predictor output for bootstrapping. Of all the potential action success prediction sources for bootstrapping, for each action, only the best result is presented. It can be seen that with bootstrapping in the simple PCA state space, similar performance can be reached as when learning in manually optimised state spaces.

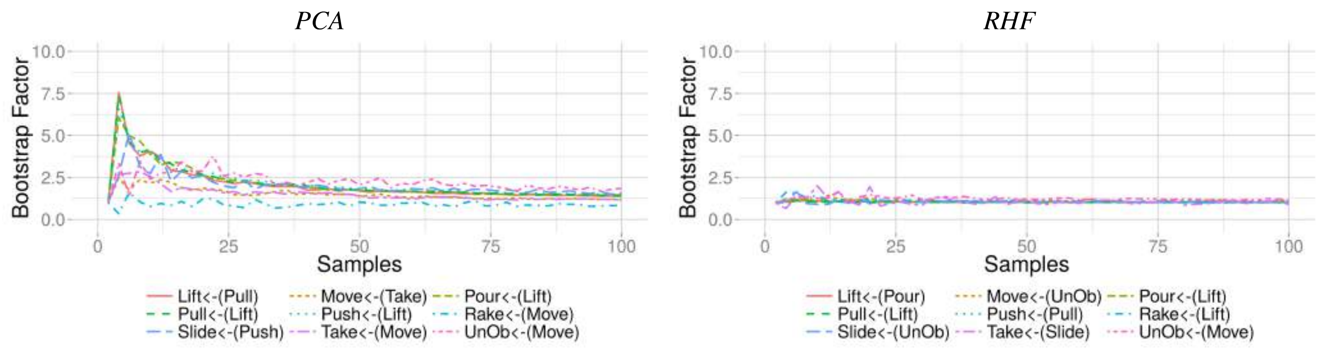


Fig. 9. Illustration of the bootstrap factors of nine action success classifiers in different state spaces when learning with another action success predictor output for bootstrapping. Of all the potential action prediction sources for bootstrapping, for each action, only the best result is presented.

spaces like the PCA case is multiple times more effective than in specialised state spaces where learning is already efficient without bootstrapping like in the RHF case.

It is evident that, in order to achieve these bootstrapping effects, the single best candidate from a group of available action success predictors has to be selected for bootstrapping. This is also the case when using a pair of action predictions (A2) or single categories (C1) for bootstrapping. How this selection can be achieved in practice is described in V-H.

As we demonstrated here that bootstrapping in PCA state spaces can lead to similar performance as manually optimised state spaces like the RHF state space and that there is no significant bootstrapping achievable in the RHF state space, we will, in the following, only present the bootstrapping results of the PCA state space.

C. Bootstrapping With Action Success Predictor Pairs As Knowledge Source (A2)

Here we present the results of learning an action success classifier when adding the output of two action success predictors of other actions as knowledge source, as described in Section III-A2. As before we added the inputs multiple times, but did not increase the overall amount of additions, i.e., the six inputs in the PCA state space where filled half by each of the actions of the action pair used for bootstrapping.

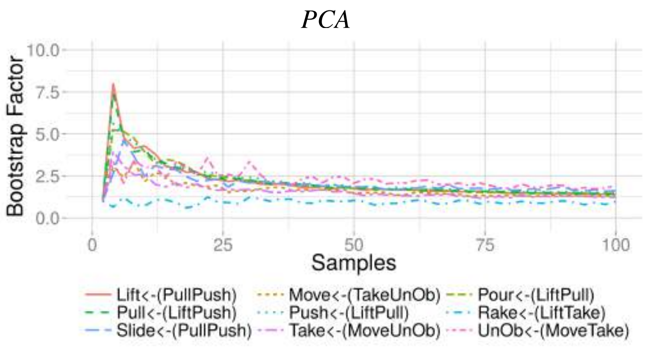


Fig. 10. Illustration of the bootstrap factors of nine action success classifiers in the PCA state space when learning with action pairs for bootstrapping. Of all the potential knowledge sources for bootstrapping, for each action, only the best result is presented.

Fig. 10 emphasises the effect of bootstrapping on the learning by presenting the corresponding bootstrap factors. Similar as before, the bootstrapping effect is largest at the early stage of learning. The result is similar as before, however, the best two candidates had to be picked from eight candidates, which allows 28 possible combinations. This adds complexity without adding benefit.

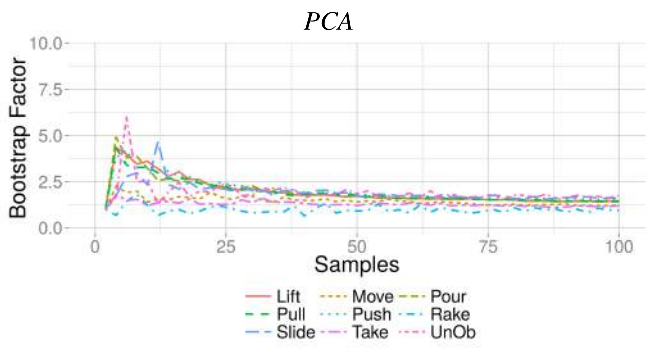


Fig. 11. Illustration of the bootstrap factors for nine action success classifiers in the PCA state space when learning with all existing action predictions for bootstrapping.

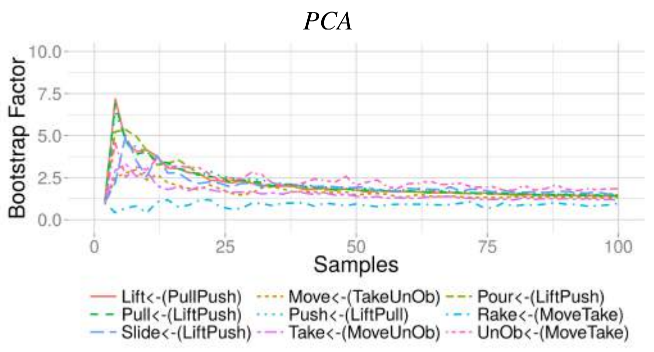


Fig. 12. Illustration of the bootstrap factors of nine action success classifiers in different state spaces when learning with the output of a single category predictor for bootstrapping. Of all the potential knowledge sources for bootstrapping, for each action, only the best result is presented.

D. Bootstrapping With All Action Success Predictors As Knowledge Source At Once (An)

Here we present the results when using the action success predictions of all eight other actions as knowledge source, as described in Section III-A2. As before we added the inputs multiple times, however, as there where more than six inputs to add, we added each input exactly once.

The bootstrapping effect when using the success predictions of all other available actions for bootstrapping is lower than with the single best other action success prediction. This effect can be seen when comparing Fig. 11 with the PCA part of Fig. 9. This is because not every action success prediction does help bootstrapping every other action success classifier and instead obstructs bootstrapping by both adding noise to the state space and thus making learning difficult and also increasing the size of the state space, also making learning more difficult (the curse of dimensionality [48]). This approach, however, is the most straightforward one, as it relieves the agent from searching for the best bootstrapping input from a set of candidate action success predictions and also does not require the learning of category predictors.

E. Bootstrapping With a Single Automatically Created Category Predictor (C1)

Here we present the results when using the output of a single automatically created category predictor as knowledge source, as described in Section III-A3. As before we added the inputs six times to the PCA state space.

Following the binomial combinatorics rule

$$\binom{9}{2} = 36$$

from our nine actions we create 36 category predictors.

Again, Fig. 12 emphasises the effect of bootstrapping on the learning by presenting the corresponding bootstrap factors. Similar as before, the bootstrapping effect is largest at the early stage of learning. It can be seen by comparing Fig. 12 with Fig. 9 that using category predictions for bootstrapping is comparable to bootstrapping from single action success predictions. The same holds true when comparing Fig. 12 with

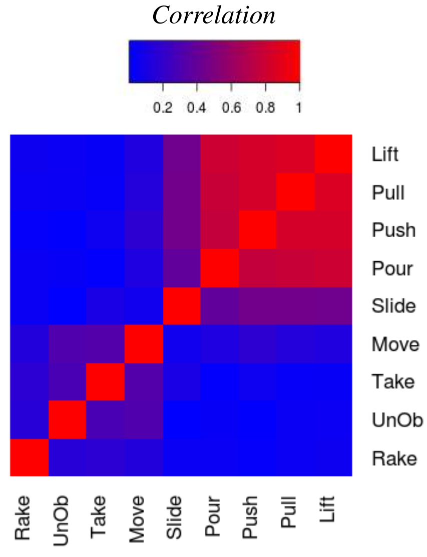


Fig. 13. Correlations between the success predictions for nine different actions after training with 2000 samples.

Fig. 10 where the pair of action success predictions used for bootstrapping stem from the same two actions that are used for creating the categories used for bootstrapping here. This slight improvement is likely due to the more abstract representation of concepts of the environment that can be directly accessed when bootstrapping from category predictors.

F. Effects Of Using the Heuristic For Creating Category Predictors

As a heuristic to avoid the combinatorial increase of the number of categories created by our approach, we limit the creation of categories to those that are based on strongly correlating pairs of action success predictors. This is based on the hypothesis, that strong correlations between action predictions indicate that a meaningful category predictor can be learnt. The correlations between the nine action success predictors after learning each from 2000 samples are shown in Fig. 13.

Two clusters can be noted in Fig 13. The first cluster contains primarily 'Lift', 'Pull' and 'Push'. These three actions have in common that they work fairly reliably when the toy object is 'ontop' or 'inside' of the base object, while they never work otherwise (i.e., in the when the objects are only beside each other on the table). The actions 'Pour' and 'Slide' are related in the sense that they share these properties with 'Lift', 'Pull' and 'Push', and differ only slightly with 'Pour' being unseccessful in the 'ontop' case and 'Slide' being unseccessful in the 'inside' case. The Second cluster is weaker and separated from the first cluster as the actions 'Move', 'Take', 'UnOb' and 'Rake' are more likely to be succesfull case of the two objects being 'beside' each other on the table than if they are 'ontop' or 'inside' each other. At the same time, no two actions in the second cluster have identical cases where they work and where not (unlike 'Lift', 'Pull' and 'Push' in the first cluster), hence there is a generally weaker correlations between them.

The heuristic we use has, as described in Sec. III-B, one parameter that can be changed. This parameter is the correlation threshold specifying the minimum correlation between two action success predictors that is required to create a category predictor based on these two actions. A high threshold will lead to few category predictors, but the resulting category predictors are more likely to be useful for bootstrapping. A low threshold instead, will lead to more category predictors, but not all of them might be useful for bootstrapping a new action success classifier.

To illustrate the effects of using the heuristic for creating category predictors, we present the best bootstrapping results of three sets of category predictors that are created using different thresholds for the correlations between action success predictors. For this, we chose as correlation thresholds T the three values of:
1) $T = 0.0$ (i.e., the heuristic is not used)
2) $T = 0.125$ (approx. $1/2$ of the category predictors created)
3) $T = 0.5$ (approx. $1/4$ of the category predictors created)

For each of these three sets we present the average bootstrap factors of bootstrapping all nine action success classifiers with the best fit of the particular set of category predictors. We call these three results 'Best of $C > T$', where C is the correlation between to action success predictors and T is the correlation threshold 1) 0.0, 2) 0.125 or 3) 0.5 respectively. Additionally, we present as expected performance of selecting a random knowledge source the overall average of bootstrapping every action success classifier with every possible category predictor as input. Comparing this with the results of bootstrapping with the best of $C < T$ highlights the difference between bootstrapping with the best and random knowledge source. All four results are shown in Fig. 14.

It can be seen in Fig. 14, that the drop of achievable performance through an increased thresholds is small. The average achievable performance of the full set of category predictors with threshold $T = 0.0$ and the $1/2$ set of category predictors with threshold $T = 0.125$ are almost identical. Even when using the threshold $T = 0.5$, where only $1/4$ of the category predictors are created, the achievable bootstrap factor does only drop slightly. Not surprisingly, the bootstrapping factor of training each action success classifier with the best knowledge source from either of the three sets is significantly better than the bootstrapping overall average when using an arbitrary pick of avalable action predictor as knowledge source.

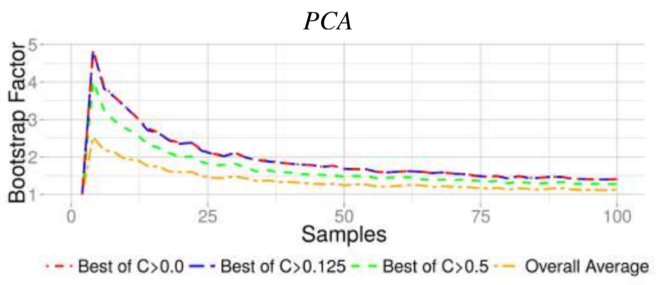


Fig. 14. Illustration of the average bootstrap factor of different category predictor sets over the course of learning action success classifiers from two training samples to 100 training samples. The plot shows the average bootstrap factor of all action success classifiers when bootstrapped with their individual best knowledge source category from the set of category predictors created according to the heuristic with a threshold of 1) 0.0 (i.e., no heuristic used), 2) 0.125 (only approximately $1/2$ of the possible category predictors are created) and 3) 0.5 (only approximately $1/4$ of the possible category predictors are created) The plot also shows the overall average bootstrap factor of all possible action success classifier / category predictor combinations.

We can conclude that it is in general good to neglect a significant proportion of category predictors. This is because even a small number of category predictors based on highly correlated actions will give good bootstrapping effects for most action success classifiers.

G. Bootstrapping With All Available Category Predictor outputs As Knowledge Source at Once (Cn)

Here we present the results when using all category predictors created according to the heuristic using the higher correlation threshold of 0.5 (see Fig. 15). This aproach of using all available category predictors for bootstrapping at once has been described in Section III-A3. As before, we added the inputs multiple times, however, as there where more than six inputs to add, we added each input exactly once.

When bootstrapping from all available action success predictions (An), we noticed that the learning and bootstrapping performance was not as high as when bootstrapping from the single best action success prediction (A1). We argued that this was due to the curse of dimensionality. However, when bootstrapping from all available category predictions (Cn), the learning and bootstrapping performance remains high and sometimes surpasses the results of learning with the single best action success prediction or single best category prediction input for bootstrapping (apart from the 'Take' action that does degrade in performance). We argue that this is due to the easier accessible knowledge captured by category predictors, compared to the knowledge made available by action success predictors. This effect can be seen when comparing Fig. 15 with the PCA part of Fig. 12.

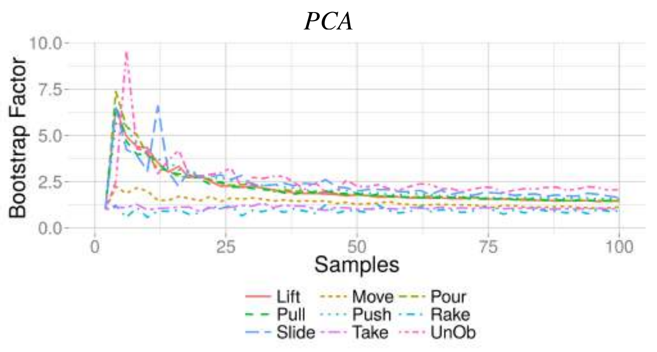


Fig. 15. Illustration of the bootstrap factors of nine action success classifiers in the PCA state space when learning with all existing category predictor outputs as inputs for bootstrapping.

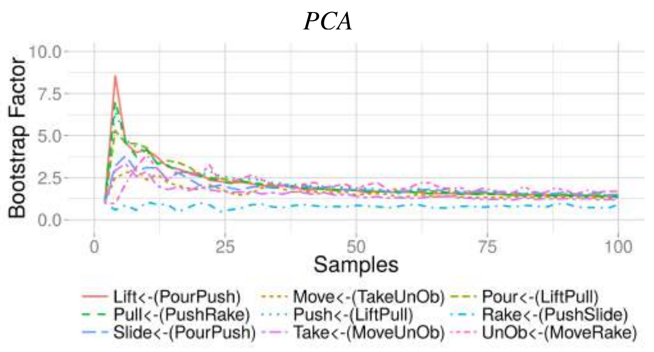


Fig. 16. Illustration of the bootstrap factors of nine action success classifiers in different state spaces when learning with the output of a single category predictor for bootstrapping that was leading to the highest classifier accuracy after ten training samples.

H. Candidate Selection For Bootstrapping

In the previous subsections we presented the results for for bootstrapping each action with the input that is 'best' for bootstrapping each action success classifier. This 'best' input was selected for presentation after all alternatives have been tested and compared by measuring the average accuracy of the action success classifiers at each step from 2 to 100 samples. This is not a valid approach in an online system and instead the input has to be selected automatically and rapidly. Our suggested approach to automatically selecting an appropriate input for bootstrapping is based on the premise that the suitability of a candidate input for bootstrapping becomes clear almost instantaneously. If this was not the case, bootstrapping of learning a new action success classifier and reaching high accuracy after few training samples (i.e., bootstrapping as presented in this paper) would not be possible.

Therefore, we suggest, to start by learning multiple action success classifiers in parallel – one for each knowledge source candidate. After few training samples, e.g., four to six training samples, the agent could begin to neglect the weaker performing bootstrapping inputs. A final decision can then be made after between eight to 16 training samples, to select the best current input for bootstrapping. With this approach the selected input is not necessarily the actual best. however, in our experiments the selection made by this approach is always either the best choice or very close with $< 5\%$ relative performance difference. Fig. 16 shows the bootstrap factors for nine action success classifiers when bootstrapped with the category predictor output that was leading to the best action success classifier accuracy when learning from 10 training samples. Comparing Fig. 16 with Fig. 12 shows that the bootstrapping factors of this automatic approach to selecting are similar to when using the best input as done in sections III-A2 and III-A3. It can be noted from the labels which inputs changed, and also, that the inputs that changed often belong to the same cluster of correlations as the actual best. E.g., For the 'Lift' action the best category for bootstrapping (see Fig. 12) is based on 'Pull' and 'Push' while the one selected here after learning from ten samples is based on 'Pour' and 'Push'. All four actions 'Lift', 'Pull', 'Push' and 'Pour' are in the same cluster of correlating actions in Fig. 13.

VI. Discussion

In this paper we demonstrated the learning of action success classifiers for means-end actions and investigated how this learning process can be accelerated via bootstrapping. We have presented a method to capture meaningful categorical features of the environment like one object being 'ontop' or 'inside' another. We have showcased a heuristic to limit this extraction of categories to the ones most likely to be useful to the robot for bootstrapping learning action success predictors. To quantify the bootstrapping performance we have introduced a bootstrap factor. And finally, we have presented results for bootstrapping the learning of success classifiers for nine means-end actions using a variation of existing action success predictors and category predictors as knowledge sources for bootstrapping. To the best of our knowledge, this is the first study that demonstrates bootstrapping in both successful and unsuccessful cases, highlighting the conditions that are necessary to exploit this approach to accelerating the learning of action success classifiers, instead of showcasing a single selected succesful scenario for bootstrapping.

We found that success predictors learnt for certain actions implicitly capture a *category*, for example the category of being 'on top', or 'not on top', or 'inside'. We avoid calling these *concepts* because a concept suggests a complex package of information, e.g. knowledge of situations or associated actions (see Barsalou [49], [50]), whereas we are talking about something more restricted: a classifier determining the presence of a critical aspect of a scene, e.g. spatial relationship category. These implicit categories can be recognised by category predictors and treated as explicit symbols to be used as input to the learning of subsequent action success classifiers.

We demonstrated that using bootstrapping can significantly speed up learning of new action success classifiers. This is important, because there is a great deal to be learnt in order to achieve a basic level of manipulation competence in everyday environments, such as a child has. Spatial relationships are only one small segment of the common sense knowledge that is required (see [8]). Also, it typically takes a large amount of data to ground knowledge in a robot's own actions, and this data is usually hard to generate (i.e. through time consuming real world experiments). Techniques that can reduce the requirement for data are therefore beneficial. In the best cases demonstrated in this paper, learning from ten samples with bootstrapping can achieve similar accuracy as learning from approximately 1000 samples without bootstrapping (see left part of Fig. 8 and 6). At early stages of learning (i.e., with few training samples), bootstrap factors above 8.0 can be reached (see Fig. 12) and bootstrap factors near 5.0 as the average of each actions best bootstrapping result (see Fig. 14). With this, bootstrapping with rather simple state spaces can achieve similar results to learning with manually optimised state spaces.

With our category predictors, we have presented a simple form of extracting more symbolic descriptions of the environments. The next logical step would be to use these categories within a larger cognitive architecture, where this symbolic knowledge is not only used for bootstrapping of new actions, but also to guide future interactions and for the development of higher cognitive competences. Categories form a first step on the path to high level concepts and a robot which has recently acquired an 'on top' category symbol, for example, may be motivated to generate further experience around this 'on top' category, which could lead to a new phase of development that facilitates to the formation of higher level concepts like 'tray' or 'carrying'.

References

[1] A. Stoytchev, "Some Basic Principles of Developmental Robotics," *IEEE Trans. on Auton. Ment. Dev.*, vol. 1, no. 2, pp. 122–130, aug 2009.

[2] R. Moore, "Spoken Language Processing: Where Do We Go from Here?" in *Your Virtual Butler*, ser. Lecture Notes in Computer Science, R. Trappl, Ed. Springer Berlin Heidelberg, 2013, vol. 7407, pp. 119–133.

[3] M. Do, J. Schill, J. Ernesti, and T. Asfour, "Learn to wipe: A case study of structural bootstrapping from sensorimotor experience," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, may 2014, pp. 1858–1864.

[4] E. Ugur, S. Szedmak, and J. Piater, "Bootstrapping paired-object affordance learning with learned single-affordance features," in *The Fourth Joint IEEE Intl. Conf. on Development and Learning and on Epigenetic Robotics (ICDL-Epirob), Genoa, Italy*, 2014, pp. 468–473.

[5] F. Wörgötter, C. Geib, M. Tamosiunaite, E. Aksoy, J. Piater, H. Xiong, A. Ude, B. Nemec, D. Kraft, N. Krüger, M. Wächter, and T. Asfour, "Structural bootstrapping - A novel, generative mechanism for the faster and more efficient acquisition of action-knowledge," *IEEE Transactions on Autonomous Mental Development (accepted)*, vol. PP, no. 99, 2015.

[6] J. Piaget, *The Origins of Intelligence in Children*. London: Routledge & Kegan Paul, 1936, (French version 1936, translation 1952).

[7] P. Willatts, "Pulling a support to retrieve a distant object," *Developmental Psychology*, vol. 35, no. 3, pp. 651–667, 1999.

[8] F. Guerin, N. Krüger, and D. Kraft, "A Survey of the Ontogeny of Tool Use : from Sensorimotor Experience to Planning," pp. 1–26, 2012.

[9] G. Drescher, *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, 1991.

[10] H. Chaput, "The Constructivist Learning Architecture: A Model of Cognitive Development for Robust Autonomous Robots," *PhD Thesis*, no. The University of Texas at Austin, Artificial Intelligence Laboratory, 2004.

[11] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 1345–1359, oct 2010.

[12] J. J. Lockman, "A perception-action perspective on tool use development," *Child Development*, vol. 71, no. 1, pp. 137–144, 2000.

[13] P. Willatts, "Development of problem-solving strategies in infancy," in *Children's Strategies: Contemporary Views of Cognitive Development*, D. Bjorklund, Ed. Lawrence Erlbaum, 1990, pp. 23–66.

[14] J. Piaget, *The Construction of Reality in the Child*. London: Routledge & Kegan Paul, 1937, (French version 1937, translation 1955).

[15] J. M. Mandler, "How to Build a Baby: II. Conceptual Primitives," *Psychological Review*, vol. 99, no. 4, pp. 587–604, 1992.

[16] A. Clark and A. Karmiloff-Smith, "The cognizer's innards: A psychological and philosophical perspective on the development of thought," *Mind & Language*, vol. 8, no. 4, pp. 487–519, 1993.

[17] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *J. Artif. Int. Res.*, vol. 29, no. 1, pp. 309–352, Jul. 2007.

[18] T. Zimmerman and S. Kambhampati, "Learning-assisted automated planning: Looking back, taking stock, going forward," *AI MAGAZINE*, vol. 24, p. 2003, 2003.

[19] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Symbol acquisition for probabilistic high-level planning," in *Proceedings of the Twenty Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[20] J. Mugan and B. Kuipers, "Autonomous learning of high-level states and actions in continuous environments," *IEEE Trans. Autonomous Mental Development*, vol. 4, no. 1, pp. 70–86, 2012.

[21] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *IEEE Intl. Conf. on Robotics and Automation*, 2012, pp. 4373–4378.

[22] S. Griffith, J. Sinapov, V. Sukhoy, and A. Stoytchev, "A behavior-grounded approach to forming object categories: Separating containers from noncontainers," *Autonomous Mental Development, IEEE Transactions on*, vol. 4, no. 1, pp. 54–69, March 2012.

[23] W. Mustafa, N. Pugeault, and N. Krüger, "Multi-View Object Recognition using View-Point Invariant Shape Relations and Appearance Information," in *IEEE International Conference on Robotics and Automation*, 2013.

[24] S. Fichtl, J. Alexander, F. Guerin, W. Mustafa, D. Kraft, and N. Kruger, "Learning spatial relations between objects from 3D scenes," in *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, aug 2013, pp. 1–2.

[25] S. Fichtl, A. McManus, W. Mustafa, D. Kraft, N. Kruger, and F. Guerin, "Learning spatial relationships from 3D vision using histograms," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, may 2014, pp. 501–508.

[26] S. Fichtl, D. Kraft, N. Krüger, and F. Guerin, "Using Relational Histogram Features and Action Labelled Data to Learn Preconditions for Means-End Actions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (Workshop on Sensorimotor Contingencies for Robotics)*, Hamburg, 2015.

[27] B. Rosman and S. Ramamoorthy, "Learning spatial relationships between objects," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1328–1342, sep 2011.

[28] S. Panda, A. H. A. Hafez, and C. V. Jawahar, "Learning support order for manipulation in clutter," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, nov 2013, pp. 809–815.

[29] S. Fichtl, J. Alexander, D. Kraft, J. Jorgensen, N. Krüger, and F. Guerin, "Learning object relationships which determine the outcome of actions," *Paladyn*, vol. 3, no. 4, pp. 188–199, 2012.

[30] H. Grabner, J. Gall, and L. Van Gool, "What makes a chair a chair?" in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 1529–1536.

[31] E. Aksoy, A. Abramov, F. Worgotter, and B. Dellen, "Categorizing object-action relations from semantic scene graphs," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 398–405.

[32] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 951–970, 2013.

[33] A. Lazaric, "Transfer in reinforcement learning: A framework and a survey," in *Reinforcement Learning: State of the Art*, ser. Adaptation,

Learning, and Optimization, M. Wiering and M. van Otterlo, Eds. Springer Berlin Heidelberg, 2012, pp. 143–173.
[34] M. H. Lee, Q. Meng, and F. Chao, "Staged competence learning in developmental robotics," *Adaptive Behavior*, vol. 15, no. 3, pp. 241–255, 2007.
[35] S. Hart and R. Grupen, "Learning generalizable control programs," *IEEE Trans. Autonomous Mental Development*, vol. 3, no. 3, pp. 216–231, sept. 2011.
[36] P.-Y. Oudeyer, F. Kaplan, and V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 265–286, 2007.
[37] P.-Y. Oudeyer, A. Baranes, and F. Kaplan, "Intrinsically motivated learning of real-world sensorimotor skills with developmental constraints," in *Intrinsically Motivated Learning in Natural and Artificial Systems*, G. Baldassarre and M. Mirolli, Eds. Springer Berlin Heidelberg, 2013, pp. 303–365.
[38] E. Ugur and J. Piater, "Emergent structuring of interdependent affordance learning tasks," in *Development and Learning and Epigenetic Robotics (ICDL-Epirob), 2014 Joint IEEE International Conferences on*, Oct 2014, pp. 489–494.
[39] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese, "Understanding indoor scenes using 3d geometric phrases," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
[40] J. A. Jørgensen, L.-P. Ellekilde, and H. G. Petersen, "RobWorkSim - an Open Simulator for Sensor based Grasping," in *ISR/ROBOTIK 2010 (41st International Symposium)*. VDE-Verlag, Jun. 2010.
[41] K. Khoshelham and S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.
[42] S. Olesen, S. Lyder, D. Kraft, N. Krüger, and J. Jessen, "Real-time extraction of surface patches with associated uncertainties by means of Kinect cameras," *Journal of Real-Time Image Processing*, vol. 10, no. 1, pp. 105–118, 2012.
[43] K. M. Varadarajan and M. Vincze, "Object part segmentation and classification in range images for grasping," in *Advanced Robotics (ICAR), 2011 15th International Conference on*, jun 2011, pp. 21–27.
[44] M. Schoeler, J. Papon, and F. Wörgötter, "Constrained Planar Cuts-Object Partitioning for Point Clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5207–5215.
[45] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[46] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.
[47] L. Breiman and A. Cutler, "Random Forests," *https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm*, 2004.
[48] R. E. Bellman, "Adaptive control processes: a guided tour," *Princeton University*, 1961.
[49] W. Yeh and L. W. Barsalou, "The Situated Nature of Concepts," *The American Journal of Psychology*, vol. 119, no. 3, pp. pp. 349–384, 2006.
[50] L. W. Barsalou, "Simulation, situated conceptualization, and prediction," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 364, no. 1521, pp. 1281–1289, 2009.

Severin Fichtl holds a M.Sc degree in Computer Science (Artificial Intelligence) from the University of Aberdeen, UK and has recently received his Ph.D. degree at the University of Aberdeen. He is currently working as research assistant at the University of Southern Denmark and is interested in General Artificial Intelligence and Developing Cognitive systems.

Dirk Kraft is an assistant Professor at the Maersk McKinney Moller Institute, University of Southern Denmark. He holds a diploma degree in computer science from the University of Karlsruhe, Germany and a Ph.D. degree from the University of Southern Denmark. His Research interests lie within cognitive systems, robotics and computer vision.

Norbert Krüger is a professor at the Mærsk McKinney Møller Institute, University of Southern Denmark. He holds a M.Sc. degree from the Ruhr-Universität Bochum, Germany and his Ph.D. degree from the University of Bielefeld, Germany. His research covers computer vision, cognitive systems and applied robotics.

Frank Guerin obtained his Ph.D. degree from Imperial College, London, in 2002. Since August 2003, he has been a Lecturer in Computing Science at the University of Aberdeen. He is interested in infant sensorimotor development, especially means-end behaviour and precursors to tool use.

Exploration in Structured Space of Robot Movements for Autonomous Augmentation of Action Knowledge

Denis Forte, Bojan Nemec and Aleš Ude
Humanoid and Cognitive Robotics Lab
Department of Automatics, Biocybernetics and Robotics
Jožef Stefan Institute, Ljubljana, Slovenia
Email: denis.forte@ijs.si, bojan.nemec@ijs.si, ales.ude@ijs.si

Abstract—**Imitation learning has been proposed as the basis for fast and efficient acquisition of new sensorimotor behaviors. Movement representations such as dynamic movement primitives were designed to enable the reproduction of the demonstrated behaviors and their modulation with respect to unexpected external perturbations. Various statistical methods were developed to generalize the acquired sensorimotor knowledge to new configurations of the robot's workspace. However, statistical methods can only be successful if enough training data are available. If this is not the case, usually the teacher must provide additional demonstrations to augment the database, thereby improving the performance of generalization. In this paper we propose an approach that enables robots to expand their knowledge database autonomously. Efficient exploration becomes possible by exploiting the structure of the search space defined by the previously acquired example movements. We show in real-world experiments that this way the robot can expand its database and improve the performance of generalization without the help of the teacher.**

***Keywords*—imitation learning, reinforcement learning, dynamic movement primitives, statistical generalization, training data, autonomous database expansion.**

I. Introduction

It has been suggested that imitation learning can provide means to limit the huge state-action space of high degree of freedom robots such as humanoids [1]. Through imitation a robot can gain the first approximation of the desired movement, which can later be improved by autonomous exploration. It has been demonstrated that this way the robot can acquire difficult to learn behaviors, e. g. the game of kendama, which consists of two parts, throwing a ball up on an attached string and catching it in a cup [2]. Based on these early works, reinforcement learning of robot movements has started to be seen as a viable approach to motion learning in robotics [3]–[7], even in the case of high degree freedom humanoid robots. Reinforcement learning algorithms such as PoWER (Policy learning by Weighing Exploration with the Returns) [8] and PI^2 (Policy Improvement with Path Integrals) [5] are able to deal with sensorimotor learning in high dimensional spaces. Reinforcement learning focuses on how to acquire optimal task performance in a given situation. Other researchers have studied how to generalize the available movement primitives to new situations [9]–[12] or represent multiple movements within a dynamical system that can represent multiple control policies [13], [14]. The main difference between both approaches is that in the first approach, the primitives are combined on-line so that the optimal movement for the current situation is generated, whereas the second approach generates a representation of all movements off-line using a global optimization approach.

The described approaches enable the robot to autonomously improve single movements using reinforcement learning and to generate new variants of the learned behaviors using statistical techniques. In this paper we focus on how the robot can autonomously acquire new example movements for its database, which is the key to improving the performance of the initially roughly learned behavior. Up to now generalization mainly relied on the availability of the sufficient amount of training data, which were provided by the demonstrator. In this paper we show that firstly, the robot can exploit the structure provided by the previously acquired example movements to accelerate its learning process and secondly, that by adding the newly acquired trajectories to the database of example movements we can increase the performance of statistical generalization. Note that successfully solving the first problem does not necessarily mean the accuracy of generalization will also be improved; statistical learning can only be successful if we can ensure that the newly acquired data is correlated with the existing pattern of motor primitives in the database. Thus we must ensure that the new trajectories identified by reinforcement learning are correlated with the ones that are already in the database.

First we obtain a few training movements that solves the given task in some specific situations. Then we apply statistical generalization to compute relatively good initial approximation of new situation inside learning space. As the next step the reinforcement learning is used to refine the approximation in few steps so the robot can accomplish the task correctly. Every learned movement is then stored in the training base, so that additional approximations of different situations can be estimated more accurately and that reinforcement learning can get faster results. When the learning space is fairly revealed, which means that enough training data is acquired, the statistical generalization itself offers movement that is good enough and the reinforcement learning is not needed any more.

II. Acquisition of Example Database and Generalization

We use kinesthetic guiding [15] to acquire the initial example movements. Lets assume that a set of S example trajectories $\mathbf{M}_i$, $i = 1, \ldots, S$, has been collected by kinesthetic guiding, and that all these trajectories result in a successful execution of the desired task in different (but related) situations. See Fig. 1 for an example where a human demonstrator guided the robot to pour water from a bottle into a glass located at different positions on the table and with different quantities of water in the bottle across different demonstrations. We call the vector defined by parameters describing the task a query point (in the above example, the query point consists of position of the glass on the table and volume of the water in the bottle). Let $\mathbf{q}_i \in \mathbb{R}^m$, $i = 1, \ldots, S$, be the query points associated with every demonstration, where m is the dimensionality of these parameters. The example trajectories $\mathbf{M}_i$ are represented as sequences of positions, velocities and accelerations $\{\mathbf{y}_{ij}, \dot{\mathbf{y}}_{ij}, \ddot{\mathbf{y}}_{ij} \in \mathbb{R}^{dof}\}$, measured at times t_{ij}, $j = 1, \ldots, n_i$, $t_{i1} = 0$, where n_i defines the number of data points on the example trajectory $\mathbf{M}_i$ and dof denotes the number of degrees of freedom on the trajectory. Both Cartesian space and joint space trajectories can be used to define the example data set. In experiments described in this paper we use joint space trajectories.

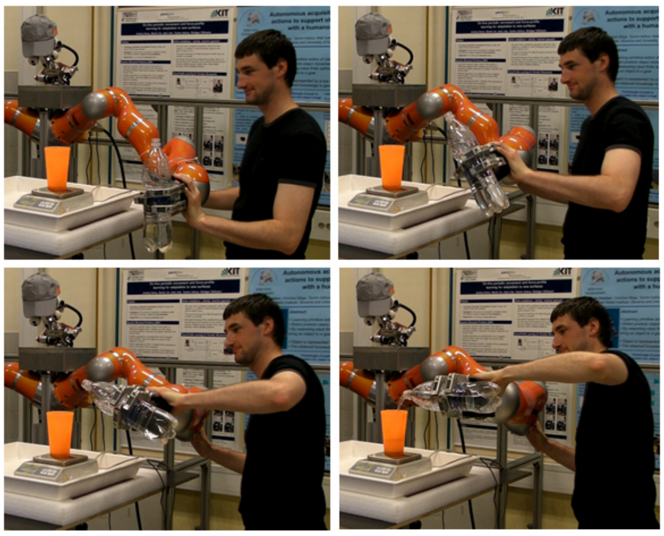


Fig. 1. Acqusition of example movements by kinesthetic guiding.

The aim of generalization is to compute a trajectory that solves the desired task at any given query point $\mathbf{q}$. In the above example, the output trajectory is the appropriate pouring movement. We implemented two different approaches to statistical generalization [9], [11]. In this paper we use the more efficient but less accurate approach described in [11]. This approach is based on dynamic movement primitives (DMPs) [16] and Gaussian process regression (GPR) [17]. For every query point $\mathbf{q}$, GPR computes an approximation of the task solution

$$\mathbf{G}_{\{\mathbf{M}_i, \mathbf{q}_i\}_{i=1}^S} : \mathbf{q} \mapsto \begin{bmatrix} \mathbf{w} \\ \tau \\ \mathbf{g} \end{bmatrix}, \tag{1}$$

where $\mathbf{w}$, τ, and $\mathbf{g}$ are the parameters defining a DMP that describes the movement for the given query point $\mathbf{q}$. For every degree of freedom, the DMP is defined by a second order dynamical system consisting of a linear and nonlinear part, where the linear part has a well-defined attractor dynamics, which ensures the convergence of the system, while the nonlinear part provides the ability to follow any desired trajectory on the given time (phase) interval

$$\begin{aligned} \tau \dot{z} &= \alpha_z(\beta_z(g - y) - z) + f(x), & (2)\\ \tau \dot{y} &= z, & (3)\\ \tau \dot{x} &= -\alpha_x x. & (4) \end{aligned}$$

Here y is the one of the robot's degrees of freedom, z is an auxiliary variable, and x is the phase variable common across all of the robot's degrees of freedom. α_x, α_z, and β_z are constants, which are specified so that the linear part of system (2) – (4) converges to the unique equilibrium point, which is at $z = 0$, $y = g$ and $x = 0$, regardless of where the movement has started. $\tau > 0$ is the temporal scaling factor that can be used to modulate the velocity of the movement. The nonlinear part f of the system contains free parameters w_k that enable the robot to track any desired trajectory from the initial position $(y_0, \dot{y}_0)$ to the final equilibrium point $(0, g, 0)$

$$f(x) = \frac{\sum_{k=1}^N w_k \Psi_k(x)}{\sum_{k=1}^N \Psi_k(x)} x, \;\; \Psi_k(x) = \exp\left(-h_k (x - c_k)^2\right). \tag{5}$$

Here c_k are the centers of radial basis functions distributed along the trajectory and $h_k > 0$. They can be computed as described in [9]. While α_x, α_z, β_z, τ, and x are the same for all of the degrees of freedom, z, y, g and the parameters w_k vary across the degrees of freedom. Note that since x converges to 0, $f(x)$ also converges to 0 and the system (2) – (4) becomes linear as time tends to infinity (and consequently phase x to zero). Given a new query point $\mathbf{q}$, the algorithm of Forte et al. [11] computes the DMP parameters for all degrees of freedom. Thus $\mathbf{w} \in \mathbb{R}^{N \times dof}$ contains the weights w_k associated with every degree of freedom and vector $\mathbf{g} \in \mathbb{R}^{dof}$ the final configuration of every degree of freedom. τ is associated with the phase variable and is therefore common across all of the degrees of freedom.

The details of the described generalization approach can be found in [11]. What is important for this paper is that for every query point $\mathbf{q}$, the generalization function (1) computes the corresponding DMP. Thus statistical generalization provides a low-dimensional parametrization of the solution space, which can be exploited to accelerate the acquisition of new example trajectories.

III. Expanding the Example Database by Autonomous Exploration

Although generalization can provide an estimate for the trajectory (with respect to the demonstrated ones), which can be used to accomplish the desired task, the computed movement can of course only be good enough for successful task accomplishment if enough training data are available. Additional training examples need to be provided if the task performance is not as good as expected. While a human teacher could provide more data, it would be advantageous if the robot could explore the solution space on its own and expand the database autonomously. In this section we show how to realize such autonomous exploration.

A straightforward approach to generate more data would be to start with the initial approximation for the desired movement as provided by statistical generalization at the given query point and continue with one of the standard reinforcement learning algorithms to find a better solution. There are several problems with this straightforward approach:

- General reinforcement learning is model-free and it might take a long time before a better solution can be found.
- The solution is often not unique, e. g. a given amount of water can be poured into a glass in many different ways, therefore general reinforcement learning might find a solution which is different from the solution selected by the demonstrator. Significantly different movements are not suitable for statistical generalization.

To address the first problem, we exploit the parametrization provided by the statistical generalization function (1). To address the second problem we need to bias the search algorithm towards the trajectory manifold defined by the generalization function.

Another issue to be considered when augmenting the database is that the performance of statistical generalization is usually better if query points are uniformly distributed throughout the desired query point space. In the proposed system, once the robot determines that the accuracy of generalization is insufficient, we systematically add new, uniformly distributed queries to the database and determine the associated movements using the structured reinforcement learning approach.

A. Exploiting the Structure of the Trajectory Space

Given a new query point, the generalization function (1) can calculate the appropriate DMP. To improve the performance of the generalization function, we propose to add additional movements into the existing database of movements, where additional movcements are generated by reinforcement learning. By exploiting the available data, we can organize reinforcement learning around two different types of parameters: query point $\mathbf{q}$, which dimensionality is always low, and DMP parameters ($\mathbf{w}$, τ and $\mathbf{g}$), which combined dimensionality is high. As noted in [18], many tasks require high accuracy only on a low-dimensional manifold of the complete movement space. In all our practical examples, the dimensionality of the query points was between 1 and 3.

We propose to compute the optimal movement at a new query point in two steps. First we perform the search for an optimal query point, from which the DMP parameters are computed using Eq. (1). Since this exploration process is limited to the trajectory manifold defined by (1), the obtained DMP parameters can be further improved by exploration in the full DMP parameter space. We considered two reinforcement learning algorithms to implement these steps: PoWER [19]–[21] which operates using terminal reward only, and PI^2 [5], [22], [23], which can also take into account intermediate rewards. Intermediate rewards are important to ensure that the shape of the new trajectory remains similar to the shape of initial trajectories (see Section III-B).

Thus to learn the weights $\mathbf{w}$ that define the shape of the movement, we propose to apply PI^2. To learn the remaining DMP parameters, i. e. the time duration of the movement τ and the goal of the movement $\mathbf{g}$, and to tune the initial query point $\mathbf{q}$, we use the PoWER algorithm.

B. Augmenting the Example Database

The search process of Section III-A enables the robot to find a new movement $\mathbf{M}$ that solves the task at query point $\mathbf{q}$. We can now expand the example database used to compute generalization function (1) with a pair $(\mathbf{M}, \mathbf{q})$. However, not every movement is suitable to be added to the database. Statistical learning can only be successful if the movements in the database smoothly transition between each other. Since generalization function (1) is smooth, it always generates smooth movement patterns. Unfortunately, in general it is not guaranteed that the optimal movement lies on the manifold defined by the current estimate of the generalization function. Hence we need to allow the robot to search also outside of this manifold. This search can be accomplished by reinforcement learning, but this could result in discontinuities in the movement pattern because the solution is not unique and the robot could find a different type of solution than the ones in the database.

To ensure that this does not occur, we utilize the first movement approximation computed by the generalization function (1). This function computes the reference DMP $\tilde{\mathbf{y}}$, which is guaranteed to transition smoothly between the neighboring DMPs (movements) in the database. Let $\mathbf{y}$ denote the current estimate for the desired movement. We can define the immediate cost function r that can be used by PI^2 as follows

$$r(t) = \kappa \|\mathbf{y}(x(t)) - \tilde{\mathbf{y}}(x(t))\|^2 + \dot{\mathbf{y}}^T \mathbf{\Gamma} \dot{\mathbf{y}}, \qquad (6)$$

where x is the common phase of the DMPs and $\mathbf{\Gamma}$ is the regularization matrix. Such immediate reward ensures that new trajectories generated by PI^2 stay close to the generalized trajectories, which results in smooth transitions between the neighboring trajectories.

PI^2 has only one free parameter, i. e. the exploration noise. In general the exploration noise can be set significantly smaller than usually because most of the exploration is expected to occur in the query point space and not in the full parameter space. We will omit the details of reinforcement learning algorithm PI^2 in this paper. See [5], [7] for more details about PI^2.

Let R be the terminal reward function and let r be the immediate cost function for the desired task. Reinforcement learning in the trajectory space parametrized by $\mathbf{q}$ and its neighborhood can then be defined as follows:

1) Estimate the DMP (first approximation) for a new query point $\mathbf{q}$ using statistical generalization (1). Execute the resulting DMP and compute the terminal reward. Use these data to initialize the PoWER algorithm and use the weights $\mathbf{w}$ of the DMP to initialize PI^2 algorithm.

2) The learning process is stopped if the terminal reward R is above a given threshold or the maximum number of iteration has been reached. Otherwise continue with one reinforcement learning epoch.
3) Repeat K times: obtain a new estimate $\mathbf{q}'$ with exploration noise. For each new $\mathbf{q}'$, compute the DMP parameters using statistical generalization (1) and add the exploration noise. Execute the resulting DMP and calculate the immediate costs and terminal reward function R.
4) Use PoWER to compute new goal $\mathbf{g}$ and time duration τ on the movement and use PI^2 algorithm to compute new weights $\mathbf{w}$ using the results of previous K steps.
5) Use the output of PoWER and PI^2 algorithms and execute the DMP to get the terminal reward R. Continue with step 2.

If the learning process has stopped due to the satisfactory terminal reward, the newly calculated movement can be added to the database of example movements.

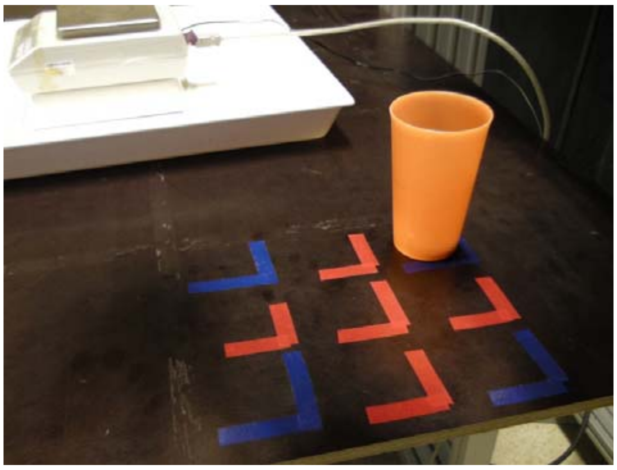


Fig. 2. Experimental setup. Movements for 4 different glass positions (shown in blue) with 2 different quantities of water were initially acquired. In red are the positions on the table that were later added to the database. The scale used in the experiments is in the background.

IV. Experimental Results

In our experiments we focused on showing that the proposed approach addresses the following two key issues: 1) the generation of task solution trajectories that are similar to the example trajectories in the existing database, and 2) augmenting the database with autonomously learned trajectories to improve the accuracy of statistical generalization, thereby increasing the overall performance of the task.

The robot's experimental task was to learn how to pour the same quantity of water (0.2 l) into a glass regardless of the amount of water in the bottle (see also Fig. 1). For this purpose we collected 8 initial movements at 4 different glass positions, as shown in Fig. 2, with 2 different quantities of water (0.3 l and 1 l) in the bottle. In this case the query point $\mathbf{q}$ is defined as $[x, y, v]^T$, where x and y denote the position of the glass on the table and v denotes the volume of the water in the bottle. We used the following terminal reward function

$$R = \begin{cases} 5 \cdot (0.2 - |0.2 - m_g| - m_s) & 0 \le R \le 1 \\ 0 & \text{otherwise,} \end{cases} \qquad (7)$$

where m_g is the amount of water in the glass and m_s the amount of spilled water. This way we ensure that the robot learns how to pour without spilling. We used a scale to measure the final amount of water in the glass and the force-torque sensor in the wrist to estimate the amount of all water poured from the bottle. The difference between the two quantities provides the amount of spilled water.

The eight task demonstrations were used to provide initial data for statistical generalization. The demonstrated trajectories in joint space were encoded as DMPs with 20 radial basis functions for each joint trajectory. It turned out that 20 is the optimal number of radial basis functions depending on the complexity and length of the pouring movements. Using the learning process described in Section III, 40 new movements together with the associated query points were added to the database. On the average, the reinforcement learning process of Section III needed about 10 executions of the pouring behavior to find a new movement with satisfactory reward. This relatively low number of trials was due to the fact that algorithm can exploit the previously estimated structure of the solution space. Lets analyze the shape of the newly learned movements. Fig. 3 shows the trajectories at 9 different positions on the table for the degree of freedom (5th joint), at which the largest movement was performed during the execution of the pouring behavior. The trajectories exhibit similar shape and transition smoothly from one to another. Fig. 4 show the original and the autonomously learned trajectories in 3-D Cartesian space after being transformed via forward kinematics. Again we can see that the trajectories found via reinforcement learning are qualitatively similar in shape to the demonstrated trajectories. Note that the new trajectories are smoother than the demonstrated trajectories, which is due to the regularization term in reward function (6). Thus we can

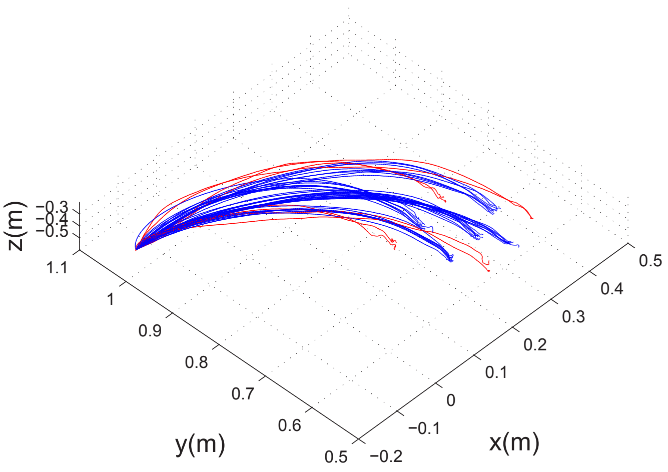


Fig. 4. Similarity of the newly acquired movements. In red are the original example trajectories obtained by kinesthetic guiding and in blue the newly acquired trajectories generated by the proposed reinforcement leaning process. All trajectories were originally determined in joint space and were mapped onto 3-D Cartesian space via forward kinematics.

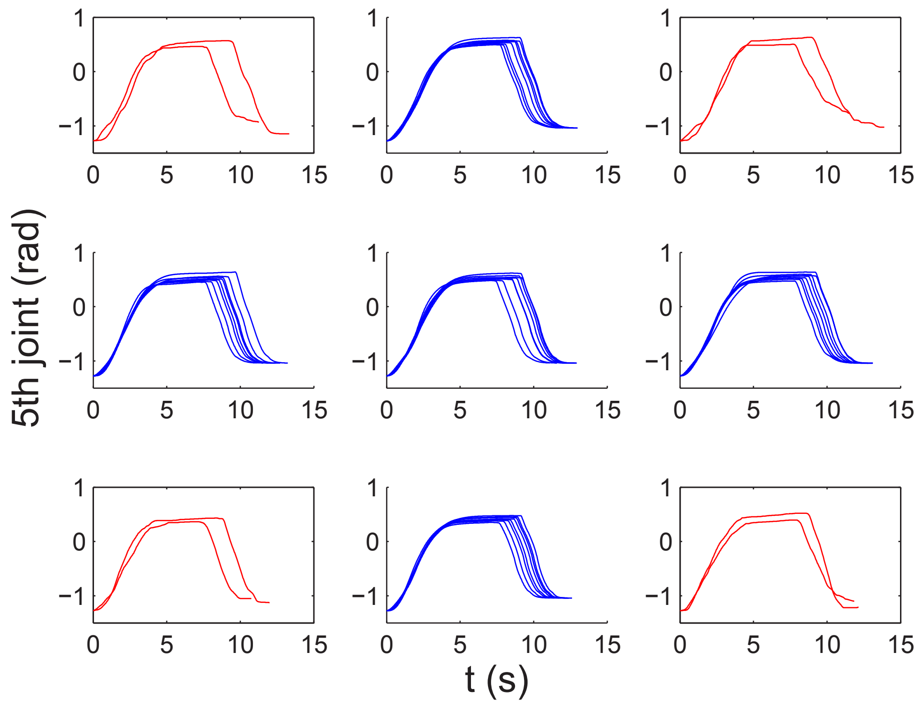


Fig. 3. Similarity of the newly acquired movements. In red are the original example trajectories obtained by kinesthetic guiding and in blue the newly acquired trajectories generated by the proposed reinforcement leaning process. For easier representation only trajectories of 5th joint, that is changing the most, are shown in the graphs.

expect that the augmented movement database will be suitable for statistical generalization.

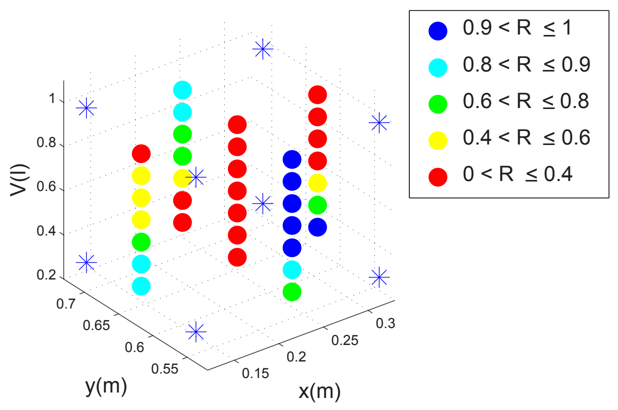
To evaluate the performance of generalization, we tested the accuracy of generalization at in-between query points (shown in Fig. 5), that is at query points that do not exactly coincide with the training query points. Fig. 6 illustrates the generation of new trajectories by statistical generalization. As expected, the generalized trajectories are smooth and similar to the nearest trajectories in the database. Fig. 5 shows that a considerable increase in performance can be achieved this way. When new test movements were generated by statistical generalization, which used only 8 initial movements as training data, the average error of the glass filling process was 0.08 l. This was reduced to 0.02 l for cases in which statistical generalization used also the newly acquired movements, that is altogether 8 + 40 example movements. The amount of spilled water was also reduced. With the initial 8 movements, statistical generalization produced movements that caused 0.03 l of water to be spilled on the average. With the additional 40 example movements, there was no spilling with generalized DMPs. We can thus claim that the proposed approach is successful at finding new movements for the database and can autonomously increase the performance of generalization.

V. Conclusion

The main result of this paper is a new approach to autonomously augment the database of example movements, which was initially obtained in a supervised manner, for example by human demonstration. The newly acquired data can be used to improve the accuracy of generalization and consequently the performance of task execution. We demonstrated experimentally that using the proposed approach, the robot can improve its performance without additional help of the teacher. We believe that our approach addresses one of the fundamental problems of imitation, that is how to transition from directly mimicking the teacher's movements to practicing, where the initial movements are modified and new data are added to the existing knowledge base.

The integration of autonomous exploration with statistical generalization also enabled us to define a new, structured reinforcement learning algorithm, which can find new example movements in the neighborhood of the estimated trajectory manifold much quicker than standard reinforcement learning algorithms. Compared to the method proposed in [6], which also combined learning in low-dimensional spaces using value function approximation with learning in high-dimensional spaces using PI^2, the approach proposed in this paper uses the results of statistical generalization in addition, thereby further increasing the speed of convergence. The key to this accelerated convergence of the learning process lies in the fact that thanks to statistical generalization function (1), a significant part of the search process could be moved from the high-dimensional trajectory space spanned by DMPs to the low-dimensional space determined by query points, which are defined as task-relevant parameters that characterize the task. Another important issue is that statistical generalization provides a reference movement, which can be used to define immediate rewards in terms of the distance between trajectories determined by reinforcement learning and the reference trajectory. This is needed to ensure that the newly found trajectories are similar to the trajectories in the database.

The developed system is of course not limited to the learn-



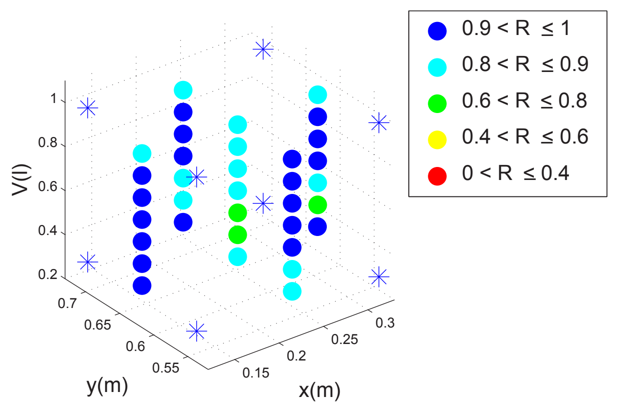


Fig. 5. Results showing the improved performance with the increasing size of the database. Points in the graphs are the test points situated in-between query points. R denotes terminal reward (7), which is normalized to values between zero and one. Demonstrated examples are marked as blue stars and are situated in the corners of the search space. First graph shows terminal rewards at test points where only 8 demonstrated examples were in the trajectory database. Second graph shows terminal rewards at the same test points after additional 40 movements were refined and stored in the trajectory database.

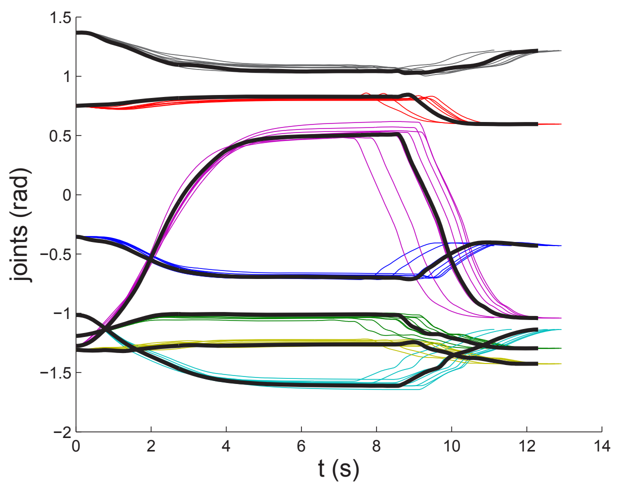


Fig. 6. Calculation of new joint trajectories using statistical generalization. In black are the generalized trajectories and in color the closest neighboring trajectories from the database.

ing of the pouring behaviour. There are only two components that are action-specific: the definition of the query point space and the definition of terminal reward functions R. Note that the immediate reward function r of Eq. (6) is not dependent on the actual behaviour to be learned. Due to space limitations, the analysis in this paper focused on pouring. However, in our previous papers [9], [11], [24] we showed that the algorithms which form the basis of the proposed system can be applied to learn many different behaviors including reaching, throwing, drumming, and kendama.

Acknowledgements

Research leading to these results was supported by the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) grant agreement no. 270273, Xperience.

References

[1] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[2] H. Miyamoto, S. Schaal, F. Gandolfo, H. Gomi, Y. Koike, R. Osu, E. Nakano, Y. Wada, and M. Kawato, "A kendama learning robot based on bi-directional theory," *Neural Networks*, vol. 9, no. 8, pp. 1281–1302, 1996.

[3] J. Morimoto and K. Doya, "Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning," *Robotics and Autonomous Systems*, vol. 36, pp. 37–53, 2001.

[4] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, pp. 682–697, 2008.

[5] E. A. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *Journal of Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[6] M. Tamosiunaite, B. Nemec, A. Ude, and F. Wörgötter, "Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives," *Robotics and Autonomous Systems*, vol. 59, no. 11, pp. 910–922, 2011.

[7] F. Stulp, E. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1360–1370, 2012.

[8] J. Kober and J. Peters, "Learning motor primitives for robotics," in *Proc. IEEE Int. Conf. Robotics and Automation*, Kobe, Japan, 2009, pp. 2112–2118.

[9] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[10] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning parametric dynamic movement primitives from multiple demonstrations," *Neural Networks*, vol. 24, no. 5, pp. 493–500, 2011.

[11] D. Forte, A. Gams, J. Morimoto, and A. Ude, "On-line motion synthesis and adaptation using a trajectory database," *Robotics and Autonomous Systems*, vol. 60, pp. 1327–1339, 2012.

[12] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013.

[13] S. Calinon, F. D'halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation: An approach based on Hidden Markov Model and Gaussian Mixture Regression," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 44–54, 2010.

[14] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian Mixture Models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[15] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1463–1467, 2008.

[16] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Dynamical movement primitives: Learning attractor models for motor behaviors,” *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[17] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.

[18] C. G. Atkeson, A. W. Moore, and S. Schaal, “Locally weighted learning for control,” *AI Review*, vol. 11, pp. 75–113, 1997.

[19] J. Kober, “Reinforcement learning for motor primitives,” in *Masters thesis*, University of Stuttgart, Germany, August 2008.

[20] J. Peters and S. Schaal, “Policy gradient methods for robotics,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2006, pp. 2219–2225.

[21] F. Guenter, M. Hersch, S. Calinon, and A. Billard, “Reinforcement learning for imitating constrained reaching movements,” *Advanced Robotics*, vol. 21, pp. 1521–1544, 2007.

[22] H. J. Kappen, W. Wiegerinck, and B. van den Broek, “A path integral approach to agent planning,” in *AAMAS*, 2007.

[23] B. van den Broek, W. Wiegerinck, and B. Kappen, “Graphical model inference in optimal control of stochastic multi-agent systems,” in *Journal of Artificial Intelligence Research*, vol. 32, 2008, pp. 95–122.

[24] B. Nemec, D. Forte, R. Vuga, M. Tamošiunaite, F. Wörgötter, and A. Ude, “Applying statistical generalization to determine search direction for reinforcement learning of movement primitives,” in *2012 12th IEEE-RAS Int. Conf. Humanoid Robots*, Osaka, Japan, 2012, pp. 65–70.

Robotics and Autonomous Systems 75 (2016) 340–351

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Adaptation and coaching of periodic motion primitives through physical and visual interaction

Andrej Gams [a,*], Tadej Petrič [a,d], Martin Do [b], Bojan Nemec [a], Jun Morimoto [c], Tamim Asfour [b], Aleš Ude [a,c]

[a] Humanoid and Cognitive Robotics Lab, Department for Automatic, Biocybernetics and Robotics, Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia
[b] High Performance Humanoid Technologies Lab, Institute of Anthropomatics and Robotics, Karlsruhe Institute of Technology, Adenauerring 2, 76131, Karlsruhe, Germany
[c] Department of Brain Robot Interface, ATR Computational Neuroscience Labs, Kyoto 619-0288, Japan
[d] Biorobotics Laboratory, École Polytechnique Fédérale de Lausanne, Station 14, CH-1015 Lausanne, Vaud, Switzerland

HIGHLIGHTS

- An intuitive and user friendly system for transferring of skills from a person to a robot.
- It allows online learning and adaptation of motion trajectories.
- It allows adaptation of trajectories through human coaching, from either force or visual feedback.
- It is based on the dynamic motion primitives framework.
- Surface wiping use-case through non-rigid contact is demonstrated and evaluated.

ARTICLE INFO

Article history:
Received 19 April 2015
Received in revised form
19 August 2015
Accepted 4 September 2015
Available online 25 September 2015

Keywords:
Dynamic movement primitives
Force control
Coaching
Human–robot interaction

ABSTRACT

In this paper we propose and evaluate a control system to (1) learn and (2) adapt robot motion for continuous non-rigid contact with the environment. We present the approach in the context of wiping surfaces with robots. Our approach is based on learning by demonstration. First an initial periodic motion, covering the essence of the wiping task, is transferred from a human to a robot. The system extracts and learns one period of motion. Once the user/demonstrator is content with the motion, the robot seeks and establishes contact with a given surface, maintaining a predefined force of contact through force feedback. The shape of the surface is encoded for the complete period of motion, but the robot can adapt to a different surface, perturbations or obstacles. The novelty stems from the fact that the feedforward component is learned and encoded in a dynamic movement primitive. By using the feedforward component, the feedback component is greatly reduced if not completely canceled. Finally, if the user is not satisfied with the periodic pattern, he/she can change parts of motion through predefined gestures or through physical contact in a manner of a tutor or a coach.

The complete system thus allows not only a transfer of motion, but a transfer of motion with matching correspondences, i.e. wiping motion is constrained to maintain physical contact with the surface to be wiped. The interface for both learning and adaptation is simple and intuitive and allows for fast and reliable knowledge transfer to the robot.

Simulated and real world results in the application domain of wiping a surface are presented on three different robotic platforms. Results of the three robotic platforms, namely a 7 degree-of-freedom Kuka LWR-4 robot, the ARMAR-IIIa humanoid platform and the Sarcos CB-i humanoid robot, depict different methods of adaptation to the environment and coaching.

1. Introduction

Learning by demonstration, as an approach of acquiring trajectories in robotics [1], can only be effective if it enables adaptation of the demonstrated policy to the current situation of

* Corresponding author.
E-mail addresses: andrej.gams@ijs.si (A. Gams), tadej.petric@ijs.si (T. Petrič), martin.do@kit.edu (M. Do), bojan.nemec@ijs.si (B. Nemec), xmorimo@atr.jp (J. Morimoto), asfour@kit.edu (T. Asfour), ales.ude@ijs.si (A. Ude).

http://dx.doi.org/10.1016/j.robot.2015.09.011
0921-8890/© 2015 Elsevier B.V. All rights reserved.

the task or the environment [2]. For example, when learning a wiping behavior, which is a rather trivial skill for humans, the robot must acquire the correct characteristics of motion, but must also maintain contact with the surface it is wiping. Such skill transfer from a human to a robot, where not only the motion but also the constraints imposed by the task are important, is the motivation behind this paper. We propose a system that enables a robot to learn actions which require continuous non-rigid contact with the environment through human demonstrations and interactive coaching. The coaching mechanisms enable a human teacher to efficiently guide the robot towards a goal-directed execution.

Learning by demonstration often exploits the means of encoding the motion characteristics of an action by generalizing demonstrated trajectories from the performing subject and the current situation. Different approaches exist, for example splines and wavelets [3,4], which are effective for imitation learning, but do not allow easy online modulation. Another options are Gaussian Mixture Regression [5] and Gaussian Mixture Models [6,7], used to estimate the entire attractor landscape of a movement skill from several demonstrations. To ensure stability of the dynamical system towards an attractor point, a constraint optimization problem in a nonconvex optimization landscape needs to be solved. Yet another option is the use of Hidden Markov Models [8]. Different dynamical systems can also be used.

Another type of dynamical systems are dynamic movement primitives (DMPs) [9], which focus on the representation of single movements by a set of differential equations. A DMP can represent a control policy in a compact way and its attractor landscape can be adapted by only changing a few parameters. Compared to representations proposed in [6,7], only a simple system of linear equations needs to be solved. DMPs can be used for representing classes of movements using statistical learning techniques [2,10], for combining trajectories in a dynamic way [11,12], and for reinforcement learning [13–16]. In this paper we exploit the DMP framework to enable continuous non-rigid contact with the environment, based on force feedback.

Adaptation of learned trajectories to external feedback was previously discussed in different settings and applications, using different trajectory representations. The use of force feedback to learn and improve task execution was widely considered in robotics, see for example the book chapter by Villani and De Schutter [17]. One of the best known approaches is the method proposed by Hogan [18], where force feedback is used to change the output velocity of a manipulator. This technology is the basis for the DMP adaptation proposed in this paper.

DMPs themselves were already used for adaptation to forces. In [19] the authors used an interaction force and the parallel force/position control law to modulate the velocity of the dynamical system. Pastor et al. [20,21] have also combined force controllers and DMPs in an approach for modifying DMPs at the acceleration level, allowing for reactive and compliant behaviors. They used the demonstrated trajectory profiles as reference, while [22] applied reinforcement learning to further optimize the behavior. A modulation approach at the acceleration level of a DMP for physically coupled dual-agent tasks was reported by Kulvicius et al. [23], but the learning was applied to acquire appropriate feedback gains instead of reference trajectories. On the other hand, Gams et al. [24] utilized coupled DMPs with force feedback at the velocity level. Combined with iterative learning control, their approach can achieve the desired force interaction for rigid contacts. Iteratively approaching a desired behavior has been applied for in some programming by demonstration approaches. For example, Sauser et al. [25] showed grasp adaptation through human corrections, while Calinon and Billard [26] showed gesture learning. On the other hand, iteratively approaching a desired behavior was also shown in combination with DMPs in a peg-in-hole task [27], where reference force-torque profiles were used as means for autonomously improving the execution performance. In this work the force controller was not applied within the DMP framework. Haptic feedback for improving the teacher demonstration was also used by Rozo et al. [28], who addressed the problem of what to imitate based on the mutual information between perceptions and actions. HMMs and GMR were used to encode the demonstrations and for the robotic execution of the learned tasks. The method was augmented in order to be applicable also for the task of pouring [29]. Adaptation of trajectories is not limited to one-arm behaviors. An approach for bimanual operation based on dynamical systems by adding local corrective terms was discussed by Calinon et al. [30].

In this paper we consider the transfer of skills from a human to a robot through coaching. The transfer is not limited to the motion, but includes the execution of the task in contact with the environment. We consider two problems of on-line motion adaptation for the actual completion of the task. The first is the adaptation to the external environment in order to achieve desired forces of non-rigid contact throughout the complete trajectory. The second is adapting the trajectories to the interventions of an instructor, modifying the trajectories through physical contact or with the use of predefined gestures. The interaction puts the instructor into the role of a tutor who coaches the robot to achieve the desired performance. Both adaptation to the environment and coaching rely on the use of a unified trajectory representation, i.e. the dynamic movement primitives (DMPs). The combination creates an intuitive and user-friendly interface to learning and modifying robotic trajectories with the potential of creating complex object-interaction trajectories.

Not many papers describe adaptation of learned trajectories for non-rigid contacts. Initial results of DMP adaptation methods, expanded on in this paper, were presented in [31,32]. The approach was expanded on by Ernesti et al. [33] to include transient motions and [34] to include structural bootstrapping from experience. Wiping with a robot has also been studied from other perspectives, including using dynamic models and operational space dynamics [35].

Coaching has been applied also in context of other robotic tasks. Gruebler et al. [36] used voice commands as a reward function in their learning algorithm. Verbal instructions of non-experts were used to modify movements obtained by human demonstration [37]. Physical contact was also used, for example, by Lee and Ott [38] who used kinesthetic teaching with iterative updates to modify the behavior of a humanoid robot. Coaching based on gestures and obstacle avoidance algorithms was applied to DMPs [39]. This approach is expanded on in this paper with force feedback.

In the next section we provide the basics of DMPs and the algorithm of encoding them. Section 3 provides the core algorithm of the adaptation approach. Three different methods are explained. Coaching, as the means to adapt parts of the trajectory based on the user input is explained in Section 4, followed by the results in Section 5 and a discussion with conclusions.

2. Learning of periodic dynamic movement primitives

In this paper we build on periodic dynamic movement primitives. For the sake of completeness we provide the basics of the DMP notation and an algorithm for extracting the frequency of the demonstrated signals. The algorithm of learning of weights that encode a DMP follows. It is the basis for both adaptation to external force and the coaching algorithms.

2.1. Periodic DMPs

The formulation of DMPs in this paper is based on [2]. For a complete DMP overview see [9]. The description is for clarity limited to a single degree of freedom (DOF), i.e. one of the external task-space coordinates, denoted by y. Temporally scaled velocity is denoted by z. Note that DMPs can be applied to joint space coordinates as well. y and z should not be mistaken with the axes of a coordinate system, which are in this paper denoted by x_p, y_p, z_p.

A nonlinear system of differential equations that defines a periodic DMP is given by

$$\dot{z} = \Omega\left(\alpha_z\left(\beta_z\left(g - y\right) - z\right) + f(\phi)\right), \tag{1}$$

$$\dot{y} = \Omega z. \tag{2}$$

The nonlinear part of (1), $f(\phi)$, known as the forcing term, is comprised of a linear combination of N radial basis functions $\Gamma_i(\phi)$

$$f(\phi) = \frac{\sum_{i=1}^{N} w_i \Gamma_i(\phi)}{\sum_{i=1}^{N} \Gamma_i(\phi)} r. \tag{3}$$

Radial basis functions $\Gamma_i(\phi)$ are defined by

$$\Gamma_i(\phi) = \exp\left(h_i\left(\cos\left(\phi - c_i\right) - 1\right)\right). \tag{4}$$

Parameter r is the amplitude control parameter, $h_i > 0$ are the widths of the kernels and c_i spreads them equally along the phase ϕ from 0 to 2π in N steps. The parameters $\alpha_z,\ \beta_z,\ > 0$ and $\alpha_z = 4\beta_z$ make the system (1)–(2) critically damped. The system oscillates as given by $f(x)$ around the goal g. To realize multiple DOFs we use separate sets of (1)–(2), and a single canonical system given by (5) to synchronize them through the common phase.

The phase variable ϕ provides the indirect dependency on time. It can increase with constant rate, where the parameter Ω denotes the frequency

$$\dot{\phi} = \Omega. \tag{5}$$

When learning the frequency does not have to remain constant, but needs to be estimated, for example with adaptive frequency oscillators as in [40,41]. In our case we used the system proposed in [41] for online extraction of frequency of motion and to encode one period of motion with the weights $w_i, i = 1, \ldots, N$, where N is the number of kernel functions. The frequency estimation is based on a feedback structure containing an adaptive frequency oscillator and an adaptive Fourier series.

The adaptive frequency oscillator is governed by the following feedback structure

$$\dot{\phi} = \Omega - Ke_o \sin\phi, \tag{6}$$

$$\dot{\Omega} = -Ke_o \sin\phi, \tag{7}$$

$$e_o = y_{\text{demo}} - \hat{y}, \tag{8}$$

where K is the coupling strength, ϕ is the phase of the oscillator, e_o is the input into the oscillator and y_{demo} is the input signal. The feedback loop signal $\hat{y}$ in (8) is provided by the Fourier series

$$\hat{y} = \sum_{a=0}^{M}(\alpha_a \cos(a\phi) + \beta_a \sin(a\phi)). \tag{9}$$

Here M is the number of components of the dynamic Fourier series and α_a, β_a are the amplitudes associated with the series. They are estimated as follows:

$$\dot{\alpha}_a = \eta\cos(a\phi)e_o, \tag{10}$$

$$\dot{\beta}_a = \eta\sin(a\phi)e_o, \tag{11}$$

where η is the learning constant and $a = 0, \ldots, M$.

2.2. *Learning of DMPs*

To encode a periodic trajectory as a DMP, we need to determine the duration of one period of motion, for example by using the above-described adaptive frequency oscillators. Once the frequency of the demonstrated motion is established, we need to learn the weights of the DMP to encode the shape of the demonstrated motion. The latter is accomplished using incremental locally weighted regression (ILWR) [42]. The target data for fitting is constructed from the demonstration trajectory y_{demo}, which is the desired trajectory of motion. The target function for fitting, originating from (1)–(2) is therefore

$$f_{targ} = \frac{1}{\Omega^2}\ddot{y}_{\text{demo}} - \alpha_z\left(\beta_z\left(g - y_{\text{demo}}\right) - \frac{1}{\Omega}\dot{y}_{\text{demo}}\right). \tag{12}$$

It is obtained by matching y from (1)–(2) to y_{demo}, z to $\dot{y}_{\text{demo}}/\Omega$, and $\dot{z}$ to $\ddot{y}_{\text{demo}}/\Omega$. This means that we basically learn how to force the otherwise critically damped spring–mass system given by the linear part of (1)–(2) to follow the desired trajectory.

Given f_{targ}, w_i is updated incrementally for each time-step j as

$$w_{i,j+1} = w_{i,j} + \Gamma_{i,j+1}P_{i,j+1}re_j \tag{13}$$

$$P_{i,j+1} = \frac{1}{\lambda}\left(P_{i,j} - \frac{P_{i,j}^2 r^2}{\frac{\lambda}{\Gamma_i} + P_{i,j}r^2}\right) \tag{14}$$

$$e_j = f_{\text{targ},j} - w_{i,j}r. \tag{15}$$

Γ_i are the kernel functions. P_i, in general, is the inverse covariance of w_i [43]. The recursion is started with $w_i = 0$ and $P_i = 1$. r is the amplitude gain. The forgetting factor is defined by $\lambda \le 1$. Useful range of λ is between 0.97 and 1. If $\lambda < 1$, then the incremental regression gives more weight to recent data, meaning that it tends to forget older ones.

3. Adaptation to environment

Adaptation to the environment, as proposed in this paper, assumes that the environment cannot change rapidly, i.e. an object, such as a table or a kitchen sink, does not rapidly change shape or height. The setting of the environment, on the other hand, can be completely arbitrary. This assumption allows gradual adaptation of motion through learning, and is the basis of the proposed algorithm. If the environment does not change rapidly, then a correct reference of motion (if followed) will achieve the desired behavior. The referential trajectory is the output of the DMP, and can be interpreted as a feed-forward control signal. This is augmented with the feedback control loop for instantaneous reaction, and to allow gradual adaptation. The use of the learned feed-forward component (the output of the DMP) reduces the need for feedback adaptation, which allows for greater accuracy. Furthermore, the use of the DMP allows for standard DMP features, such as easy modulation with the change of only a few parameters.

In this paper we propose two means of applying force feedback to change the output of the DMP, i.e. the feed-forward component of the control signal. The first is in changing the reference for learning a DMP. The second is in bootstrapping the force signal directly into the DMP weight adaptation.

3.1. *Changing the reference*

The original use of DMPs allows the encoding of demonstrated trajectories for imitation, i.e., the demonstrated trajectory is the reference. If the reference is changing over time, so is the output of the DMP. In this algorithm we exploit force feedback to change the reference of the DMP, i.e. y_{demo}. The change of the reference trajectory occurs through the change of the end-effector velocity as a function of force, known as the velocity-resolved approach [17]

$$\mathbf{v}_r = \mathbf{S}_v\mathbf{v}_v + \mathbf{S}_F(\mathbf{K}_i\mathbf{e}_f + \mathbf{K}_p\dot{\mathbf{e}}_f), \tag{16}$$

$$\mathbf{e}_f = \mathbf{F}_0 - \mathbf{F}_m. \tag{17}$$

The variable $\mathbf{v}_r$ stands for the resolved velocities vector, $\mathbf{S}_v$ for the velocity selection matrix, $\mathbf{v}_v$ for the desired velocities vector, $\mathbf{K}_i$, $\mathbf{K}_p$ for the force gain matrices, $\mathbf{S}_F$ for the force selection matrix, $\mathbf{F}_m$ for the measured force and $\mathbf{F}_0$ the desired force of contact. Essentially, the selection matrices $\mathbf{S}_v$ and $\mathbf{S}_F$ determine which directions of motion are affected by the force. They are determined by the user, who knows which directions of motion need to be altered.

To get the desired positions we use

$$\begin{aligned}\mathbf{y}_r &= \mathbf{y}_{\text{demo}} + \mathbf{S}_F\left(\int \mathbf{v}_r dt\right)\\ &= \mathbf{y}_{\text{demo}} + \mathbf{S}_F\left(\int \mathbf{K}_i(\mathbf{F}_0 - \mathbf{F}_m)dt + \mathbf{K}_p(\mathbf{F}_0 - \mathbf{F}_m)\right). \qquad (18)\end{aligned}$$

Here $\mathbf{y}_r$ is the resolved position (and possibly also orientation) of the robot, taking the place of $\mathbf{y}_{\text{demo}}$. We see in (18) that $\mathbf{y}_{\text{demo}}$ has both an integral and a proportional feedback loop, with $\mathbf{K}_p$ being the proportional gain. Combining the two allows for zero steady-state error within the integral loop and faster reactions to possible unforeseen perturbations within the proportional loop.

When wiping a flat horizontal surface, such as an average table, (16)–(18) become less complex. In this case the robot needs to establish contact in a vertical direction, typically z. We obtain: $\mathbf{S}_v = 0$, $\mathbf{K}_i = \text{diag}(0, 0, k_i, 0, 0, 0)$, $\mathbf{K}_p = \text{diag}(0, 0, k_p, 0, 0, 0)$, $\mathbf{S}_F = \text{diag}(0, 0, 1, 0, 0, 0)$. Only the desired end-effector height z_p is modified in each discrete time step Δt, and (16) becomes (19).

$$\dot{z}_p(t) = k_i e_f(t) + k_p \dot{e}_f(t), \qquad (19)$$

$$e_f(t) = F_0(t) - F_z(t). \qquad (20)$$

Taking into account the initial condition and numerical integration, it results in (21)

$$z_p(t) = z_{p0} + k_i e_f(t)\Delta t + k_p e_f(t). \qquad (21)$$

Here z_{p0} is the initial z_p value (starting height), k_i and k_p are positive constants for force gains, F_z is the measured force in the z_p direction and F_0 is the desired force of contact. The movement is constant in $-z_p$ direction when there is no contact, or maintains contact force F_0 when an object is encountered.

The learning of the DMP in the direction that is being modified by force, for example in z_p direction as shown in (19)–(21), is done by modifying the weights w_i for the selected DOF (determined by $\mathbf{S}_F$) in every time-step by using incremental locally weighted regression as given by (13)–(15). The *demonstrated* trajectory is being constantly modified by the force feedback and therefore the DMP weights are constantly re-evaluated until a steady-state is reached. Since this approach uses the position of the end-effector as input, and not the force, it has no difficulties with the noisy measured force signal.

The admittance control scheme given by (16)–(18) is subject to the gains $\mathbf{K}_i$ and $\mathbf{K}_p$. High gains will result in fast reactions when encountering a force. At the same time, being subject to time discretization, specifically at low sampling rates, too high gains may produce instabilities. A trade-off has to be made based on the desired behavior. We empirically set $\mathbf{K}_i$ and $\mathbf{K}_p$ values and also limited the force feedback to a maximum absolute value.

3.2. Direct adaptation of the DMP

The change of y_{demo} will inherently cause some delay typical for feedback controllers. To cancel the delay of the algorithm that changes the reference for learning, we exploit the incremental weight fitting algorithm (13)–(15) for learning of periodic DMPs. The basic idea here is that we replace the error signal for weight fitting associated with imitation with a different signal, for example the difference between the measured and desired forces in force interaction.

Let's assume that a given trajectory is encoded as a DMP with weights $\mathbf{w}$. The trajectory follows the demonstrated trajectory if the error signal in (15) is equal to $e_j = 0$, meaning that $\mathbf{w}$ does not change. We now replace the imitation-related error signal (15) with a force-dependent term

$$e_j = k_l(F_0 - F_m). \qquad (22)$$

By using e_j from (22) in (13), the weights of the DMP will be updated whenever the measured and the desired forces are different. Therefore it will adapt the trajectory to fulfil the condition of (22), which is that the actual force of contact F_m is the same as the desired force F_0 in the given direction. Parameter k_l is a positive constant, determined empirically. Note that the implementation of adaptation should take care that the values of the inverse covariance P_i do not decrease to $P_i \cong 0$ as this will stop the adaptation, given that the update of weights is multiplied by P.

A feedback term can also be added to the acceleration level of the DMP for instantaneous reaction, changing (1) into

$$\dot{z} = \Omega\left(\alpha_z\left(\beta_z\left(g - y\right) - z\right) + f(\phi) + d(F)\right). \qquad (23)$$

The feedback term can be a simple proportional control law with gain $k_{fb} > 0$, for example $d(F) = k_{fb}(F_0 - F)$. In this paper we name the trajectory adaptation method based on (22) the *Direct* method.

In simulation, where we can model the forces of contact with displacement of the elastic environment with stiffness k_{env}, we can rewrite (22) into $F = k_{\text{env}}(y_0 - y) = k_{\text{env}}\bar{y}$. Any difference of forces at end-effector will therefore introduce a position difference $k_l k_{\text{env}}(y_0 - y)$, which will through (13) reflect in $f(\phi)$. From (1)–(2) we can see that through integration of the DMP differential equations, $f(\phi)$ (and consequently $y_0 - y$) is integrated twice, which results in a slight delay.

From a physical standpoint, the linear part of (1) represents accelerations of a spring–mass system, while $f(\phi)$ provides the modification for accelerations that force the system to follow the desired trajectory (hence earning the name *forcing* term). In order to exclude the above mentioned delay from position-difference integration, we need to change (22) so that it provides proper accelerations for the second order DMP spring–mass system. These are calculated according to (12). We therefore write

$$e_j = \frac{1}{\Omega^2}k_2\ddot{\bar{y}} - \alpha_z\left(\beta_z\left(g - k_2\bar{y}\right) - \frac{1}{\Omega}k_2\dot{\bar{y}}\right), \qquad (24)$$

where $k_2\bar{y} = k_l k_{\text{env}}(y_0 - y)$ models the forces. In this paper we name the trajectory adaptation method based on error signal (24) the *Derived* method.

Table 1 provides the basic characteristics and differences of the three methods described in this section.

4. Coaching

During trajectory learning the demonstrator repeats several periods of motion and the collected data are given as reference to the incremental locally weighted regression. The trajectory is learned, but it might not exactly perform the desired task, as is often the case when giving instructions to another person on how to perform something. When the resulting motion is not satisfactory, the demonstrator can coach the other person, specifying how to alter the motion in certain parts, or simply showing the complete motion again.

In order to avoid re-learning of the complete trajectory, we can exploit the same mechanism as was applied for the *Direct* method to change only parts of the trajectory. We again rely on changing (15). If $e_j = 0$, there is no learning and the robot just repeats the trajectory it learned during the demonstration. Again, for a single degree of freedom, we change (15) into

$$e_j = C(\text{input}), \qquad (25)$$

Table 1
Properties, advantages and drawback of the variations of the DMP adaptation to the environment.

	Chg. Ref.	Direct	Derived
Property	Velocity-resolved approach (3.1) changes the reference of DMP learning	Error of DMP learning (3.2) is defined with the error of force tracking	Error of DMP learning is defined with the error of force tracking, modified into accelerations of the DMP spring–mass system
Advantage	Classical velocity resolved approach with well known stability properties and behavior	Simple; no additional parts of the algorithm; core of the coaching algorithm.	Completely reduces the error
Drawback	Subject to delay due to the integral part of the force controller	Will not completely cancel out the error due to the delay of integration	Subject to noise of the derivation that modifies the error of force tracking

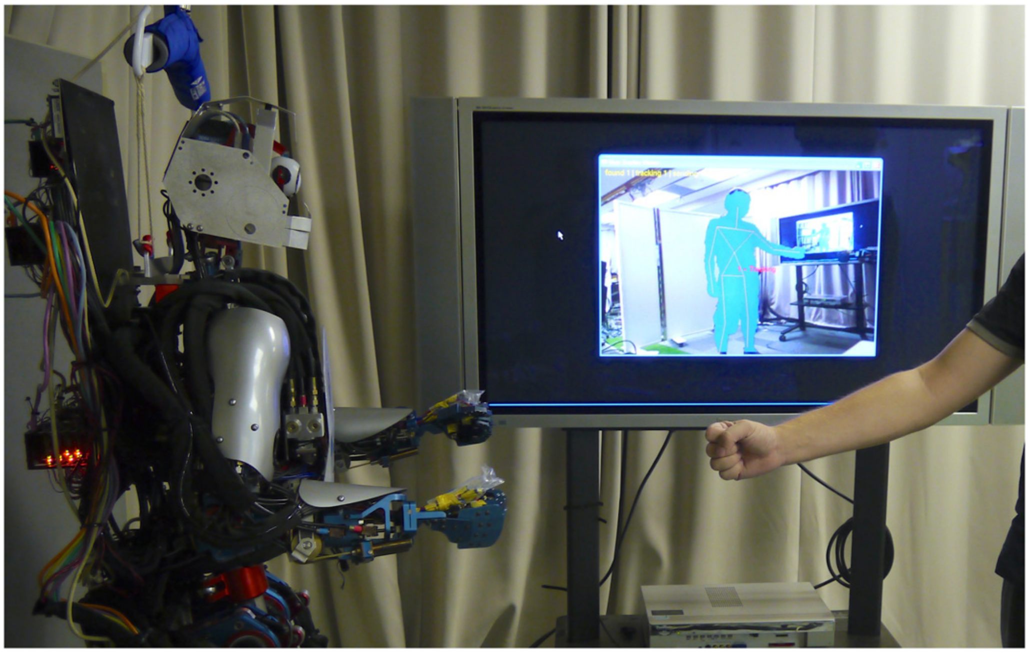


Fig. 1. Experimental setup for coaching of periodic motion on the Sarcos CBi humanoid robot using predefined gestures. A kinect RGB-D camera detected the posture of the human next to the robot. The position and the choice of the arm (left or right) determined the coaching behavior.

making the error a function of the input, where input can be either the force applied to the robot or the demonstrator's pointing gesture, visually illustrating in which direction to change the trajectory. For the case of force input, (25) changes into

$$e_j = k_l F, \tag{26}$$

where parameter k_l scales the measured force F. The measured force in this case should be the force exerted by the coach on the robot. If the robot is in contact with an object, for example when wiping the table, one must distinguish between the forces that arise from the contact with the table and as a result of friction, and the forces applied by the human operator. A simple solution is to decouple forces by direction.

Pointing gestures can be used instead of the force. We used active motion capture markers to first demonstrate a motion and later use the same markers and their relative positions for tutoring. We defined the following repulsive force field

$$e_j(x) = \begin{cases} 0 & p > 0.1 \\ (0.001/p^2 - 0.1)/40 & p \le 0.1,\ p_{1z} > p_{2z} \\ (-0.001/p^2 - 0.1)/40 & \text{otherwise} \end{cases} \tag{27}$$

where p stands for the distance between the robot and the closest marker attached to the coach's hand. Index $_{iz}$ is the z_p axis location of the ith marker. The given force field has no effect on the robot if the closest marker is more than 10 cm away, whereas its effect increases quadratically with proximity, effectively pushing the robot away if $p \approx 0$. The relative location of the markers also defines if the robot is being pushed away or pulled towards the tutor. The given force field was determined empirically.

The design of the force field has a direct impact on the behavior of the robot. Force fields have previously been applied to DMPs for obstacle avoidance [44]. The same field can be used for coaching. In this case we coach the robot through predefined gestures as depicted in Fig. 1.

A 3-DOF DMP is defined by

$$\dot{\mathbf{z}} = \Omega \left(\alpha_z \left(\beta_z (\mathbf{g} - \mathbf{y}) - \mathbf{z}\right) + \boldsymbol{C}_y + \boldsymbol{f}\right), \tag{28}$$

where $\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{g}$, $\boldsymbol{C}_y$, and $\boldsymbol{f}$ are three dimensional values (for positions, additional dimensions can be added for orientations). The definition of the coupling term $\boldsymbol{C}_y$ prescribes the behavior of the robot. We designed the coupling term as a modified obstacle avoidance coupling term $\boldsymbol{C}_y$ from [44], now given by

$$\boldsymbol{C}_y = \gamma\ \mathrm{s}(\|\boldsymbol{o} - \boldsymbol{x}\|)\ \exp(-\beta\phi)\ \boldsymbol{d}. \tag{29}$$

Here $\boldsymbol{x}$ is the Cartesian position of the end-effector, $\boldsymbol{o}$ is the center position of the perturbation potential field (defined by hand position), $\boldsymbol{d}$ is the perturbation direction (defined by the pointing gesture), γ and β are the scaling factors, ϕ is given by

$$\phi = \arccos\left(\frac{(\boldsymbol{o} - \boldsymbol{x})^T \dot{\boldsymbol{x}}}{\|(\boldsymbol{o} - \boldsymbol{x})\|\ \|\dot{\boldsymbol{x}}\|}\right) \tag{30}$$

$\mathrm{s}(r)$ is defined as

$$\mathrm{s}(r) = \frac{1}{1 + e^{\eta(r - r_m)}}, \tag{31}$$

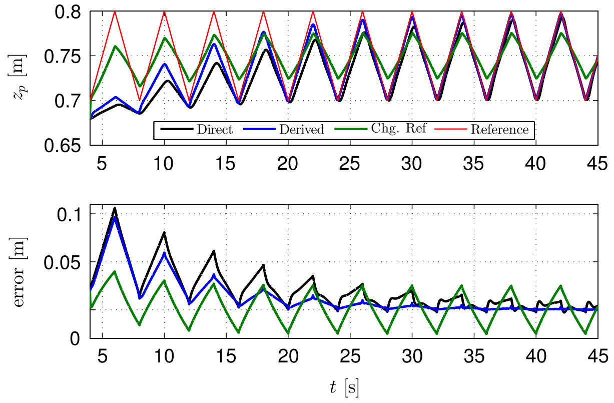


Fig. 2. The results of simulated trajectory adaptation using three different control methods, with a tilted flat surface as a reference. The experiment started with the robot already in contact with the surface. We can see the reference (red) and the three resulting trajectories in the top plot. The errors of adaptation are shown in the bottom plot. See the text for a description of separate lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

where η is the scaling factor and r_m the distance at which the perturbation field should start affecting the robot's motion.

5. Results

In this section we discuss simulated and real-world results of the adaptation to the environment and the coaching.

5.1. Simulated results

We first present the results of a comparison of the three possibilities of adapting to the environment, namely by changing the reference trajectory and the two methods of DMP adaptation—the Direct and the Derived methods.

The simulated experiment was designed to show adaptation of all three methods to a tilted flat surface. In the top plot of Fig. 2 we can see red line depicting the reference, i.e. the table. As it is tilted and the robot is moving left–right, it is a saw-signal. The three output signals of the adaptation are also shown. The green line depicts the trajectory when using the approach of changing the reference for learning, as given by (16)–(18). Note that there is some delay in the adaptation as a consequence of the velocity-resolved force control approach of changing the reference. The bottom plot shows that the error does not completely disappear, but is reduced. The values of $\mathbf{K}_i$, $\mathbf{K}_p$ were determined empirically.

The *Direct* and the *Derived* methods also need some time to adapt but considerably reduce the error in the steady state. We can see that the derived method, given by (24) and depicted in blue, completely cancels out the error, unlike the direct method, which is given by (22) and is depicted in black. This is because the transformation of the error signal is in fact an inverse of the DMP itself and the adaptation is therefore linear at the output, while the direct method uses a second order DMP system that receives linear correction signals. As stated in Table 1, the derived method utilizes first and second order derivatives of the error signal, which could prove extremely noisy.

5.2. Adaptation to environment

This task was performed using a Kuka 7 degree-of-freedom LWR-4 robot, controlled at 500 Hz through Matlab Simulink. The wiping motion was first transferred from a human to a robot using an Optitrack motion capture system with markers on the sponge. The recorded task-space motion was reproduced by the robot while the method of changing the reference, given by (18), ensured that the robot achieved the contact with a surface needed for effective wiping. Fig. 3 shows the 3-D trajectory resulting from an adaptation of the demonstrated motion to a flat, yet slightly tilted surface. The tilting angle was set completely arbitrarily and was not measured.

The force feedback signal for adaptation to a flat, horizontal surface is shown in Fig. 4. The top plot shows the trajectory in the z_p direction. Once the adaptation has started, the robot approaches the table at a finite speed, which was limited beforehand. The bottom plot of Fig. 4 shows the forces. Some oscillations are the

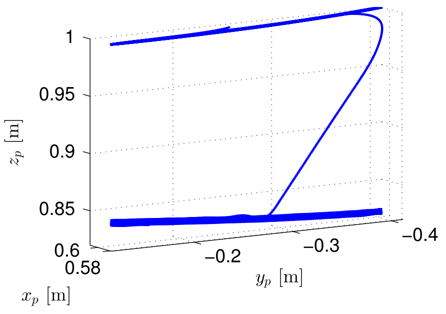


Fig. 3. The complete 3-D trajectory resulting from the adaptation of the demonstrated trajectory in p_z direction using the approach of changing the reference. The force results are depicted in Fig. 5.

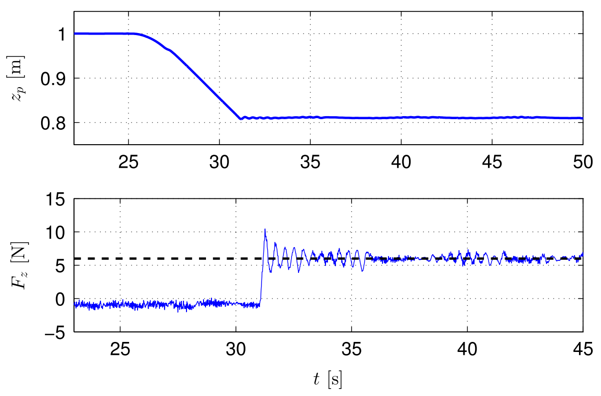


Fig. 4. The top plot shows real world results of adaptation of motion in p_z direction (downwards). The resulting forces with the referential contact force set to 6 N (dashed line) are shown in the bottom plot. As the robot performed left–right wiping motion, some oscillations due to the contact are visible in the force measurement.

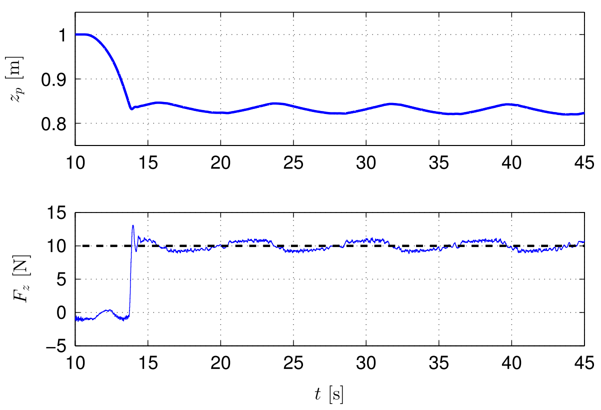


Fig. 5. The trajectory of motion when adapting to a flat but tilted surface in the top plot. The resulting forces show a clear hysteresis resulting from moving up or down the slope in the bottom plot. The oscillations in the force plot are a result of the delay of adaptation, caused by the integral part of the adaptation in (18).

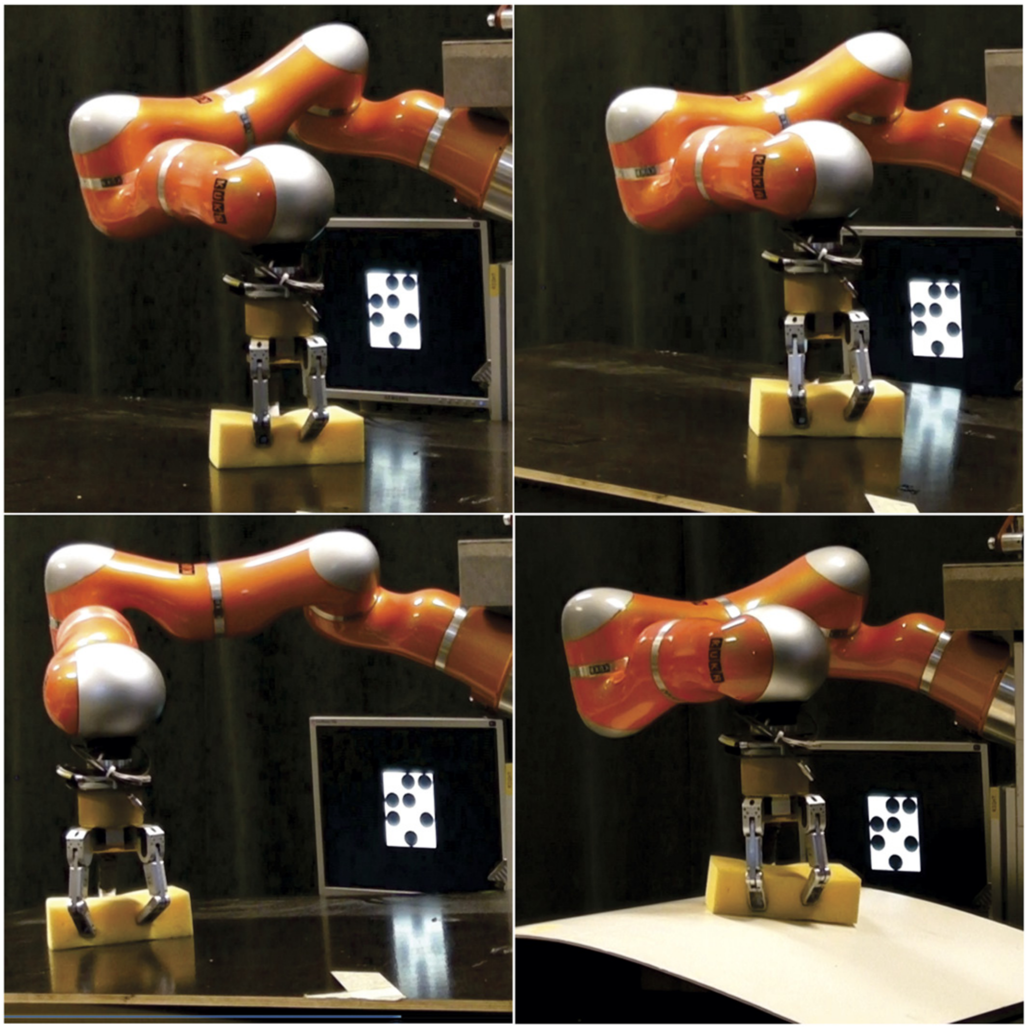


Fig. 6. Kuka LWR-4 7DOF robot wiping differently tilted surfaces and a curved surface. Top left: horizontal surface. Top right: right-tilted surface. Bottom left: left-tilted surface. Bottom right: curved surface. Kuka LWR-4 robot experiments are also depicted in the accompanying video (see Appendix A).

result of friction from dragging the sponge left–right during the wiping.

Fig. 5 shows the results of wiping a slightly tilted surface, with the z_p trajectory in the top plot and the force profile in the bottom plot. Notice the hysteresis of resulting forces, which shows that adaptation takes some time. The integral part of adaptation in (18) introduces delays, which cause these force oscillations. Just as in the simulated environment, the gains $\mathbf{K}_i$, $\mathbf{K}_p$ determine the behavior of the robot.

The real-world wiping experiment with different, arbitrarily tiled flat surfaces and a curved surface is shown in Fig. 6. All experiments on the Kuka LWR-4 robot are depicted in the accompanying video (see Appendix A).

We also implemented the direct method, given by (22). In this scenario the robot was already in contact with the surface and the reference was a sinusoidal force trajectory. Fig. 7 shows the results. A low value of k_l was used in (22) for safety. A higher value would reduce the time needed for adaptation, but a too high value would make the contact unstable. The value used was determined empirically.

5.3. *Adaptation to the environment and coaching*

In this section we show the results of changing only a part of the trajectory using coaching. Results using force interaction are presented first, followed by results based on coaching gestures.

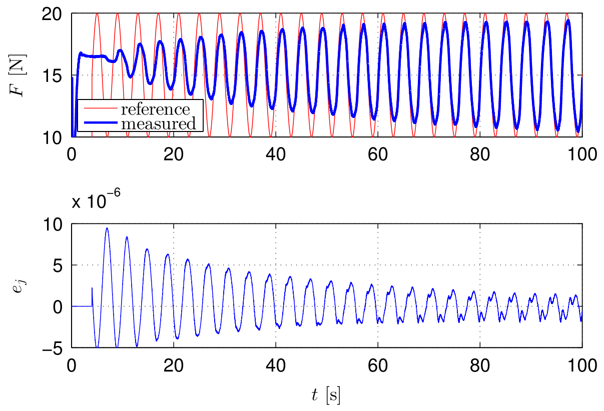


Fig. 7. The results of adapting the robot trajectory using the direct method, with a sinusoidal referential force. The experiment started with the robot already in contact with the surface. The referential and resulting forces are in the top plot, while the error signal, given by (22) is in the bottom plot.

Fig. 8 shows the robot end-effector trajectory before and after coaching. The initial robot wiping motion is in green. The blue line shows the trajectory of the robot during coaching, i.e. while the human was pushing/pulling on it. The measured contact forces are shown in Fig. 9. Four clear peaks of force show where the human

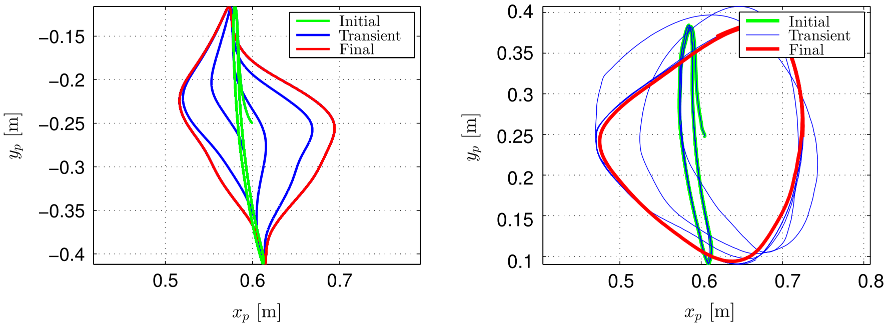


Fig. 8. Left: $X - Y$ plot of the end-effector motion depicts the initial motion in green, the motion during coaching in blue, and the final trajectory in red. Right: the same result that led to a different trajectory. The approach of (32) was used. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

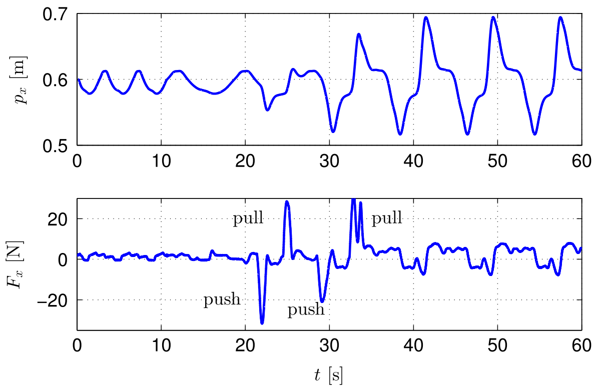


Fig. 9. The top plot depicts the trajectory of motion in x_p direction of the end effector of the robot. The external forces applied by the tutor, depicted in the bottom plot, modified the motion to achieve the intended result. The proposed method provides smooth transition from non-coaching to coaching behavior.

pushed/pulled on the robot. The final wiping motion of the robot after coaching is shown in red. The initial motion was performed using a previously learned DMP, the one from Fig. 3. The robot found and maintained a contact with a flat surface from the start of the experiment. Coaching was applied in x direction only.

Instead of forces as the error of learning as defined in (26), we can also use the position of the robot. By using impedance control mode for the robot and setting a lower stiffness in the direction we want to coach the robot, for example x_p, it will move compliantly in that direction if pushed/pulled. We can now use the difference between the desired and the actual position as the error signal for learning,

$$e_j = x_{p,\text{des}} - x_{p,\text{act}}. \tag{32}$$

The results are shown in the right plot of Fig. 8. It should be noted that when the robot is compliant, contact forces with the surface might affect its trajectory. While this might be solved by changing the stiffness values during coaching and during pure motion reproduction, in our experiment we kept the stiffness constant. Modifying the stiffness is simply a matter of the interface.

The difference of plots in Fig. 8 comes from the compliance of the robots. If the feedback term $d(F)$ in (23) was set higher, the robot would give way much more, and the same principle as in (32) could be applied. We observed that coaching became much more intuitive when the robot was compliant.

When using gestures, we used pointing gestures to coach the robot as defined in (28). We implemented this form of coaching on the JST-ICORP/SARCOS humanoid robot CBi [45]. We used the Microsoft Kinect sensor and the associated body tracker to capture human coaching gestures. Fig. 1 shows the experimental setup, where the body tracking results can be seen on the display in the background.

To make coaching intuitive, the interface was set so that the human coach can modify the trajectory by either pushing it away from him using his right hand or pulling it towards him with his left hand. The coaching direction was calculated using the wrist and the elbow location. For the right hand, i.e. pushing the trajectory away from the coach, the direction is given by

$$\boldsymbol{d}_R = \frac{\boldsymbol{x}_{w,R} - \boldsymbol{x}_{e,R}}{\|\boldsymbol{x}_{w,R} - \boldsymbol{x}_{e,R}\|}, \tag{33}$$

where the $\boldsymbol{x}_{w,R}$ and the $\boldsymbol{x}_{e,R}$ are the Cartesian positions of the right hand wrist and the right hand elbow in the robot's base coordinate system. For pulling the trajectory, the direction is given by

$$\boldsymbol{d}_L = -\frac{\boldsymbol{x}_{w,L} - \boldsymbol{x}_{e,L}}{\|\boldsymbol{x}_{w,L} - \boldsymbol{x}_{e,L}\|}. \tag{34}$$

Here $\boldsymbol{x}_{w,L}$ and the $\boldsymbol{x}_{e,L}$ are respectively the Cartesian positions of the left hand wrist and the left hand elbow in the robot's base coordinate system.

The center of the potential field generated by each hand was moved slightly away from the respective hand. For the right hand, the origin of the potential field defined by the coaching gesture was moved in the direction of the coaching gesture

$$\boldsymbol{o}_R = \boldsymbol{x}_R + \xi_R \boldsymbol{d}_R, \tag{35}$$

where ξ_R is the scalar that defines the distance between the hand and the center of the coaching point in the direction of $\boldsymbol{d}_R$. Similar equation is used also for the left hand which attracts the trajectory towards the hand.

$$\boldsymbol{o}_L = \boldsymbol{x}_L - \xi_L \boldsymbol{d}_L. \tag{36}$$

Here, the effective coaching point is moved in the opposite direction of perturbation $\boldsymbol{d}_L$. With such modifications the effective origins of potential fields are always in front of the human hands in the direction of pointing at the distance defined by ξ_R and ξ_L.

To determine which hand is active, we use the distance between both wrist positions $\boldsymbol{x}_{w,L}, \boldsymbol{x}_{w,R}$ and the robot's end-effector position x_p. The active hand is the one which is closer to the robot's hand position.

To show the applicability of the interface for online modification of the initial rhythmic movement using human in the loop coaching gestures, we first provide an example of pulling-in the task space trajectory. The parameters were set to $\gamma = 10$, $\eta = 10$, $r_m = 0.15$ and $\beta = -10/\pi$.

The adaptation is not limited to task space. To update the trajectories in joint-space when they are perturbed in task space, with the coupling term denoted by $\boldsymbol{C}_y$, a pseudo inverse of the task Jacobian is used. This essentially maps the task space velocities into the joint space velocities with $\dot{\boldsymbol{q}} = \mathbf{J}^\dagger \dot{\boldsymbol{x}}$. By applying a similar transformation to $\boldsymbol{C}_y$ we obtain

$$\boldsymbol{C}_q = \mathbf{J}^\dagger \boldsymbol{C}_y \tag{37}$$

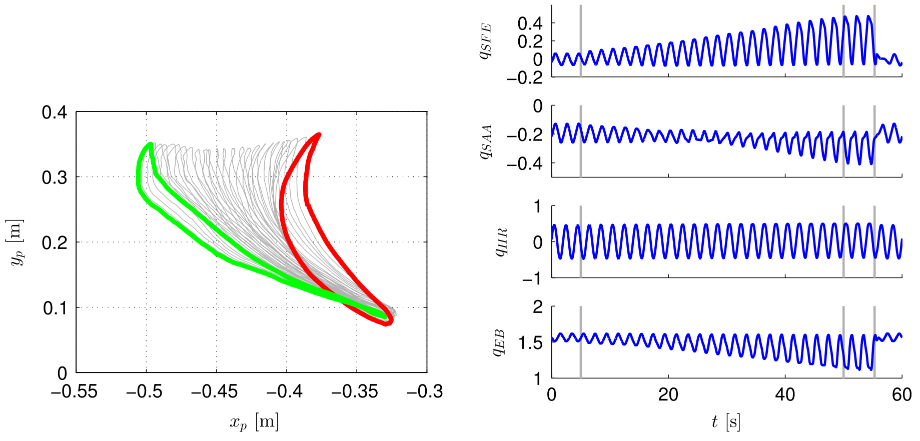


Fig. 10. Left: Task space motion of the robot's end-effector, where human coach was modifying the motion pattern. The initial trajectory is in red and the final trajectory is in green. The time evolution of the trajectory modification is indicated with gray line. Right: Joint space motion in time of the robot's right hand, while coaching. Vertical lines indicate the important events described in text. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

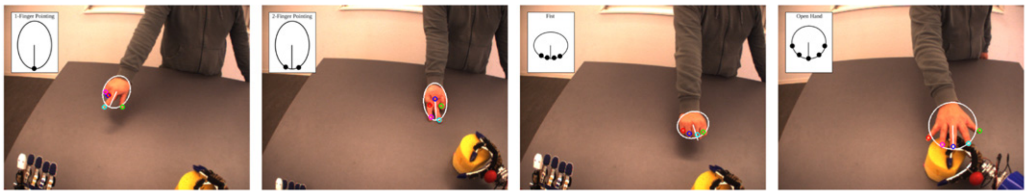


Fig. 11. The four hand gesture for the coaching: 1-finger pointing, 2-finger pointing, a fist, and an open hand.

where $\boldsymbol{C}_q = [C_{q,1}\ C_{q,2}\ \dots\ C_{q,j}]^T$ and j is the number of the robot's degrees of freedom. The components of (37) are now used for updating the DMP weights w_i using (14) and (13). In this way we ensure that the joint space trajectories encoded by the DMPs are properly modified according to the coach's instructions.

Keeping the movement representation in the joint space is beneficial because our initial movement trajectories, which are encoded by DMPs, are usually acquired by kinesthetic guiding. By using joint space trajectories we avoid losing information about the selected robot configuration during human guiding on a redundant robot.

Fig. 10-left shows the task space motion of the robot's end-effector in the (x_p, y_p) plane. We can see a successful modification of the motion based on the human coaching gestures. In Fig. 10-right we show the corresponding joint space trajectories as a function of time. The teaching of the new motion pattern begins after 5 s, indicated with the first vertical line. The joint space trajectory was modified successfully to achieve the desired task space motion. In Fig. 10-right we can see that at approximately 50 s the human coach stopped modifying the behavior and at approximately 55 s the new motion pattern was switched back to the original motion pattern. At this point the difference between original motion trajectory and the modified motion trajectory is even more evident.

5.4. Human–robot interface expansions

In this section we demonstrate the features of the complete system using an advanced human–robot interface. The system allows the initial transfer of motion, the adaptation to the environment and coaching based on predefined gestures and force interaction. It has been implemented on the humanoid robot ARMAR-IIIa and is used to train periodic DMPs for a wiping task in an online manner. Initially, a DMP is learned from a human wiping movement which is demonstrated in a predefined work space.

Given the color of the wiping tool, the robot tracks the movements of the tool using the stereo camera system of its active head. For the subsequent force-based adaptation of the learned DMP, we rely on the readings of the force torque sensor installed at the humanoid's wrist.

Using the implemented human–robot interface, the human coach can change the learned and adapted DMP using hand gestures. For the recognition of human hand gestures, the robot visually observes the predefined work space in order to localize and track the fingertips of the coaching human hand. To do so, a fingertip tracking algorithm is used which has been introduced in [46]. The fingertips are described with regard to the principal axes spanned by an ellipsoid which circumscribes the entire hand area. Based on these positions a feature vector for the representation of hand gestures was derived. To recognize a gesture, the feature vector is compared with labeled examples which represent a certain gesture. For each coaching mode we defined a distinctive gesture: 1-finger pointing, 2-finger pointing, a fist, and an open hand gesture. The different gestures are depicted in Fig. 11. The 1-finger pointing gesture generates a new target position which is used to change the goal g of the DMP and thus the center of the wiping motion. The results of changing the center of the wiping motion are shown in Fig. 12.

In order to change the periodic pattern of the wiping motion, the two-finger pointing gesture is used to pull the movements of the robot towards the coaching hand. In contrast, a repelling behavior is triggered using the fist which pushes the robots end-effector away from the human hand. The pushing and pulling behaviors are generated by a virtual potential field imposed on the position of the human hand and, thus, creating virtual forces for the coaching of the DMP. An open hand denotes the approach movement of the coaching hand and invokes a reduction of the frequency with which the wiping motion is reproduced. This facilitates the coaching for the human and allows a smooth transition from vision-based to force-based coaching. The system switches to force-based coaching once an external force is applied on the robots wrist. Using the force-torque sensor, we can coach and further adapt the DMP. During the force-based coaching, the active head shifts its view towards the end-effector. The systems

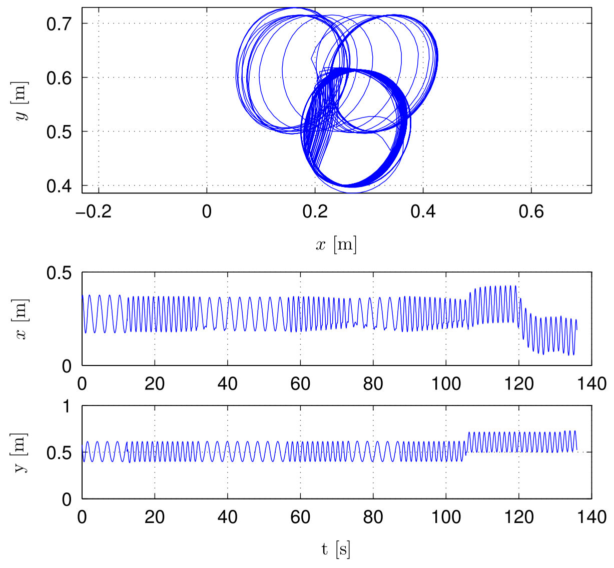


Fig. 12. $(x_p - y_p)$ plane trajectories of wiping and coaching on the ARMAR-IIIa robot in the top plot. We can see that the center of the circular trajectory was changed through the coaching interface. Separate directions of motion are depicted in the lower plots. Reduction of the frequency during wiping can be observed in the bottom plots.

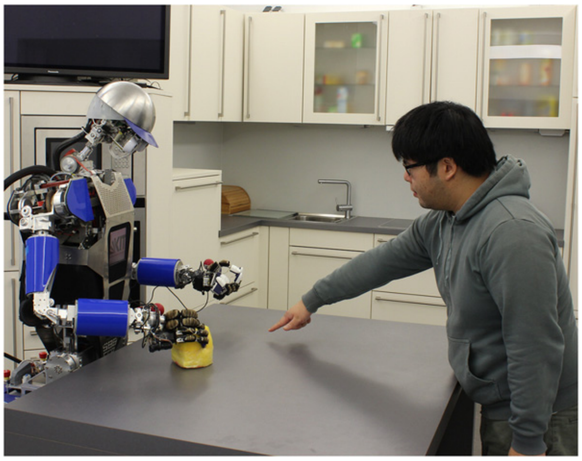


Fig. 13. Coaching of the ARMAR-IIIa humanoid robot through the use of hand gestures.

returns to the vision-based coaching once the human hand leaves the currently observed work space. Fig. 13 depicts coaching of the ARMAR-IIIa humanoid robot with predefined gestures.

6. Discussion and conclusion

The main advantage of learning the motion required for sustaining a contact is that it allows the combination of feedback and feed-forward control loops. While this by itself is nothing new, the novelty stems from the fact that the feed-forward component is autonomously *learned* and encoded in a dynamic movement primitive. By using the feed-forward component, the feedback component is greatly reduced if not completely canceled, making the behavior exactly as desired. By exploiting the DMP learning mechanisms, we remain in the framework which allows easy modulation with only changing a small set of parameters. Mitigating the need to create models beforehand, as they are learned through exploration and coaching, allows non-experts to effectively transfer motion to the robot by demonstrating everyday tasks.

In comparison to other approaches, several sub-areas of research need to be considered. Force control has been applied in robotics in many different contexts. The benefit of our method is that it allows easy and intuitive transfer of motion from a human to the robot. This transferred motion adapts to the conditions of the task—for example, that it needs to maintain contact with the environment. As stated in the introduction, various techniques exist for that, but the methods presented in this paper extend this feature to a well developed and extensively applied framework. The approach can be used on position controlled robots, or on torque controlled robots, exploiting the properties of different control methods, such as impedance control as depicted in Fig. 8. On the other hand, it also allows the use of position controlled robots, such as the ARMAR-IIIa, with the only difference in behavior due to lower bandwidths.

For learning, the exploration of the trajectory space uses the learning algorithm of the DMPs, which is computationally light and allows for quick adaptation. In the case of periodic motions, this can be in the rank of a few periods [40]. The method of direct DMP adaptation, which was also applied in the context of coaching, observes similar principles as iterative learning control, and could be considered an instance of it. It enables direct learning of the weights of DMP kernel functions instead of the signal. Again, remaining in the DMP framework has beneficial properties for robot control.

For motion adaptation by coaching, our approach thus retains the beneficial properties of DMPs with time-invariance and the means of modulation, but additionally enables the modulation of complex motions through intuitive coaching gestures. An advanced but intuitive coaching interface, which was demonstrated on the ARMAR-IIIa robot, has proven to be a viable solution.

Adaptation to the environment, as presented in this paper, exploits the knowledge of the demonstrator to determine the needed references and directions for adaptation. An open research issue remains, how such adaptations can be performed autonomously. While a complex cognitive reasoning system behind this is beyond the scope of this paper, simple conditions could greatly improve the autonomy of adaptation. For example, in the context of wiping, we could direct the robot to increase the force of contact with the surface in case the wiping does not actually remove the identified dirt. Vision systems, specifically using RGB-D sensors, are also efficient at detecting surfaces. These surfaces could be the targets of wiping when a wiping command is issued. Any such augmentation of the interface can greatly improve the user experience.

In the paper we proposed and evaluated several methods for adaptation of DMPs based on force feedback. We have shown that all can be effectively used for acquiring and maintaining non-rigid contacts with the environment. They thus offer a viable solution for an inclusion in future household assistants. In the future we will combine the method of DMP adaptation, effectively applied to coaching, to modify feed-forward models of complex tasks. For example, one might update the demonstrated DMP so that the postural stability of the robot is observed. Another possible application of our approach is the learning of the required torque signals for robot control.

Acknowledgments

This work was supported by the European Communitys Seventh Framework Programme FP7/2007–2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) grant agreement no. 270273, Xperience. Additionally, A.U. was supported by NICT, Japan Trust International Research Cooperation Program; J.M. was supported by SRPBS from MEXT/AMED;

by MEXT KAKENHI 23120004; by MIC-SCOPE; by JST-SICP; by ImPACT Program of Council for Science, Technology and Innovation.

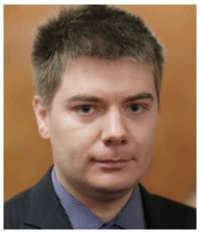
Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.robot.2015.09.011.

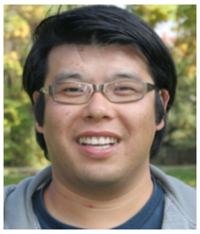
References

[1] R. Dillmann, Teaching and learning of robot tasks via observation of human performance, Robot. Auton. Syst. 47 (2–3) (2004) 109–116.
[2] A. Ude, A. Gams, T. Asfour, J. Morimoto, Task-specific generalization of discrete and periodic dynamic movement primitives, IEEE Trans. Robot. 26 (5) (2010) 800–815.
[3] A. Ude, C.G. Atkeson, M. Riley, Programming full-body movements for humanoid robots by observation, Robot. Auton. Syst. 47 (2–3) (2004) 93–108.
[4] Y. Wada, M. Kawato, A via-point time optimization algorithm for complex sequential trajectory formation, Neural Netw. 17 (3) (2004) 353–364.
[5] S. Calinon, F. D'halluin, E.L. Sauser, D.G. Caldwell, A.G. Billard, Learning and reproduction of gestures by imitation: An approach based on hidden Markov model and Gaussian mixture regression, IEEE Robot. Autom. Mag. 17 (2) (2010) 44–54.
[6] S. Khansari-Zadeh, A. Billard, Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Taipei, Taiwan, 2010, pp. 2676–2683.
[7] E. Gribovskaya, S. Khansari-Zadeh, A. Billard, Learning non-linear multivariate dynamics of motion in robotic manipulators, Int. J. Robot. Res. 30 (1) (2011) 80–117.
[8] T. Inamura, I. Toshima, H. Tanie, Y. Nakamura, Embodied symbol emergence based on mimesis theory, Int. J. Robot. Res. 23 (4–5) (2004) 363–377.
[9] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, S. Schaal, Dynamical movement primitives: Learning attractor models for motor behaviors, Neural Comput. 25 (2) (2013) 328–373.
[10] T. Matsubara, S.-H. Hyon, J. Morimoto, Learning parametric dynamic movement primitives from multiple demonstrations, Neural Netw. 24 (5) (2011) 493–500.
[11] T. Kulvicius, K. Ning, M. Tamosiunaite, F. Wörgötter, Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting, IEEE Trans. Robot. 28 (1) (2012) 145–157.
[12] B. Nemec, A. Ude, Action sequencing using dynamic movement primitives, Robotica 30 (5) (2012) 837–846.
[13] J. Peters, S. Schaal, Reinforcement learning of motor skills with policy gradients, Neural Netw. 21 (2008) 682–697.
[14] F. Stulp, E.A. Theodorou, S. Schaal, Reinforcement learning with sequences of motion primitives for robust manipulation, IEEE Trans. Robot. 28 (6) (2012) 1360–1370.
[15] J. Kober, A. Wilhelm, E. Oztop, J. Peters, Reinforcement learning to adjust parametrized motor primitives to new situations, Auton. Robots 33 (2012) 361–379.
[16] M. Tamosiunaite, B. Nemec, A. Ude, F. Wörgötter, Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives, Robot. Auton. Syst. 59 (11) (2011) 910–922.
[17] L. Villani, J. De Schutter, Force control, in: B. Siciliano, O. Khatib (Eds.), Handbook of Robotics, Springer, 2008, pp. 161–185.
[18] N. Hogan, Impedance control: An approach to manipulation I II III, J. Dyn. Syst. Meas. Control 107 (1985) 1–24.
[19] V. Koropouli, D. Lee, S. Hirche, Learning interaction control policies by demonstration, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2011, pp. 344–349.
[20] P. Pastor, L. Righetti, M. Kalakrishnan, S. Schaal, Online movement adaptation based on previous sensor experiences, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, San Francisco, CA, 2011, pp. 365–371.
[21] P. Pastor, M. Kalakrishnan, L. Righetti, S. Schaal, Towards associative skill memories, in: IEEE-RAS International Conference on Humanoid Robots, Humanoids, 2012, pp. 309–315.
[22] M. Kalakrishnan, L. Righetti, P. Pastor, S. Schaal, Learning force control policies for compliant manipulation, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2011, pp. 4639–4644.
[23] T. Kulvicius, M. Biehl, M.J. Aein, M. Tamosiunaite, F. Wörgötter, Interaction learning for dynamic movement primitives used in cooperative robotic tasks, Robot. Auton. Syst. 61 (12) (2013) 1450–1459.
[24] A. Gams, B. Nemec, A. Ijspeert, A. Ude, Coupling movement primitives: Interaction with the environment and bimanual tasks, IEEE Trans. Robot. 30 (4) (2014) 816–830.
[25] E.L. Sauser, B. Argall, G. Metta, A. Billard, Iterative learning of grasp adaptation through human corrections, Robot. Auton. Syst. 60 (1) (2012) 55–71.
[26] S. Calinon, A. Billard, Incremental learning of gestures by imitation in a humanoid robot, in: Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction, HRI, 2007, pp. 255–262.
[27] F.J. Abu-Dakka, B. Nemec, J.A. Jørgensen, T.R. Savarimuthu, N. Krüger, A. Ude, Adaptation of manipulation skills in physical contact with the environment to reference force profiles, Auton. Robots 39 (2) (2015) 199–217.
[28] L. Rozo, P. Jimnez, C. Torras, A robot learning from demonstration framework to perform force-based manipulation tasks, Intell. Serv. Robot. 6 (1) (2013) 33–51.
[29] L. Rozo, P. Jimenez, C. Torras, Force-based robot learning of pouring skills using parametric hidden Markov models, in: Workshop on Robot Motion and Control, RoMoCo, 2013, pp. 227–232.
[30] S. Calinon, Z. Li, T. Alizadeh, N.G. Tsagarakis, D.G. Caldwell, Statistical dynamical systems for skills acquisition in humanoids, in: IEEE-RAS International Conference on Humanoid Robots, Humanoids, 2012, pp. 323–329.
[31] A. Gams, M. Do, A. Ude, T. Asfour, R. Dillmann, On-line periodic movement and force-profile learning for adaptation to new surfaces, in: IEEE-RAS International Conference on Humanoid Robots, Humanoids, Nashville, TN, 2010, pp. 560–565.
[32] A. Gams, T. Petrič, B. Nemec, A. Ude, Learning and adaptation of periodic motion primitives based on force feedback and human coaching interaction, in: IEEE-RAS International Conference on Humanoid Robots, 2014, pp. 166–171.
[33] J. Ernesti, L. Righetti, M. Do, T. Asfour, S. Schaal, Encoding of periodic and their transient motions by a single dynamic movement primitive, in: IEEE-RAS International Conference on Humanoid Robots, Humanoids, 2012, pp. 57–64.
[34] M. Do, J. Schill, J. Ernesti, T. Asfour, Learn to wipe: A case study of structural bootstrapping from sensorimotor experience, in: IEEE International Conference on Robotics and Automation, ICRA, 2014, pp. 1858–1864.
[35] v. Ortenzi, M. Adjigble, J.A. Kuo, R. Stolkin, M. Mistry, An experimental study of robot control during environmental contacts based on projected operational space dynamics, in: IEEE-RAS International Conference on Humanoid Robots, 2014, pp. 407–412.
[36] A. Gruebler, V. Berenz, K. Suzuki, Coaching robot behavior using continuous physiological affective feedback, in: IEEE-RAS International Conference on Humanoid Robots, Humanoids, 2011, pp. 466–471.
[37] M. Riley, A. Ude, C. Atkeson, G. Cheng, Coaching: An approach to efficiently and intuitively create humanoid robot behaviors, in: IEEE-RAS International Conference on Humanoid Robots, 2006, pp. 567–574.
[38] D. Lee, C. Ott, Incremental kinesthetic teaching of motion primitives using the motion refinement tube, Auton. Robots 31 (2–3) (2011) 115–131. http://dx.doi.org/10.1007/s10514-011-9234-3.
[39] T. Petrič, A. Gams, L. Žlajpah, A. Ude, J. Morimoto, Online approach for altering robot behaviors based on human in the loop coaching gestures, in: IEEE International Conference on Robotics and Automation, ICRA, 2014, pp. 4770–4776.
[40] A. Gams, A.J. Ijspeert, S. Schaal, J. Lenarčič, On-line learning and modulation of periodic movements with nonlinear dynamical systems, Auton. Robots 27 (1) (2009) 3–23.
[41] T. Petrič, A. Gams, A.J. Ijspeert, L. Žlajpah, On-line frequency adaptation and movement imitation for rhythmic robotic tasks, Int. J. Robot. Res. 30 (14) (2011) 1775–1788.
[42] C.G. Atkeson, A.W. Moore, S. Schaal, Locally weighted learning, AI Rev. 11 (1997) 11–73.
[43] L. Ljung, T. Söderström, Theory and Practice of Recursive Identification, MIT Press, 1986.
[44] H. Hoffmann, P. Pastor, D.-H. Park, S. Schaal, Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance, in: IEEE International Conference on Robotics and Automation, ICRA, Kobe, Japan, 2009, pp. 2587–2592.
[45] G. Cheng, S.-H. Hyon, J. Morimoto, A. Ude, J.G. Hale, G. Colvin, W. Scroggin, S.C. Jacobsen, CB: A humanoid research platform for exploring neuroscience, Adv. Robot. 21 (10) (2007) 1097–1114.
[46] M. Do, T. Asfour, R. Dillman, Particle filter-based fingertip tracking with circular hough transform features, in: Proceedings of the 12th IAPR Conference on Machine Vision Application, 2011, pp. 1–4.

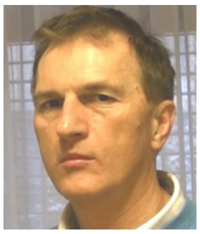
Andrej Gams received the Diploma degree in Electrical Engineering in 2004, and the Ph.D. degree in Robotics from the University of Ljubljana, Slovenia, in 2009. He is currently a Research Fellow with the Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana. He was a Postdoctoral Researcher with the Biorobotics Laboratory, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, during 2012–2013. He was a visiting researcher at the ATR Computational Neuroscience Laboratories in the summers of 2009 and 2014. His research interests include imitation learning, control of periodic tasks, and humanoid cognition. He received the Best Paper Research Award at the 19th International Workshop in the Alpe-Adria-Danube region in 2010 and the Best Scientific Paper Award at the 23rd International Conference on Robotics in Alpe-Adria-Danube Regions in 2014. He is also the recipient of the Jožef Stefan Golden Emblem Award for his Ph.D. dissertation in 2012. He received the SCIEX NMS-CH fellowship for postdoctoral studies at EPFL, Switzerland, in 2012. He has been a program committee member of several conferences, including Humanoids 2011.



Tadej Petrič attended the University of Maribor, Slovenia, where he obtained an M.Sc. in Electrical Engineering in 2008. His M.Sc. covered modeling and robotic control of underactuated dynamic system. For his work, he received the Prof. Dr. Vratislav Bedjanič award in 2008. In 2013 he received a D.Sc. in Robotics from the Faculty of Electrical Engineering at the University of Ljubljana. He performed a part of his doctoral research at the Department of Robotic Systems for Dynamic Control of Legged Humanoid Robots at the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany. In 2013, he was a visiting researcher at ATR Computational Neuroscience Laboratories in Japan. He is currently a Postdoctoral Fellow working with Prof. Auke Ijspeert in the Biorobotics Laboratory at the Swiss Federal Institute of Technology (EPFL) in Lausanne, Switzerland. His current research is concerned with the design of biologically plausible robot controllers that achieve robustness and adaptation to changing environments comparable to that found in humans.



Martin Do received his Diploma and Ph.D. from Karlsruhe Institute of Technology (KIT), Germany, in 2007 and 2014, respectively. Currently, he is a Postdoctoral Researcher with the High Performance Humanoid Technologies Laboratories, Institute of Anthropomatics and Robotics, KIT. His research interests include methods and models for the learning of grasping and manipulation actions from human observation and strategies for the adaptation and augmentation of robot skills through exploration and predictive reasoning.



Bojan Nemec received his diploma degree in Electrical Engineering in 1979, M.Sc. degree in 1982 and his Ph.D. degree in Robotics from the University of Ljubljana, Slovenia, in 1988.

He is currently a senior research associate at the Dept. of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute. He spent his sabbatical leave at the Institute for Real-Time Computer Systems and Robotics, University of Karlsruhe in 1993. His research interests include robot control, robot learning, service robotics and sports biomechanics.

He received NTF award for best paper in category of Entertainment Robots and Systems at IROS 2009. He has been a program committee member of several conferences and general chair of the Workshop in Alpe-Adria-Danube region in 2013.

Jun Morimoto received the Ph.D. degree in Information Science from the Nara Institute of Science and Technology, Nara, Japan, in 2001.

From 2001 to 2002, he was a Postdoctoral Fellow at Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. Since 2002, he has been at the Advanced Telecommunications Research Institute International, Kyoto, Japan, where he was a Researcher in the Computational Brain Project, the International Cooperative Research Project, Japan Science and Technology Agency from 2004 to 2009. He is currently the Head of the Department of the Brain Robot Interface, Computational Neuroscience Laboratories, Kyoto, Japan.

Tamim Asfour is Professor at the Institute for Anthropomatics and Robotics at the Karlsruhe Institute of Technology (KIT). He is Chair of Humanoid Robotics Systems and Head of the High Performance Humanoid Technologies Lab (H^2T). His current research interest is high performance 24/7 humanoid robotics. Specifically, his research focuses on engineering humanoid robotics systems integrating the mechano-informatics of such systems with the capabilities of predicting, acting and learning from human demonstration and sensorimotor experience. He is developer of the ARMAR humanoid robot family and is leader of the Humanoid Research Group at KIT (2003–now).

He is Editor-in-Chief of the IEEE-RAS conference on Humanoid Robots, is European Chair of the IEEE RAS Technical Committee on Humanoid Robots (2010–2014), Associate Editor of Transactions on Robotics (2010–2014). He is a member of the Executive Board of the German Association of Robotics (DGR) and member of the Board of Directors of euRobotics (2013–2015).

He is principle investigator in several national projects (SFB 588, Autonomous Learning, Invasive Computing) and Integrated European Cognitive Systems projects (PACO-PLUS, Xperience, GRASP, WALK-MAN, KoroiBot, SecondHands, TimeStorm, I-Support).

Aleš Ude received the Diploma degree in Applied Mathematics from the University of Ljubljana, Slovenia, and the Ph.D. degree from the Faculty of Informatics, University of Karlsruhe, Germany. He was awarded the Science and Technology Agency fellowship for postdoctoral studies in ERATO Kawato Dynamic Brain Project, Japan. He is currently the head of Department of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana. He is also associated with the ATR Computational Neuroscience Laboratories, Kyoto, Japan. His research interests include autonomous robot learning, imitation learning, humanoid robot vision, humanoid cognition, and humanoid robotics in general.

2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)
November 3 - 5, 2015, Seoul, Korea

Bio-inspired Learning and Database Expansion of Compliant Movement Primitives

Tadej Petrič[1,2], Luca Colasanto[1], Andrej Gams[2], Aleš Ude[2] and Auke J. Ijspeert[1]

Abstract— **The paper addresses the problem of learning torque primitives – the torques associated to a kinematic trajectory, and required in order to accurately track this kinematic trajectory. Learning torque primitives, which can be interpreted as internal dynamic models, is crucial to achieve at the same time 1) high tracking accuracy and 2) compliant behaviour. The latter improves the safety concerns of working in unstructured environments or with humans. In the proposed approach, first learning by demonstration is used to obtain the kinematic trajectories, which are encoded in the form of Dynamic Movement Primitives (DMPs). These are combined with the corresponding task-specific Torque Primitives (TPs), and together they form new task-related compliant movements, denoted as Compliant Movement Primitives (CMPs). Unlike the DMPs, the TPs cannot be directly acquired from user demonstrations. Inspired by the human sensorimotor learning ability, we propose a novel method which can autonomously learn task-specific Torque Primitives (TPs) associated to given kinematic trajectories in the form of DMPs. The proposed algorithm is completely autonomous, and can be used to rapidly generate and expand the database of CMPs motions. Since the CMPs are parameterized, statistical generalisation can be used to obtain an initial TP estimate of a new CMP motion. Thereby, the learning rate of new CMPs can be significantly improved. The evaluation of the proposed approach on a humanoid robot CoMan performing reaching task shows fast TP acquisition and accurate generalization estimates in real-world scenarios.**

I. INTRODUCTION

One of the key skills that a humanoid robot should posses is the ability to learn new motor behaviours based on human demonstration [1]. The most common way of learning new behaviours is programming by demonstration (PbD) [2], which can be done by using different sensory systems, e.g., visual [3] or kinematics guidance [4]. The key advantage of PbD in joint space is that the robot kinematics is already adapted to the task and the posture is preserved even when redundant robots are used. Different methods have been proposed for PbD. Dynamic Movement Primitives (DMPs) [5] are generally used for learning kinematic trajectories. To execute the desired DMP trajectory accurately, an underlying robot controller that guarantees accurate tracking is usually employed. For example an impedance controller with high gains in the feedback loop as in [6]. However, using high gains makes robots inherently unsafe for interaction with the environment or humans, due to the high interaction forces that may occur during unforeseen contacts [7]. Moreover, in the case of humanoid robots this might also lead to an unsuspected fall.

Different approaches can be used to minimise the interaction forces and at the same time assure accurate trajectory tracking. For example, combining high gain impedance control for accuracy together with proximity sensors or an artificial skin for detecting contacts [8]; by using bi-articular mechanical structures based on artificial pneumatic muscles [9] or by accurate inverse dynamic models for control algorithms with active compliance [10]. However, changing mechanical structures or adding artificial skin to the system will increase its overall price. On the other hand, it is impossible to obtain an accurate generic dynamical model, even for a simple task like table wiping, due to the unknown parameters – for example the friction between the sponge and the table.

To overcome the problem of using impedance control by using dynamical models we propose a novel method which can learn missing dynamic parameters of a given task, and encode them as task-specific torque primitives. Hence, there is no need for modelling the task dynamics. Moreover, if a dynamical model is available, our method compensates for uncertainties with the learned torque primitives. By learning the task-specific torque primitives, the control of the robot allows 1) accurate trajectory tracking and 2) compliant behavior, which is the result of using a low gain feedback loop in an underlying impedance controller. While accurate tracking is required for proper task execution, compliant behaviour is essential for ensuring safe interaction with the environment or, most importantly, humans [7].

To enable both accurate trajectory tracking and compliant behaviour, the DMP [11]–[14] framework needs to be extended toward torque-controlled robots. Inspired by the human sensory motor ability [15]–[18], where hand kinematics are learned from errors in extent and direction in an extrinsic coordinate system, and dynamics are learned from proprioceptive errors in an intrinsic coordinate system, we propose an extension by augmenting the DMP framework with Torque-Primitives (TP). The combination of a DMP and a TP forms a *Compliant Movement Primitive* (CMP) and represents a new model-free control approach, while keeping modulation and parametrization properties of the position-based DMPs.

The first step of gaining the CMPs is to learn the desired motion trajectory in DMPs. Different approaches were established in the past, allowing to learn motions (off-line or on-

Research leading to these results was supported by Sciex-NMSCH project 14.069, European FP7 ICT project Xperience (no. 270273) and project WALK-MAN (No. 611832).

[1] Biorobotics Laboratory, École Polytechnique Fédérale de Lausanne (EPFL), Station 14, CH-1015 Lausanne, Switzerland tadej.petric@epfl.ch

[2]Dept. for Automatics, Biocybernetics and Robotics, Jožef Stean Institute (JSI), Jamova cesta 39, 1000 Ljubljana, Slovenia

line) by using for example kinesthetic or haptic guidance [4], [19]. Once the desired motion is obtained, the corresponding TPs need to be acquired. Our contribution is achieving this goal by using a low impedance control loop combined with a recursive regression method that uses the difference between the desired and the actual movement to update the TPs. Hence the acquisition of the TPs is autonomous. In each subsequent time step, and through a few iterations, the TPs are learned. TPs are employed as feedforward terms, essentially representing new learned task-specific dynamics and allowing the robot to accurately and compliantly execute the desired motion. The proposed approach is similar to one observed in humans [17], where the kinematic trajectory is learned in Cartesian space (DMPs) and the task dynamics in the joint space (TPs).

While the proposed approach eliminates the need for dynamical modeling, the CMPs still have to learn TPs for each task variation. However, since the CMPs are structured, they can be added into a database and statistical generalization (as in [20]) can be used to generate new instances to previously unexplored regions within the database. This also significantly improves the rate of learning, because the initial TPs for a new motion is the outcome of the generalization and hence potentially already a good approximation, depending on the size of the database and the actual query. This allows rapid autonomous expansion of the database of CMPs allowing the robot to perform different variations of the same task in a compliant manner without the need of any analytic models of the task or programming experts.

II. LEARNING OF COMPLIANT MOVEMENT PRIMITIVES

We define Compliant Movement Primitives (CMPs) $h(t)$ as a combination of kinematic trajectories encoded in Dynamic Movement Primitives (DMPs) and corresponding task-specific dynamics encoded in Torque Primitives (TPs)

$$\boldsymbol{h}(t) = [\boldsymbol{p}_d(t),\ \boldsymbol{\tau}_f(t)], \tag{1}$$

where $\boldsymbol{p}_d$ are the desired task-space trajectories encoded in the DMPs, and $\boldsymbol{\tau}_f$ are the corresponding task-specific feedforward joint torques encoded in TPs. In the proposed approach the kinematic motion trajectories are first obtained by human demonstration and encoded as DMPs [5], [12]. Next the corresponding torques are obtained using recursive regression based on error learning. The corresponding Torque Primitives (TP) are encoded as a linear combination of radial basis functions. Since DMPs are encoded in task-space, the error mapping of the recursive regression for the joint-space of TPs is done using the Jacobian transpose. A pair of a DMP and a TP now describes a Compliant Movement Primitive (CMP).

A. Cartesian Motion Trajectories

The following equations for encoding Cartesian motion, i.e. the motion of the end-effector, are valid for one DOF, for multiple DOF the equations are used in parallel. For a point-to-point movement the trajectory for each DOF is described by the following system of nonlinear differential equations that specifies the attractor landscape of a trajectory y towards the anchor point g

$$\tau \dot{z} = \alpha_z(\beta_z(g-y)-z)+f(x), \tag{2}$$

$$\tau \dot{y} = z, \tag{3}$$

$$\tau \dot{x} = \frac{-\alpha_x x}{1+h}, \tag{4}$$

where x is the phase variable and h is used for modifying the execution speed, i.e., usually to slow it down. Therefore it is usually referred to as the slow-down feedback parameter. Note that when controlling more DOF, the phase variable x is common for all DOFs, while (2) and (3) are separate for each DOF. τ is the temporal scaling parameter, α_z, β_z and α_x are defined such that the system converges to the unique equilibrium point. Usually $\alpha_z = 4\beta_z$, which makes system critically damped. The nonlinear term $f(x)$ is given by

$$f(x) = \frac{\sum_{i=1}^{N}\psi_i(x)w_i}{\sum_{i=1}^{N}\psi_i(x)}x, \tag{5}$$

where parameters w_i define the dynamics of the second-order differential equations system. They are estimated with regression such that the DMP encodes the desired trajectory. The basis functions $\psi_i(x)$ are given by

$$\psi_i(x) = \exp(-h_i(x-c_i)^2). \tag{6}$$

N radial basis functions with a width $h_i > 0$ and centres c_i are distributed along the trajectory.

B. Joint Torque Trajectories

The Torque Primitives are learned recursively by executing the encoded DMP motion while using low gain impedance control. The desired motion $\ddot{\boldsymbol{p}}_d$, $\dot{\boldsymbol{p}}_d$, $\boldsymbol{p}_d$ encoded in DMPs is executed using the following control law

$$\boldsymbol{\tau}_u = \mathbf{J}^T(\mathbf{K}_p\boldsymbol{e} + \mathbf{K}_d\dot{\boldsymbol{e}} + \mathbf{K}_i\ddot{\boldsymbol{e}}) + \mathbf{N}\mathbf{K}_n\dot{\boldsymbol{q}} + \boldsymbol{\tau}_f(s), \tag{7}$$

where, $\mathbf{J}^T$ is the Jacobian transpose, $\mathbf{N}$ is the null-space matrix, $\boldsymbol{e}$, $\dot{\boldsymbol{e}}$ and $\ddot{\boldsymbol{e}}$ are the differences between desired and actual position $\boldsymbol{p}$, velocity $\dot{\boldsymbol{p}}$ and acceleration $\ddot{\boldsymbol{p}}$, respectively and $\mathbf{K}_p$, $\mathbf{K}_d$, $\mathbf{K}_i$ and $\mathbf{K}_n$ are the constant gain matrices selected such that the robot behaves compliantly, i.e. set to match the low impedance control requirements. For details on Cartesian DMPs including rotations see [21]. The $\boldsymbol{\tau}_f(s)$ is vector of feedforward torque trajectories $\boldsymbol{\tau}_f(s) = [\tau_{f,1}(s), \tau_{f,2}(s), ..., \tau_{f,j}(s), ..., \tau_{f,M}(s)]^T$, where M is the number of DOF. For one DOF, $\tau_{f,j}(s)$ it is given by

$$\tau_{f,j}(s) = \frac{\sum_{i=1}^{N}\psi_i(s)w_{i,j}}{\sum_{i=1}^{N}\psi_i(s)}, \tag{8}$$

where the phase s goes form 1 towards 0, similarly to the DMP phase from (4), resulting in

$$\tau \dot{s} = -\alpha_s s. \tag{9}$$

Note that unlike the DMPs, the TPs cannot be time invariant, and cannot be stopped, i.e., the torque does not scale linearly. Therefore an execution time must always be provided in advance, i.e., set by parameter τ. However, for each movement variant, e.g., the same movement but at a different speed, a new TP has to be obtained. A library of TPs for a separate motion, or a library of complete CMPs can be build. Generalization and/or graph search can be used to execute new, previously not explicitly learned movements, see details in [22].

However, in some cases, when the desired movement are not covered by the database, the method which can autonomously learn new CMPs needs to be used. To learn new CMPs we propose a new method that recursively updates the weights $w_{i,j}$ of TPs. The recursive regression method is given by

$$w_{i,j}(t+1) = w_{i,j}(t) + \psi_i P_{i,j}(t+1)\epsilon_j(t), \quad (10)$$

$$P_{i,j}(t+1) = \frac{1}{\lambda}\left(P_{i,j}(t) - \frac{P_{i,j}^2(t)}{\frac{\lambda}{\psi_i}P_{i,j}(t)}\right), \quad (11)$$

$$(12)$$

where $P_{i,j}$ is the covariance. The initial parameters are set to $P_i = 1$, $w_i = 0$, $\lambda = 0.995$ and the update rate is defined similar as in [23], with

$$\boldsymbol{\epsilon}(t) = \mathbf{J}^T(\alpha_t(\boldsymbol{p}_d(t) - \boldsymbol{p}(t)) + \beta_t(\dot{\boldsymbol{p}}_d(t) - \dot{\boldsymbol{p}}(t))). \quad (13)$$

Here the rate of learning is defined by setting the parameters α_t and β_t. The error vector is defined as $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, ..., \epsilon_j, ..., \epsilon_M]$, where M is the number of DOFs. Note that each DOF is updated separately using (10)-(11). The learning, i.e., the motion execution with learning, is repeated as long as the desired error metric is not below the desired threshold. Once the required accuracy of motion is met, either only the TP or the complete CMP can be added into the database.

III. AUTONOMOUS DATABASE EXPANSION

Autonomous learning of CMPs simplifies the execution of dynamically versatile tasks while ensuring accurate and compliant execution of the motion. However, since torques are not linearly scalable, TPs have to be learned for every variation of the task. These include, for example, different speeds, payloads, goals, etc. This new learning can be, however, avoided or at least significantly accelerated by using statistical generalization techniques, which can generate first approximations of the TPs based on a given query point. In case the generalized TPs satisfy the given error criteria, they can be immediately added to the database of motion. If not, the recursive regression method (Section II) can be applied using the generalized TP for the initial approximation, vastly reducing the number of needed iterations of motion.

Assuming that the robot should accurately track the desired trajectory encoded in DMPs, the sum of task space error throughout the iterations is used to determine if the new TPs should be added to the database, i. e. $\boldsymbol{H}_x^{\mathrm{TP}}$. The error metric is defined by

$$\boldsymbol{e}_p = \sum_{j=1}^{L} ||\boldsymbol{e}(j)||, \quad (14)$$

where $\boldsymbol{e}(j)$ is the vector of absolute difference between the actual task space position $\boldsymbol{p}$, and the desired position (DMP) $\boldsymbol{p}_d$. L is the number of steps inside one movement execution (iteration).

By encoding the torque signals as TPs, we obtain a set of M examples

$$\boldsymbol{H}_x^{\mathrm{TP}} = \{\boldsymbol{w}_{\tau k}, \boldsymbol{c}_k\},\ k = 1, 2, ..., M, \quad (15)$$

where a CMP, defined by weights $\boldsymbol{w}_q$ in DMP and weights $\boldsymbol{w}_\tau$ in TP is used to execute a task, defined by the query $\boldsymbol{c}$, with a low feedback gain and thus in a compliant manner. By using Gaussian process regression (GPR) for statistical regression

$$\boldsymbol{F}_{\boldsymbol{H}_x^{\mathrm{TP}}} : \boldsymbol{c} \longmapsto [\boldsymbol{w}_q, \boldsymbol{w}_\tau]. \quad (16)$$

we can compute the appropriate TP parameters for the given query $\boldsymbol{c}$ i. e., for the task variation. For the details on generalization for DMPs see [20] and for CMPs see [22];

The process of learning TPs repeats as long as the $\boldsymbol{e}_p > \boldsymbol{e}_c$, where $\boldsymbol{e}_c$ is a predefined constant. Once the following criteria is met, the TPs are added into the database of motion $\boldsymbol{H}_x^{\mathrm{TP}}$. With the proposed approach, the database of CMPs can be autonomously expanded.

IV. EXPERIMENTAL EVALUATION

The proposed method was evaluated in simulations and on a real humanoid robot CoMaN developed by at the Italian Institute of Technology [24].

The method was evaluated in three different scenarios. First, demonstrating the ability to learning new torque primitives while moving an arm to reach towards a certain point in space in a well structured and unconstrained space. Second, learning new torque primitives while using the previous experience by applying statistical generalisation. Third, learning of torque primitives while transferring a skill from human tutor in an unstructured environment to perform a hammering task.

The experimental setup and the initial robot pose used for the first task can be seen in Fig. 1. We denote the initial robot Cartesian position as $\boldsymbol{p}_0 = [0, 0, 0]$. The robot was commanded to reach a desired point in space. This task was chosen because it is similar as the one used in studying human learning of sensory motor ability reported in [17]. To compare the behaviour of the proposed controller to human sensory motor ability, we have choose 8 different end points on a $\boldsymbol{p}_x - \boldsymbol{p}_y$ plane equally spaced on a circle with radius of 14 cm. The Cartesian reaching trajectories were then encoded in the DMPs as parts of CMPs. The desired trajectories are shown as red-doted lines in Fig. 2. To learn the proper torque primitives an algorithm ,i.e. Section II, was employed. Note that the desired trajectories are encoded in the Cartesian space, i. e. task-space, and the corresponding torque primitives are encoded in the joint space, see Eq. (7). The mapping of the task-space error $\boldsymbol{e}$ to the joint-space was done by using the task-space Jacobian transpose. The learning of the TPs was done recursively using Eq. (10)-(13)

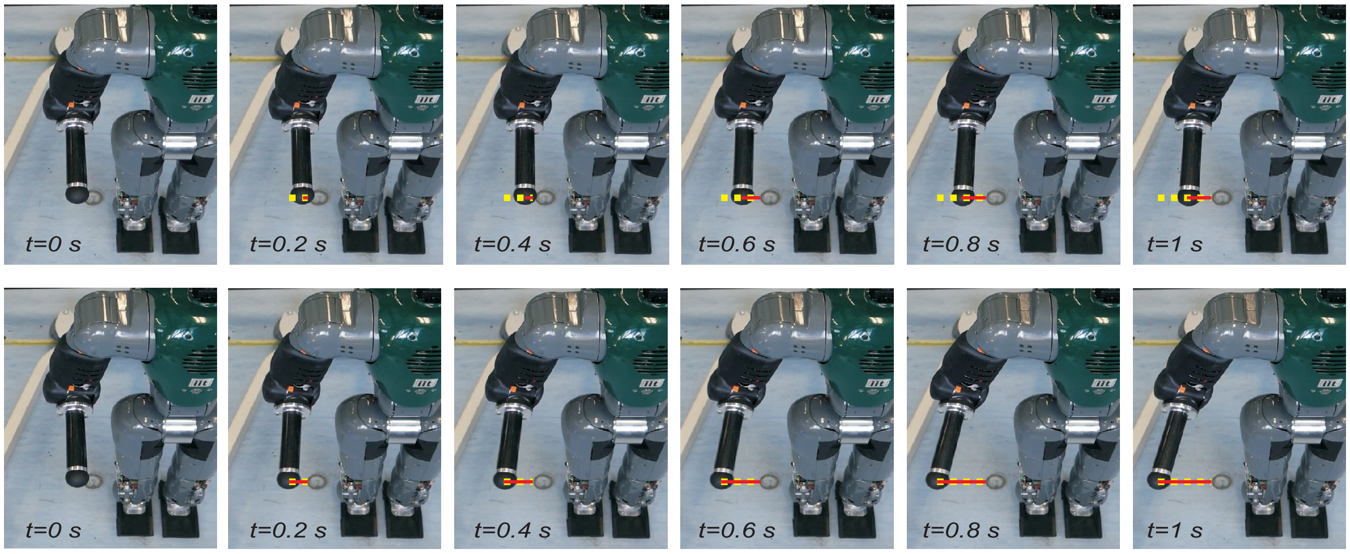


Fig. 1. Sequence of motion for one representative example, i.e. moving to the left. The top row shows the sequence of motion for the first iteration where the TPs are zero, and the bottom plot shows the sequence of motion after learning of TPs.

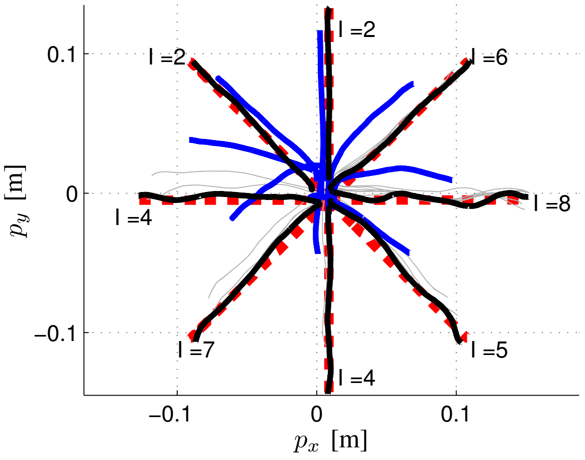


Fig. 2. Results from the real CoMan robot shows eight different reaching examples. The desired trajectory is shown with doted red line, initial trial is shown with blue line, grey thin lines show the intermediate learning iterations and the black thick line shows the learned behaviour. The I indicates the number of iterations needed for successful learning of TPs, i. e. once the error e was smaller than the desired threshold, see Eq. (14).

in each time step. Note that each executed motion started from the same initial position.

The learning results for all 8 examples are shown in Fig. 2, where we can see the difference between the initial movement execution (blue lines) and the last iteration of learning (black lines). We can see that in all eight examples the proposed approach was able to significantly improve the tracking accuracy by updating the TPs. Note that in this example no inverse dynamics or friction compensation of the robot arm was used.

The successful learning is also shown in the Fig. 1, where we compare the behaviour on one representative example, i.e. moving to the left. The top row of Fig. 1 shows the sequence of motion for the initial movement execution where the initial torque primitives were zero. It can clearly be seen that without TPs the impedance control with low gains is not able to track the desired motion. However, if the TPs are learned using the proposed human inspired controller, we can see in the bottom sequence of Fig. 1, that the tracking is significantly improved. In fact, we can see perfect matching between the desired (yellow dotted line) and actual position (red line).

Although learning of the TPs simplify the execution of dynamically versatile task, TPs still need to be learned for every variation of task execution. If no prior knowledge is used during the learning, this might be a time consuming task. However, by storing the known examples into the database, and applying statistical generalization for estimating the initial TPs for the first trial, we can significantly improve the rate of learning. In fact if the initial approximation already satisfied given criteria it can immediately be added into the database of CMPs motions. Otherwise, the proposed learning of TPs can be applied to update them accordingly.

The evolution of torques for one representative example, i.e. moving sideways towards $g = [0.1, -0.1]$, for all four joints of the right hand, of TPs for both learning without or with a prior knowledge with generalization, is shown in Fig. 3. The left hand side plots shows the evolution of torques of learning without generalisation, and the right hand side plots shows the evolution with the use of prior knowledge and statistical generalization.

The red doted line shows the initial state of TPs, the blue lines shows the TPs during learning iterations and the black line shows the TPs once the learning criteria was met. We can see on the left side that 5 iterations were needed for learning reaching the desired criteria, i. e. desired accuracy of motion. On the other hand on the right side we can see that with the use of statistical generalization the initial TPs were already close to the final solution. Therefore, fewer

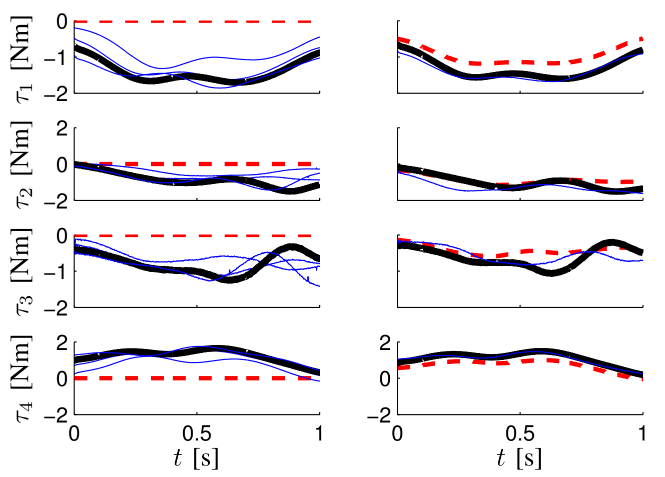


Fig. 3. The comparison between learning of TPs without (left hand side) and with (right hand side) prior knowledge, using statistical generalization for initial TPs estimation. The evolution of torques is shown for one representative example, i.e. moving to the side $g = [0.1 - 0.1]$. The red doted line shows the initial TPs, blue line shows TPs during learning iterations, and black line shows learned TPs.

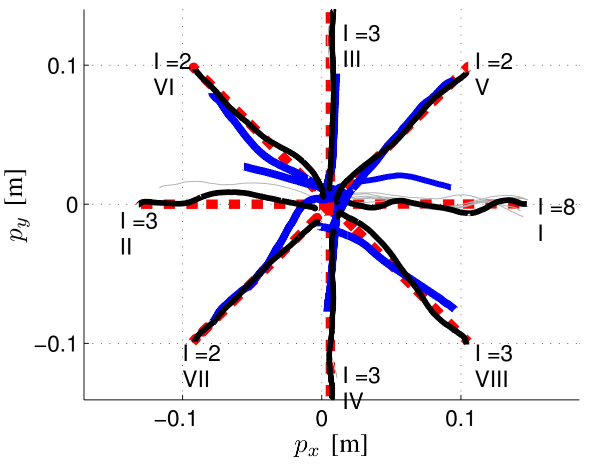


Fig. 4. Results from the real CoMan robot shows eight different reaching examples using prior knowledge for initial TPs approximation. The sequence of learning is determined with the Roman number. The desired trajectory is shown with doted red line, initial trial is shown with blue line, grey thin lines shows the intermediate learning iterations and the black thick line shows the learned behaviour. The I indicates the number of iterations needed for successful learning of TPs, i.e. once the error e was smaller than desired treshold, see Eq. (14)

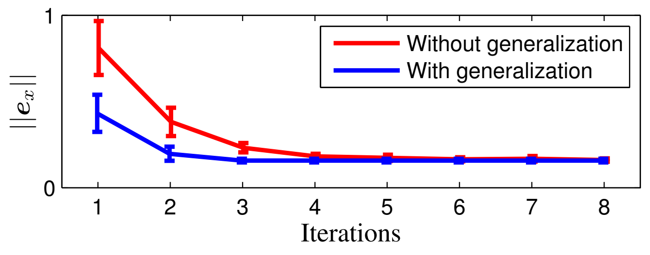


Fig. 5. The average error and the standard deviation for all eight examples for both cases, i.e. with and without prior knowledge and statistical generalization.

learning iterations were needed for satisfying the criteria. By using statistical generalization to define the initial torques and further applying recursive regression for updating TPs to minimise the error shows also that the proposed algorithm has the ability to re-learn the torques if needed.

By combining the statistical generalization based on prior knowledge in the database and the recursive regression algorithm for updating the TPs, we can rapidly extend the database with exploration. An example showing rapid database expansion with learning of TPs is shown in Fig. 4, where the database of CMPs was gradually expanded. The learning sequence is indicated with Roman numbers. Note that for the first trajectory, i. e. I, the database was empty, therefore the behaviour is similar as in the Fig. 2. For the second case, i. e., II, only the knowledge from case I was in the database, therefore the initial TPs were completely wrong. Nonetheless, the proposed approach was able to update the TPs to meet required criteria, which show that the proposed approach has the ability to re-learn the TPs if needed. This also shows that the proposed approach can cope with sudden changes in the task dynamics and re-adapt if needed.

The ability to re-learn is also one of the key features that a human possess. It was shown in [17] on a similar reaching experiment as presented here but with humans, that they also have to adapt to changes in the task dynamics, i.e. if the task dynamic is different than before even humans have to re-learn. From Fig. 4, we can also see that once more knowledge is present in the database, the initial TPs estimations are better, from which follows that fewer iterations for updating TPs are needed to meet the required criteria.

In general the learning rate of the proposed algorithm not only depends on the initial TPs state but also on gains α_t and β_t. If these gains are too high, the system might potentially be destabilised, or vice versa, if they are too low, the learning rate would not be efficient. In our experiments we set them empirically to $\alpha_t = 1$ and $\beta_t = 0.5$, which resulted in slightly faster learning rate than reported for the human experiment [17]. In Fig. 5 we can see the learning rate for both cases, i.e. with and without using the prior knowledge for initial TPs estimation. We can see on the plot that learning is almost twice as fast if the initial TPs are defined using statistical generalization.

Since CMP is combined with low gain impedance control, we can also assume that interaction forces in case of collision with an unforeseen object will be significantly smaller compared to the impedance control with similar tracking performance. Note that to achieve similar tracking performance with only impedance control the robot would be stiff. To investigate the performance in case of a collision, we chose a task of hammering. Here the robot needs to learn the tool dynamic, i.e. the dynamics of the hammer, and then interact with the environment. In case of hammering the impact with the environment is instantaneous.

The results of learning the skill of hammering are shown in Fig. 6. The top plot shows the adaptation of kinematic

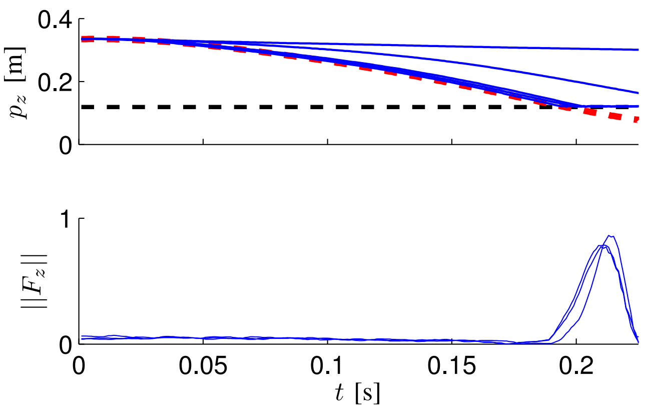


Fig. 6. Learning if dynamics while learning skill of hammering. Top plot shows the z motion in Cartesian space. The red doted line is the desired trajectory, the blue lines shows the learning iterations and the black doted line is the table height. Bottom plot shows the estimated impact force.

motion and the bottom plot shows the estimated interaction forces. Since no dynamical model was used, the robot had to learn the complete dynamical model of the hand and the tool, i. e. the hammer. This also explains why the robot was not able to track the desired trajectory in the first two iterations. Once the TPs were properly adapted to the task dynamic, the tracking error was small up to the point of contact with environment. Since low impedance control with CMPs was used, the interaction force is mainly a result of inertia of the robot and the hammer.

V. CONCLUSIONS

We have showed that the proposed approach (CMPs) can successfully learn both the kinematic trajectory in Cartesian space encoded as a DMP and the corresponding dynamics encoded as a TP. The main contribution of this paper is a new approach for autonomously learning previously unknown TPs based on a given DMP and storing them in a database. Moreover, we propose to exploit previous experiences and statistical learning to accelerate the learning of TPs. We demonstrated that this way we can significantly improve the rate of learning and rapidly expand the database of TPs/CMPs. The proposed approach significantly improves the tracking accuracy of the low gain impedance control. Low gain impedance control implies low impact forces in case of unforeseen collisions, which makes robot safer for working in unstructured environment or with humans. As such, the proposed approach enables simple and computational inexpensive control of dynamically challenging tasks. The obtained results can be related to the findings of studies on human sensorimotor learning abilities.

REFERENCES

[1] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey." *Cognitive processing*, vol. 12, no. 4, pp. 319–40, Nov. 2011.

[2] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 79–91, June 2004.

[3] A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47, pp. 93–108, 2004.

[4] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1463–1467, 2008.

[5] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control–a unifying view." *Progress in brain research*, vol. 165, no. 1, pp. 425–45, Jan. 2007.

[6] A. Albu-Schaffer and G. Hirzinger, "Cartesian impedance control techniques for torque controlled light-weight robots," *Proceedings 2002 IEEE International Conference on Robotics and Automation*, pp. 657–663, 2002.

[7] S. Haddadin, A. Albu-Schaffer, and G. Hirzinger, "Requirements for Safe Robots: Measurements, Analysis and New Insights," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1507–1527, Aug. 2009.

[8] J. Ulmen and M. Cutkosky, "A robust, low-cost and low-noise artificial skin for human-friendly robots," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4836–4841, 2010.

[9] D. Shin, I. Sardellitti, Y. L. Park, O. Khatib, and M. Cutkosky, "Design and Control of a Bio-inspired Human-friendly Robot," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 571–584, Nov. 2009.

[10] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, Apr. 2011.

[11] A. Gams and T. Petrič, "Adapting periodic motion primitives to external feedback: Modulating and changing the motion," in *Robotics in Alpe-Adria-Danube Region (RAAD), 2014 23rd International Conference on*, 2014, pp. 1–6.

[12] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors." *Neural computation*, vol. 25, pp. 328–73, 2013.

[13] D. Basa and A. Schneider, "Movement Primitives Learning Point-to-Point Movements on an Elastic Limb Using Dynamic Movement Primitives," *Robotics and Autonomous Systems*, 2015.

[14] N. Gopalan, M. P. Deisenroth, and J. Peters, "Feedback error learning for rhythmic motor primitives," *2013 IEEE International Conference on Robotics and Automation*, pp. 1317–1322, May 2013.

[15] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control." *Neural networks : the official journal of the International Neural Network Society*, vol. 11, no. 7-8, pp. 1317–29, Oct. 1998.

[16] H. Gomi and M. Kawato, "Neural network control for a closed-loop System using Feedback-error-learning," pp. 933–946, Jan. 1993.

[17] J. W. Krakauer, M. F. Ghilardi, and C. Ghez, "Independent learning of internal models for kinematic and dynamic control of reaching." *Nature neuroscience*, vol. 2, no. 11, pp. 1026–31, Nov. 1999.

[18] E. Oztop, M. Kawato, and M. Arbib, "Mirror neurons and imitation: A computationally guided review," *Neural Networks*, vol. 19, pp. 254–271, 2006.

[19] D. Bruno, S. Calinon, and D. G. Caldwell, "Learning adaptive movements from demonstration and self-guided exploration," in *4th International Conference on Development and Learning and on Epigenetic Robotics*. IEEE, Oct. 2014, pp. 101–106.

[20] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, Oct. 2010.

[21] A. Ude, B. Nemec, T. Petric, and J. Morimoto, "Orientation in Cartesian space dynamic movement primitives," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, no. 3, pp. 2997–3004, 2014.

[22] M. Denisa, T. Petrič, T. Asfour, and A. Ude, "Synthesizing Compliant Reaching Movements by Searching a Database of Example Trajectories," *International Conference on Humanoid Robots*, pp. 540–546, 2013.

[23] M. Kawato, "Feedback-Errror-Learning Neural Network for Supervised Motor Learning," in *Advanced Neural Computers*, R. Eckmiller, Ed. North-Holland: Elsevier, 1990, pp. 365–372.

[24] L. Colasanto, N. G. Tsagarakis, and D. G. Caldwell, "A compact model for the compliant humanoid robot COMAN," in *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2012, pp. 688–694.

Emergent structuring of interdependent affordance learning tasks using intrinsic motivation and empirical feature selection

Emre Ugur, *Member, IEEE,* and Justus Piater, *Member, IEEE*

***Abstract*—This paper studies mechanisms that produce hierarchical structuring of affordance learning tasks of different levels of complexity. Guided by intrinsic motivation, our system detects easy tasks first, and learns them in selected environments which are maximally different from the previously encountered ones. Easy tasks are learned from observed low-level attributes of the environment, and provide abstractions over these attributes. As learning progresses, the system shifts its focus and starts learning harder tasks not only from low-level attributes but also from previously-learned abstract concepts. Therefore, hard tasks are autonomously placed higher in the hierarchy if the easy task concepts are identified as distinctive input attributes of hard tasks. Use of abstract concepts allows hard tasks to be learned faster than learning them from scratch, i.e., from low-level perception only. We tested our system with the tasks of learning effect predictions for poke and stack actions using a dataset that includes 83 real-world objects. On the basis of a large number of runs of the method, our analysis shows that the hierarchical task structure emerged as expected, along with a consistent learning order. Furthermore, a significant bootstrapping effect in learning speed of the stack action was observed with the discovered hierarchy, albeit only when fully-learned poke actions were used from the beginning.**

***Index Terms*—intrinsic motivation, affordance, bootstrapping, discriminative learning, structure learning, transfer learning, re-use of concepts, prediction hierarchies, feature selection, diversity maximization**

I. Introduction

One hallmark feature of bootstrapped learning is that learning problems stack in the sense that higher-level learners use as input attributes concepts produced by lower-level learners. These higher-level attributes should allow faster learning than if the higher-level concepts had to be learned from the lower-level attributes alone. Consider an example where the robot learns stackability affordances, i.e. learns to detect if two given objects would stably stack on top of each other. In this case, the robot needs to explore a high-dimensional search space if it learns from low-level shape features of these objects such as curvatures of different sides. On the other hand, if the robot uses previously learned high-level abstractions, such as rollability, it would learn which objects can be stacked on top of each other by associating rollability and stackability from fewer examples, This is achieved because the robot already learns and encodes part of object-robot-environment dynamics in the higher-level attribute of rollability, and can re-use this attribute to bootstrap other related learning problems that share similar characteristics.

E. Ugur and J. Piater are with University of Innsbruck, Institute of Computer Science, Innsbruck, Austria.

In this paper, we formulate the skill learning problem as learning inter-dependent affordances where a robot is expected to detect affordances of different complexities by learning the relations between objects, actions and effects. In our system the affordance detection is achieved by predicting what kind of effect would be created on an object given the visual properties of the object and the action applied. Following the re-use idea, an expert can design a system which learns simple affordances first, and gradually shifts its focus to more complex affordances taking advantage of the learned simple affordances. However in a life-long learning scenario where the objects in the environment are unknown and changing, and any arbitrary action can be added to the robot repertoire any time, the difficulty of affordances cannot be known in advance by the experts. A truly developmental agent should discover the best means to organize its learning strategy so that the skill re-use is most effective without such prior knowledge. In other words, the agent should should simultaneously discover the learning order and the re-use structure of affordances.

Discovering the learning order of affordances means realizing an exploration strategy that shifts the focus of learning from simple to complex affordances. Since the robot learns from observations of the consequences of its actions on different objects, this exploration strategy practically corresponds to selecting the ‘best’ actions and objects in an active learning setting in a potentially open-ended system. For action selection, we used Intrinsic Motivation (IM) approach, which guides the robot with intelligent exploration strategies. IM is regarded as a set of active learning mechanisms for developmental robots, which enables efficient and effective learning in high-dimensional search spaces [1]. IM approach [2] in developmental robots [3] was inspired from curiosity based motivation mechanisms in human development, and has recently been effectively applied to cognitive robots where object knowledge is developed through self-exploration [4] and social guidance [5]. This approach adaptively partitions agent’s sensorimotor space into regions of exploration and guides the agent to select the regions that are in intermediate level of difficulty. This is achieved by maximizing reduction in prediction error, in other words by maximizing the learning progress. In this paper, we propose to use this approach to guide the robot to explore different affordances by adaptively selecting the actions to execute, and updating the models of the affordance predictions based on the results of these actions.

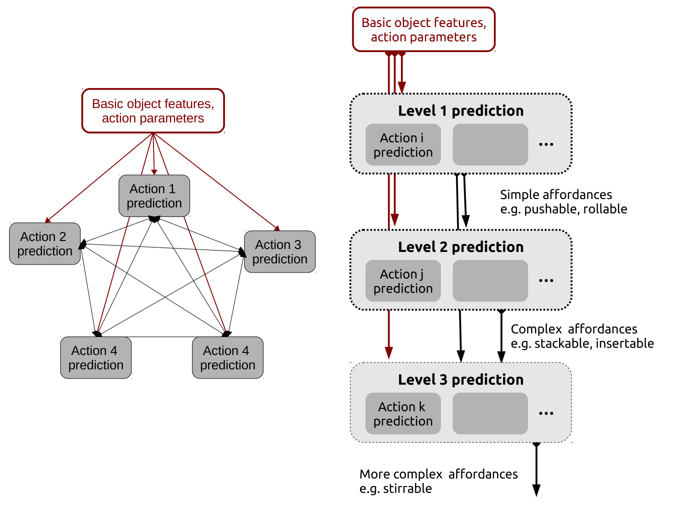
Through IM approach, we aim to achieve a developmental progression similar to those of infants in learning simple-to-complex skills and affordances.

In order to actively select which objects to interact with, on the other hand, we used a heuristic which maximizes the diversity of the objects used in training. In each learning step, the object, which is maximally different from the objects that are already used in previous learning steps, is selected with the aim to cover the variety in the object space.

Discovering the re-use structures of affordance corresponds to autonomously selecting the set of affordances that are useful in learning and predicting other affordances. Practically, this corresponds to deciding input/output links among affordance predictors. Our solution to this problem is inspired from Eleanor J. Gibson who was a developmental psychologist studying on mechanisms of affordance learning in biological systems. She argued that learning affordances is neither the construction of representations from smaller pieces, nor the association of a response to a stimulus. Instead, she claimed that learning is "discovering distinctive features and invariant properties of things and events" [6]. Learning is not "enriching the input" but discovering the critical perceptual information in that input. Following this idea, we identify distinctive inputs for each affordance predictor, and establish connection between output of an affordance predictor and input of another predictor only if the former affordance is discriminative in predicting the latter one. We formalized this through a relevant feature selection mechanism that selects the most relevant, non-redundant, and minimal set of inputs, which in turn can achieve maximal prediction accuracy.

In summary, we study the mechanisms that enable autonomous structuring of affordance learning tasks. Our system starts in a flat form as shown in Fig. 1(a) where output of each affordance predictor can be potentially used as an input of any other predictor. As learning progresses the robot actively updates these connections by selecting the most distinctive inputs of the corresponding action predictors. In detail, in each learning step, the robot selects the most 'interesting' action to perform based on Intrinsic Motivation, the most different object to explore based on diversity maximization, and updates the predictor of the corresponding action along with its distinctive inputs based on the observed effect on the object. We demonstrate these mechanisms, namely (i) the IM based selection of actions, and (ii) diversity maximization based selection of objects, and (iii) the use of the most distinctive inputs in affordance predictions, enable emergence of a hierarchical structure, similar to the one shown in Fig. 1(b). With this, the system autonomously discovers which tasks should be learned first, and which ones become simpler if expressed in terms of other tasks without any prior knowledge on the relative complexity of these tasks.

The rest of this paper is organized as follows. First, a summary of related work on affordances, re-use of learned concepts, and intrinsic motivation based robotics work is given in the next section. Next, representation of affordances, and mechanisms of active action, object and connection selection are detailed in Section III. In Section IV, the discovered learning order of predictors and evolution of the input/output



(a) Flat prediction (b) Hierarchical prediction

Fig. 1. (a) shows a flat affordance learning structure, where the affordances are predicted based on low level object features, action parameters, and all other perceived affordances. (b) shows a simple hierarchical structure where simple affordance predictions can be used to detect complex affordances. This paper aims automatic discovery of such a hierarchical structure where the lower level predictors are learned first, and connected to the higher level predictors autonomously.

connections are given along with the results on speed-up in learning complex affordances with the emerged structures. Finally, in Sections V and VI, we discuss how a number of assumptions can be addressed and how our system can be extended to handle more complex learning spaces.

II. Related Work

Affordances, which were introduced by James Gibson in his ecological approach to visual perception [7], provide the agents a 'direct' means to perceive the action possibilities provided by the environment, and act on them. As detecting action possibilities and reasoning on them are crucial in robotics, affordances concept has been very influential in the last decade. Affordance research in robotics can be coarsely divided into two categories: learning affordances from passive observations or through robots own interaction with the world. The research in the first category does not involve the robot in active learning. For example Koppula et al. [8] studied learning of object affordances along with human activities using Markov Random Fields where the nodes encode the sub-activities and affordances; and the edges correspond to the learned relations between these components. Myers et al. [9] labelled tool parts with multiple affordances and learns generalizable affordance detectors. Schoeler and Worgotter recently studied generalizing object affordances through effective transfer of part functionalities in real world objects [10]. The research in the second category, which involves learning affordances in the form of object-action-effect relations has been widely studied in robotics in recent years [11, 12, 13, 14, 15]. While all different kinds of clustering and classification methods are used in literature, the standard learning setup includes a robot that actively explores the object in the environment. It interacts with objects using generally pre-defined set of behaviors, observes the effects generated by these behaviors, and learns the relations between the behaviors, behavior parameters, object properties and generated effects. Then, given a goal, the robot can chain effect predictions to find the sequence of behaviors, effectively generating a plan to reach that goal. Our approach in this paper belongs to the second category as we also study learning effects of actions of the robot on different objects using self-discovered effect categories that were transferred from previous robot interactions[12, 16].

Re-use of learned affordances and relational affordances have been addressed by only a few studies. Transfer of learned single-object affordances to bootstrap learning of paired-object affordances was studied by Moldovan et al. [17, 18]. The authors first used Bayesian Networks (BN) to learn (object, action, effect) relations from single and paired-object interactions, where relative position and orientation were encoded explicitly. Single-object action rules were transferred in learning paired-object action models when objects do not interact with each other. A relational knowledge representation model was derived from the BN affordance models, and later generalized to arbitrary number of objects. The system, which learned probabilistic rules for push and grasp actions in single and paired-object settings, were able to generalize its predictions to multi-object settings. Different from our model, Moldovan et al. focused on generalizing the rules that encode the position and orientation relations between objects; whereas our focus is to learn complex affordances that encode non-linear relations between arbitrary features of objects. In [18], the authors extended their system to continuous settings, however high-level shape primitives such as cubes and cylinders were pre-defined, whereas our system can discover such primitives from low-level shape features. Fichtl et al. also used predictions of action effects as inputs in predicting effects of other actions [19]. In a setting with 9 simulated actions, they used effect predictions of one or two actions in predicting effects of other actions. They showed that bootstrapping in learning speed can be achieved in the initial phases of learning especially if the learning problem is hard. This study is similar to ours as we also use of outputs of predictors as inputs of other predictors. However, our system can discover the input/output structure itself, and co-develop affordance predictors in an active learning setting that is guided by Intrinsic Motivation.

Intrinsic Motivation (IM) was first used as a 'manipulation drive' that explains why monkeys spend time on mechanical puzzles for long durations without any extrinsic reward [20], Activities that do not directly serve the goals of survival or material advantage such as play, curiosity, interest in novel stimuli and surprising events are said to be driven by intrinsic motivations [21, 22]. In order to achieve an open-ended development like infants, this mechanism has been heavily utilized in robotics in the last decade. Law et al. [23] realized a staged development with iCub that models an infant from birth to 6 months. iCub, through motor babbling driven with a novelty metric, started from uncontrolled motor movements, passed through several distinct behavioral stages, and achieved reaching and basic manipulation of objects, similar to the human infants. Ivaldi et al. [5] proposed a system where the iCub humanoid robot learned object properties by actively choosing among objects to explore, actions to execute and caregivers to interact. This socially guided intrinsic motivation framework [24] that combined robot's manipulatory actions with social guidance significantly increased object recognition performance, and could be directly used to increase speed of affordance learning. Hart and Grupen [25, 26] realized a longitudinal development, where a robot self-organized its sensorimotor space by assembling basic actions into hierarchical programs in a bottom-up way, and by learning to apply these programs in novel contexts in a top-down fashion. The staged learning of behaviors was guided by an intrinsic rewards mechanism that maximizes detection of and acting on affordances with the corresponding behaviors. Hart and Grupen's work was focused on behavior formation in a staged progression with mechanisms similar to accommodation and assimilation [27] through the so-called affordance discovery motivator with emphasis on closed-loop control programs as coupled dynamical systems. Other researchers used intrinsic motivation for autonomous acquisition of motor skills [28], and for autonomous selection of tasks to explore based on a measure of competence progress [29]. Please see Baldassarre et al. [21] for a recent Electronic Book (eBook) that compiled large number of interdisciplinary articles on Intrinsic Motivation within Neuroscience, attention, robotics, and social interactions research.

We previously studied intrinsic motivation and relevance based affordance learning in [30], and bootstrapping complex affordance learning with learned simple affordances in [31]. The method, discussions, and analysis of the system have been significantly extended in this paper as follows. First of all, the main components of our system, namely IM-based action selection, diversity-based object selection, and relevance based distinctive input selection are first time combined and analyzed in an integrated framework. Second the current work rigorously analyzes the proposed approach by running the algorithm large number of times with different configurations, and by providing the statistics obtained from these several runs. Finally, all material presented in the results section is new, and supported by more thorough discussions.

III. Active Learning of Affordances

A. Affordance representation

In this paper, affordances of an object (aff^o) correspond to the list of effects generated by the set of available actions:

$$\mathit{aff}^o = (\varepsilon^o_{a_1}, \varepsilon^o_{a_2}, ...)$$

where $\varepsilon^o_{a_1}$ is the discrete effect created on object o by action a_1. For example, a lying cylinder affords rollability when pushed from one side, pushability when pushed from another side, and liftability when grasped. The affordances of this cylinder is represented by the list of effects created with push and grasp actions, i.e. {rolled, pushed, lifted}.

A robot can predict the effect of an action on an object based on the visual properties of the object and/or through reasoning over how the object is affected from other actions.

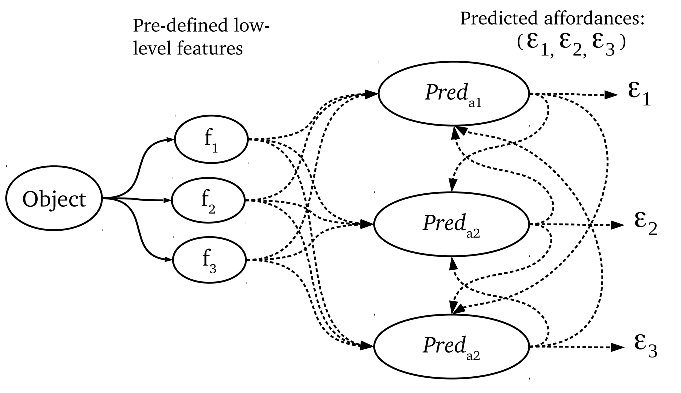

Pre-defined low-level features
Predicted affordances: (ε1, ε2, ε3)
Object
f1
f2
f3
Pred a1
Pred a2
Pred a2
ε1
ε2
ε3

Fig. 2. Input/output links of the affordance predictors in a setup with three low-level features and three single-object actions.

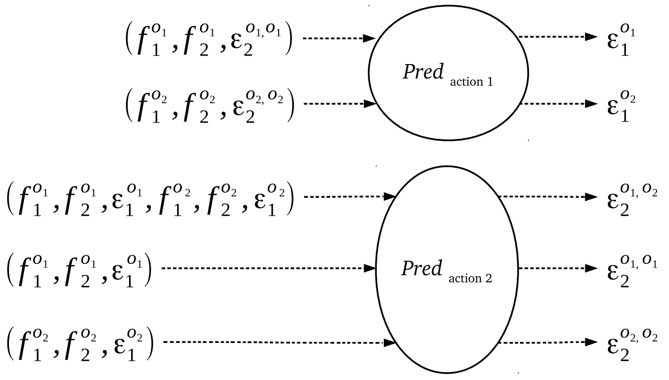


Fig. 3. Input/output of the affordance predictors in a setup with two low-level features, one single-object actions, and one paired-object action. In order to give a clear picture, we used a two action setting and showed only the important information.

In this paper, we will call the first source of information as 'low-level features' of objects, and the second source as 'high-level features'. While the low-level features such as size and shape are previously defined by the programmer, the high-level features will be learned by the system. These features are used as inputs to $Pred$ operator that predicts the effect of action a_i on object o as follows:

$$\varepsilon^o_{a_i} = \mathit{Pred}_{a_i}(o) = \mathit{Pred}_{a_i}(\mathit{feat}^o, \mathit{aff}^o \backslash \varepsilon^o_{a_i})$$

Above, $\mathit{feat}^o = (f^o_1, f^o_2..)$ corresponds to the low-level features. High-level features, on the other hand, are represented by aff^o, which encodes 'other affordances', i.e. the effect predictions for other actions:

$$\mathit{aff}^o \backslash \varepsilon^o_{a_i} = \{\mathit{Pred}_{a_i}(o) | a_i \neq a_j\}$$

Actions that involve interaction with single objects and with pairs of objects are called single-object actions and paired-object actions, respectively. Likewise, affordances that are offered by single objects and pairs of objects are called single-object affordances and paired-object affordances, respectively.

Fig. 2 gives the general input/output structure for an affordance prediction system in a setup that includes three different types of pre-defined low-level features, and three single-object actions. Each predictor takes low-level features and the effect predictions of other actions as inputs. The links illustrated by the dashed lines correspond to only potential connections, which are selectively established by the learning system only if the corresponding links are necessary for predictions. While establishing the links, cyclic dependencies are not allowed because a predictor can produce an output only if all its inputs are available. In our previous work [30] we did not need to disallow cyclic dependencies because predicted features were attached to object identities, obliterating the need to re-predict. Note that outputs of predictors before initialization are fixed to non-existing effect categories (-1).

For simplicity, we described the effect prediction mechanism using actions that involve interactions with single objects. This prediction mechanism also supports actions that involve more than one object. In multi-object case, features and affordances of all the involved objects are used as input attributes of the corresponding action predictor. Fig. 3 illustrates the potential links in a setup with two low-level features, one single-object action, and one paired-object action. In paired-object actions, the predictor takes the following form:

$$\varepsilon^{(o_1,o_2)}_{a_i} = \mathit{Pred}_{a_i}(\mathit{feat}^{o_1}, \mathit{feat}^{o_2}, \mathit{aff}^{(o_1,o_2)} \backslash \varepsilon^{(o_1,o_2)}_{a_i}) \quad (1)$$

Paired-object affordances, therefore, correspond to the collection of effects obtained from single-object actions for each object, and paired-object actions for each pair:

$$\mathit{aff}^{(o_1,o_2)} = (\varepsilon^{o_1}_{a_1}, \varepsilon^{o_1}_{a_2}, ...\varepsilon^{o_2}_{a_1}, \varepsilon^{o_2}_{a_2}, ..., \varepsilon^{(o_1,o_2)}_{a_i}, \varepsilon^{(o_1,o_2)}_{a_j})$$

Multi-class Support Vector Machines (SVMs) with Radian Basis Function kernels and optimized parameters are used to learn these predictors [32]. Although we used a batch version of SVM as implemented in LibSVM library [33] for practicality, incremental versions, which are more suitable in active learning setups, were also shown to be provide similar performance [34]. Our focus in this paper is how to discover the connection structure and learning order of these predictors. Therefore the details of prediction mechanisms have minor importance, and one can replace SVMs with their favorite classification method as long as the following active learning principles are preserved.

B. General learning approach

We followed an active learning approach to train the complete prediction system. This learning is achieved in episodes, whose major steps are summarized in Fig. 4. In the first step, the action that has the highest learning progress is selected for exploration. Next, a number of objects that are maximally different from the previously selected objects are selected. Based on the observed effects on these objects, the effect predictor of the corresponding action is updated by updating the SVM classifiers described in previous paragraph. During this update, the input links for this predictor are also found and these established connections are registered to the system.

C. Active selection of actions

The first step of each episode is to select which action to explore next. As mentioned in Section I, we used Intrinsic

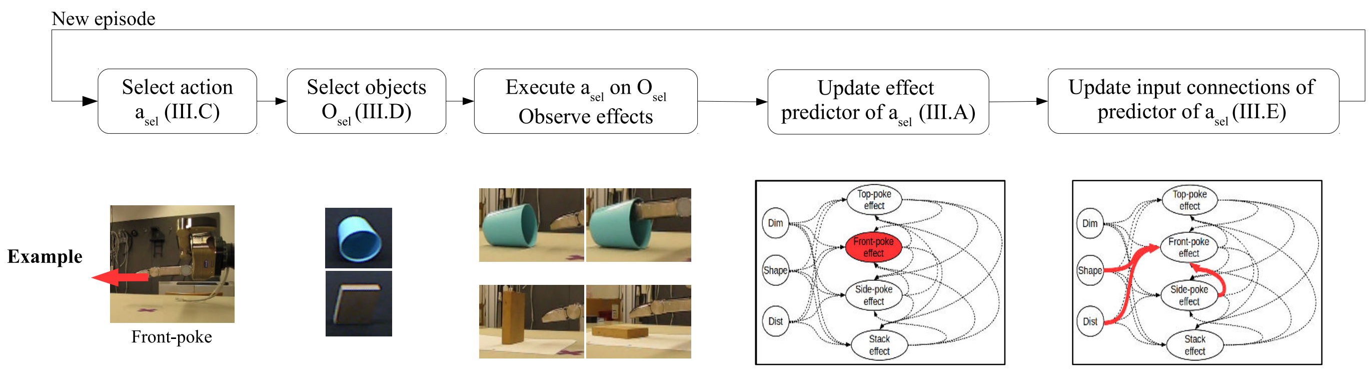

New episode
Select action a_sel (III.C)
Select objects O_sel (III.D)
Execute a_sel on O_sel Observe effects
Update effect predictor of a_sel (III.A)
Update input connections of predictor of a_sel (III.E)
Example
Front-poke
Dim
Shape
Dist
Top-poke effect
Front-poke effect
Side-poke effect
Stack effect

Fig. 4. Summary of one learning episode.

Motivation (IM) criteria in active selection of the action. In its original formulation [3], IM can adaptively partition exploration space into distinct regions and guide the agent towards the regions that are neither too complicated nor too simple. The medium difficulty is formalized by guiding the agent to regions that provide maximal learning progress. In a similar manner, our system actively selects the actions with maximal learning progress. Learning progress (LP) of an action is defined based on the actual increase in the mean prediction accuracy of the predictor of the corresponding action($Pred_{a_i}$):

$$LP_{a_i}(t+1) = \overline{\gamma}_{a_i}(t+1) - \overline{\gamma}_{a_i}(t+1-\tau)$$

where $\overline{\gamma}_{a_i}(t+1)$ and $\overline{\gamma}_{a_i}(t+1-\tau)$ are defined as the current and previous mean prediction accuracies of the effect predictor, and τ is a time window, set to 2.

Here we define mean prediction accuracy by setting a smoothing parameter θ to 5:

$$\overline{\gamma}_{a_i}(t+1) = \frac{\sum_{j=0}^{\theta} \gamma_{a_i}(t+1-j)}{\theta+1}$$

where predictor accuracy of the action ($\gamma_{a_i}(t)$) is equal to the ratio of the correct predictions on objects explored by the action at step t. This is only a local measure that approximates the real accuracy. We used this local accuracy measure in our online incremental learning setup as the robot cannot access to ground truth, i.e. it cannot know the effect categories of the objects without actually executing its actions on all of them in a real setting.

Finally, the next action to be explored is selected based on the above learning progress criteria using ϵ-greedy strategy [35].

$$a_{\text{sel}}^t = \begin{cases} a_r & \text{if } t < t_{\text{init}} \\ a_r & \text{if } \zeta < \epsilon \\ \arg\max_{a_i} LP_{a_i}(t) & \text{otherwise} \end{cases}$$

where a_{sel}^t denotes the selected action at learning step t, $0 \geq \zeta \geq 1$ is a uniform random number, and ϵ is set to 0.10. This IM based action selection strategy requires an initial random exploration phase where the learning progress (LP) of each predictor is initialized. t_{init} is empirically set to 48 interactions.

D. Active selection of objects

After selecting the action, the system needs to decide which objects to interact with this action. For this, we propose a heuristic that maximizes the diversity of the objects used in training of the corresponding predictor. The system chooses an object from the available set of objects with maximal distance to the set of already interacted objects (O_{used}). The distance between pairs of objects can be computed by calculating the distance between feature vectors that represent the corresponding objects. However, each object feature channel is represented in a different metric, therefore the distance would depend on the relative weighting of the features in different channels. Here, we propose a heuristic that selects computes the Euclidean distance in a randomly selected feature channel space:

$$o_{\text{sel}} = \arg\max_{o_1 \in O \backslash O_{\text{used}}} \sum_{o_2 \in O_{\text{used}}} dist_{\mathfrak{c}}(o_1, o_2)$$

where O, O_{used}, and $O \backslash O_{\text{used}}$ correspond to the set of all objects, the set of objects used for training the corresponding predictor, and the set of the objects not used yet. $\mathfrak{c}$ corresponds to the feature channel where distance is calculated, and is uniformly sampled from the set of low-level features $\{$size, shape, distance$\}$. If the predictor of the selected action, $Pred_{a_{\text{sel}}}$, takes input from other effect predictions, a random effect prediction from the corresponding set ($\{\varepsilon_{a_i} | a_i \neq a_{\text{sel}}\}$) is used instead:

$$o_{\text{sel}} = \arg\max_{o_1 \in O \backslash O_{\text{used}}} \sum_{o_2 \in O_{\text{used}}} \| \varepsilon_{a_i}^{o_1} - \varepsilon_{a_i}^{o_1} \|$$

In each episode, we chose four objects if a_{sel} is a single-object action, and four pairs of objects if a_{sel} is a paired-object action. Empirically, we observed that this active object selection strategy is effective only if it is applied in the initial training steps for each predictor. If it is applied for complete training, the choice of objects degenerates probably because the whole range of diversity cannot be represented by the available features, thus the overall performance of the predictors decrease. Therefore, we applied this active object selection strategy in the first 12 steps, which is a value empirically found. After 12 steps, o_{sel} is always selected randomly from $O \backslash O_{\text{used}}$.

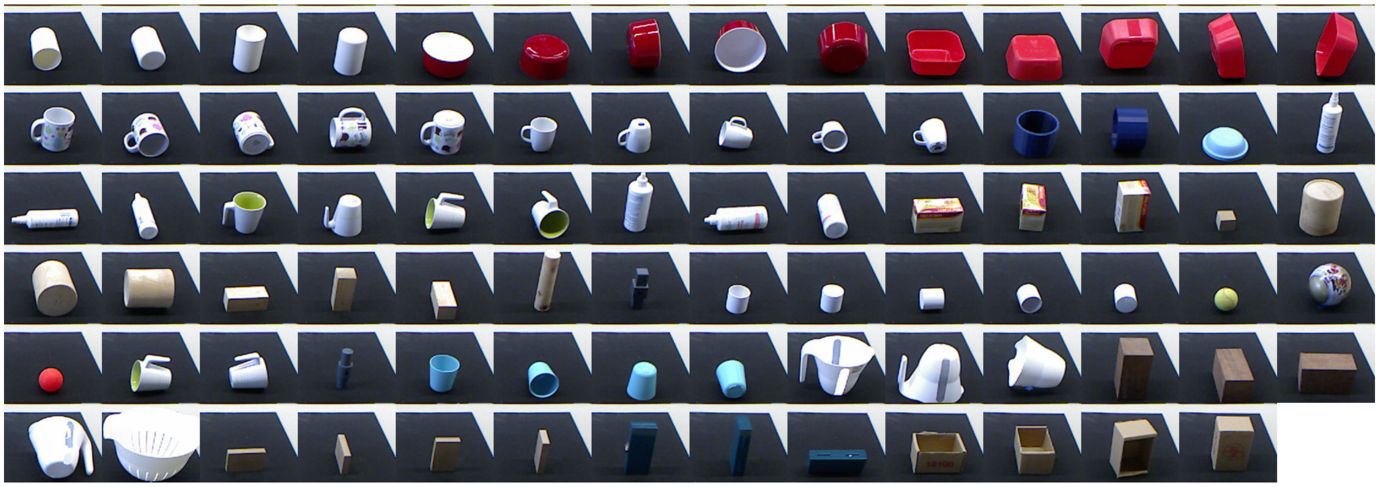


Fig. 5. Objects used in the experiments.

E. Active selection of input connections

The input connections of the $Pred_{a_{sel}}$ (therefore input/output connections of the whole system) are updated in this step. For this, the minimal set of input connections that can achieve maximum available prediction accuracy for $Pred_{a_{sel}}$ is selected. One of the feature selection methods, which generates near-optimal feature sets [36], namely Sequentialfs, is used for this purpose. Each input connection in our system correspond to either a low-level feature channel or prediction of another action. The algorithm starts with an empty input set. At each iteration, a new input is selected and added to the input set of previous iteration. In order to select this new input, all candidate inputs are separately added to the previous input set and separate candidate input sets are formed. Then, the candidate input sets are evaluated through 10-fold cross-validation on trained predictors. The best performing candidate set is then transferred to the next iteration. In the experiments, we eliminated the ones that have no effect in accuracy increase, finalizing the most distinctive inputs for each trained predictor $Pred$. Therefore, we expect minimal redundancy in the input connections. Finally, in order to avoid cyclic predictions, the output of $Pred_{a_i}$ is not considered as a potential/candidate input for $Pred_{a_j}$, if the output of $Pred_{a_j}$ is previously selected as an input for $Pred_{a_i}$.

IV. Experiment Setup

1) Low-level object features: The system can visually detect the objects in the environment using the depth information of Kinect sensor, and compute a number of features from the point cloud data. $feat$ represents the continuous low-level feature vector that combines various visual properties of the object. It is composed of three different channels, which are feature vectors themselves:

$$feat^o = (dim^o, shape^o, dist^o)$$

dim represents the dimensions of the object in three different axes. $shape$ corresponds to the distribution of normal vectors obtained from the local surfaces on the point cloud. Normal vectors are obtained using Point Cloud Library[37], normal estimation package. This distribution is encoded by histograms of normal vectors, where each component in different axis is regarded separately. 8 bins are used to discretize the distribution in each axis, making the total size of $shape$ feature vector is $3 \times 8 = 24$. Finally, in order to capture the local discontinuities and distribution of distances in the neighborhood of each voxel, we use $dist$ feature channel. For each voxel, the neighboring voxels are identified in the 2-d depth image, and distances to the neighbors are computed along each four direction. For each direction, we created a histogram of 20 bins with bin size of $0.5cm$, obtaining a $4 \times 20 = 80$ sized vector for the $dist$.

2) Actions: The robot [1] is equipped with a number of manually coded actions that enable single and multi object manipulation. The robot can poke a single object from different sides using *front-poke*, *side-poke*, and *top-poke* actions. It can also stack one object on the other using *stack* action, where it grasps the first object, moves it on top of the other one and releases it.

3) Interaction Dataset: An interaction dataset, which is composed of (object, action, effect) tuples, is generated to run and analyze our method. 83 objects with different sizes, shapes, and affordances were used for this purpose (Fig. 5). The low-level visual features of these objects were computed by placing them on the table and by extracting the point cloud using the Kinect sensor. A number of sample robot interactions that involve some of these objects are given in Fig. 6 and Fig. 4 where different effects can be observed. With three single-object actions, and one paired-object action, the total number of possible interactions with these objects is $3 \times 83 + 1 \times (83 \times 83) = 7138$. As we aim for a systematic analysis of the system behavior with large number of learning runs, we would like to collect and store the dataset

[1]The robot system is composed of a 7 DOF Kuka Light Weight Robot (LWR) arm placed on a vertical bar similar to human arm, a A 7 DOF 3 fingered Schunk gripper mounted to the robot arm, and a Kinect sensor placed over the 'torso' with a view of the table in front of the robot. Because the main focus of this paper is not on physical execution of the robot, we will omit the details concerning the robot setup.

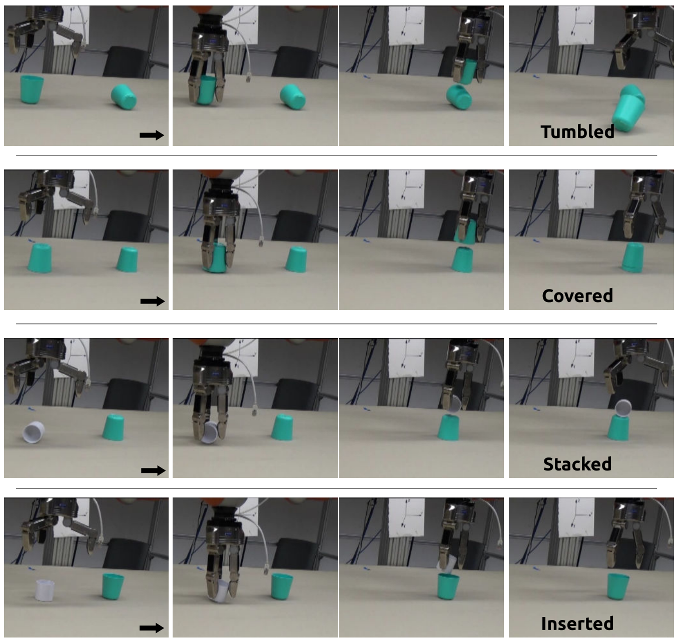


Fig. 6. Sample interactions observed during *stack* action execution on different object pairs.

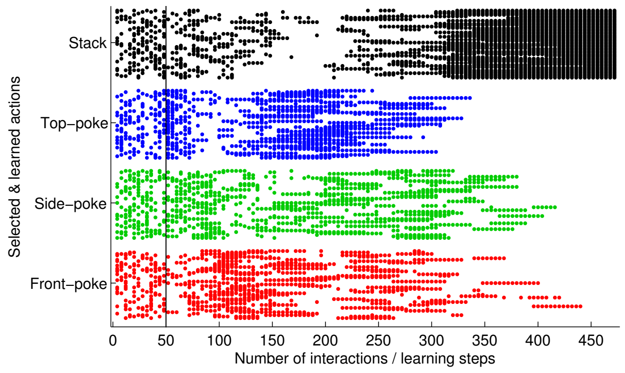


Fig. 7. The action selected for exploration and learning in each iteration of online learning of affordances. As shown, more complex actions, e.g. *stack*, are learned later based on Intrinsic Motivation selection mechanism. This might be due to the fact that learning *stack* affordance is more difficult compared to others, and *stack* prediction of the effect of stack action requires learning of high-level attributes, i.e. single-object affordances, of both objects. Note that before the time-point shown with vertical line, the system is in its random exploration phase, i.e. randomly selects the actions to learn.

in the beginning, and use the interaction samples whenever necessary. On the other hand, making over 7000 interactions is not feasible in the real world. It is also not critical to validate our method as our method is designed as a general approach for inter-dependent affordance learning rather than solving specific affordance problems in real-world settings. Therefore, we used a human expert, who observed the actions of the robot on different objects, and generalized his observations to other objects in order to manually fill-up the effect fields of the interactions [2].

4) Action effects: The effect of an action on a single object depends on various properties of the object being interacted. The same poke action generates different effects on different objects or even on the same objects in different orientations. For example, when poked from side, a lying cylinder will roll away, an upright thick cylinder will be pushed, an upright thin cylinder will topple down, and a lying hollow cylinder will stay still if the robot finger would go through the hole. When paired-object actions are considered, the effect depends on the properties of both objects and the relations between these properties. For example, while an 'inserted-in' effect is generated when a small cylinder is stacked on an upright bigger hollow cylinder, a 'tumbled' effect is observed if the big cylinder is not upright. Based on our previous work where effect categories were self-discovered in single-object actions[12] and in paired-object actions[16], and based on our observations obtained from the sample real-robot executions mentioned in the previous paragraph, the following effect categories are defined:

- Poke-effects: {pushed, rolled, toppled, resisted, nothing}
- Stack-effects: {stacked, inserted, covered, tumbled}

V. Experiment Results

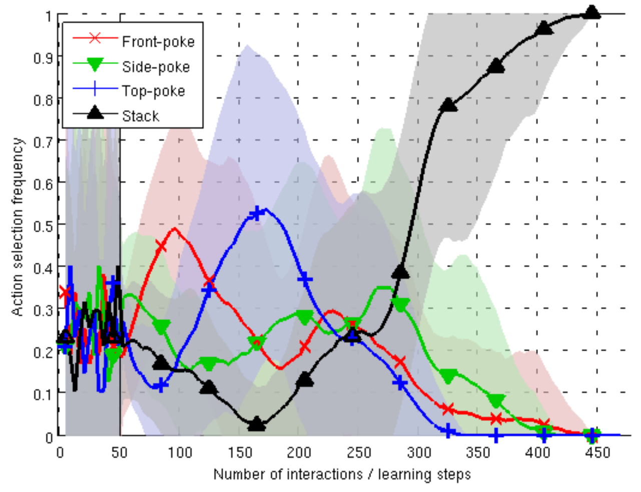
Using the database of 83 objects, 4 actions, and their corresponding effects, we applied active learning of affordances method to discover the learning order (Section V-A) and prediction structure (Section V-B) of the affordance learning system. Furthermore, we verified that with the structure emerged, the learning of complex affordances significantly speeds up especially in the beginning of the learning trials (Section V-C).

In order to statistically analyze the system, we generated 50 separate learning runs that start with random seeds. In other words, our analysis provides the results from 50 different affordance prediction structures that are trained with the same method but with different random seeds. These seeds affect action and object selection in different ways. First of all, in each run, the objects and actions are shuffled to avoid the effect of position of the elements in the lists. Next, ϵ-greedy action selection strategy, random channel selection in object distance calculation, and finally random object selection all use random selection from uniform distributions.

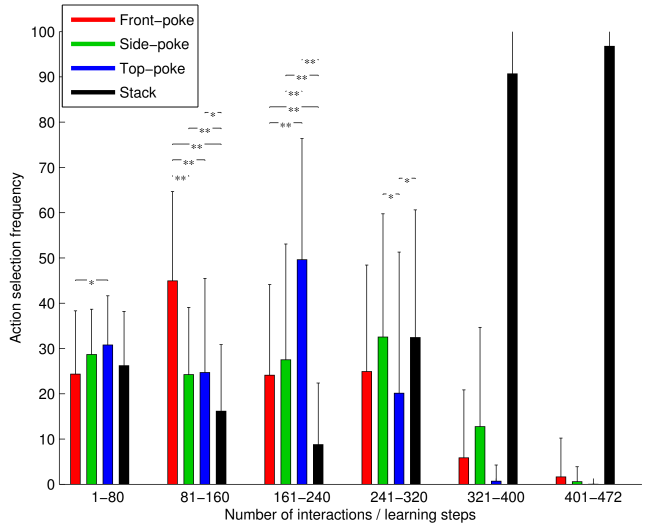
A. Discovered exploration and learning order

This section provides the obtained learning order of the affordance predictors. The learning order of predictors can be analyzed by examining the order and frequency of the corresponding actions that are selected during each iteration of the online learning of the complete system. The selected action for exploration in each learning step is shown in Fig. 7. As the

[2]Predicting the effects of actions on different objects without any reference to the real world performance of the robot has the risk of creating a human-biased interaction dataset. In order to reduce this risk, the human physically 'simulated' the actions on objects when it was difficult to assess the effect. For example, it is difficult to predict whether containers of similar sizes would be inserted, stacked or tumbled, when one is released on top of another one. To disambiguate these situations, the human expert dropped the object physically from some height with a small offset, and stored the generated effects. While this approach should be enough to collect the interaction dataset for our purpose, if one aims to verify a real-robot solution, the uncertainty in action level should be better addressed by the actual execution of the actions.



(a)



(b)

Fig. 8. The number of times each action is selected in each learning step. (a) The lines and shades correspond to mean and standard deviation of 50 runs, respectively. (b) The learning episodes are divided into 6 portions, and statistical analysis on action selection is performed across 80 samples $\times$ 50 runs. Means and standard deviations are provided by the bars and lines over bars. $*$ and $**$ correspond to significance levels of $p < 0.05$ and $p < 0.01$, respectively.

effect of paired-object actions depend on the relations between the properties of two objects, stack is a more complicated action, difficult to learn. As shown, the less complex poke actions are learned first, and more complex stack action is learned later. Prediction of the stack action can also benefit from simple-affordances (as we will show in the next section). Thus, stack action is explored and learned automatically after all other simpler actions are explored. In the figure, the stack action is observed to be explored also in the beginning of the learning in a number of steps either because of momentarily increases in local accuracy or due to the ϵ-greedy strategy. Fig. 8 provides a more detailed statistical analysis of the action selection strategy of the system, where the number of times each action is selected in each learning step is provided in mean and standard deviation. As shown, after the random exploration phase, the system quickly loses its focus on stack action. There are two visible peaks in exploration: first, front-poke action and then top-poke action are explored and learned with visible peaks in frequency. Side-poke is also explored more compared to stack action in the beginning. Only towards end (275 interactions) exploration of the stack action takes lead while others are not learned anymore. This figure clearly shows that the system shifts its focus to stack action after learning all other simpler actions, however there was no such significant difference in learning order in between side-poke and stack actions. This requires further analysis of the behavior of the system.

We plotted the local prediction accuracy γ evolution of each action in Fig. 9. As shown, the learning progresses of poke actions are high initially. After around 100 interactions, the learning progress of side-poke and front-poke slows down, however the system continues learning from top-poke action. The learning progress for side-poke and front-poke starts increasing again after 200 interactions, while the learning progress of stack action is lower in general. After all actions are learned sufficiently and there is no further progress, the learning progress of stack action increases.

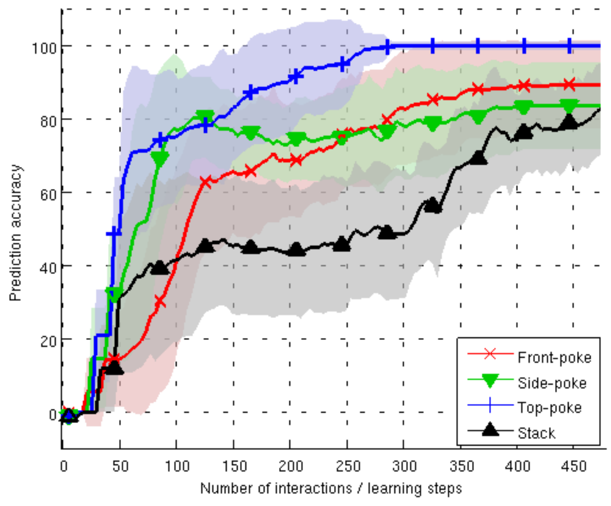
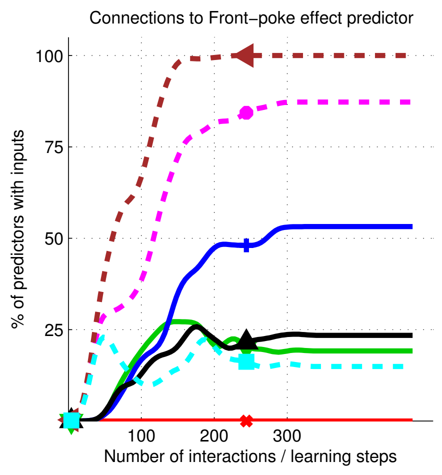


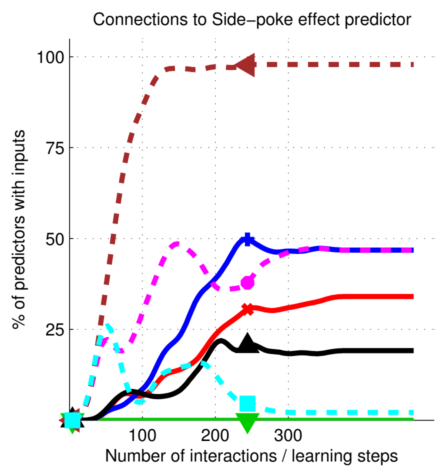
Fig. 9. The evolution of prediction accuracy of each predictor during online learning. The lines and shades correspond to mean and standard deviation of 50 different runs, respectively.

B. Discovered affordance prediction structure

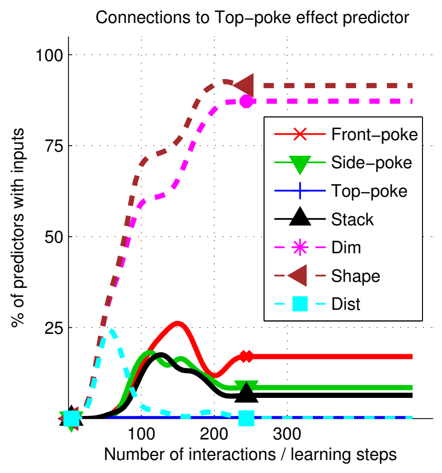
This section gives the results of the structure evolution of the affordance prediction system. Recall that the prediction structure is defined over the most distinctive inputs that are discovered to be most effective in predicting affordances. The ratio of the types of distinctive features used in prediction in different phases of the active learning are shown in Fig.10. Each plot in this figure corresponds to the evolution of the used features and affordances for a different action. One can notice the following emergent structuring properties and tendencies:



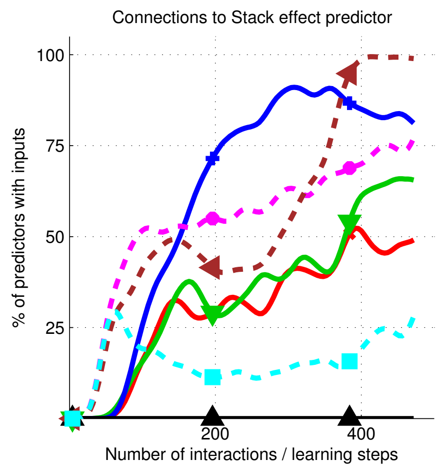
(a) Links to front-poke action predictor



(b) Links to side-poke action predictor



(c) Links to top-poke action predictor

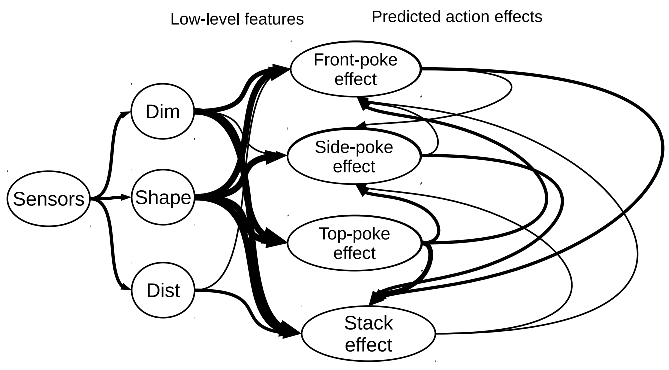


(d) Links to stack action predictor

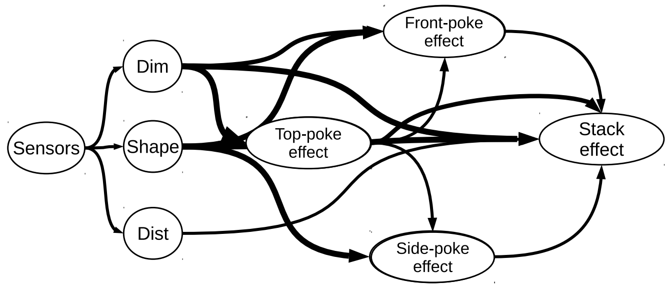
Fig. 10. Evolution of the distinctive features for prediction, where dashed lines correspond distance, shape and dimension features, and solid lines correspond to the predicted affordances. The percentages are computed across 50 runs.

- Shape and dim low-level features are used by almost the predictors in order to predict action effects. Shape features, i.e. distribution of the normal vectors, are probably important in distinguishing rollable objects from pushable objects. Dimension related properties are also important in order to differentiate objects that are toppled by front- or side-poke actions. It is peculiar that dim feature is also found to be important and used for prediction of top-poke action effect because size of the objects should have no influence on how they are affected from top-poke action. With a closer inspection, we found out that walls of some hollow objects were not detected by the perception system, and these hollow objects were perceived as very thin objects.
- Dist low-level feature was originally designed to capture the characteristics related to hollowness of the objects. However, probably shape features capture convexity and concavity good enough, therefore poke action predictors did not heavily use dist features.
- Most of the front-poke, side-poke and stack action predictors used predictions of top-poke action predictor. Top-poke predictor would predict if the object is hollow or not, therefore we were expecting that it would be used for predicting stack effects. Top-poke predictor also encodes rollability to different directions, probably that was why it was used by other poke action predictions.
- Around 25% of the front-poke predictors used side-poke and stack predictions, and around 25% of the side-poke predictors used front-poke and stack predictions. This result shows that no strict hierarchy appeared between front-poke and side-poke effect predictors. On the other hand, around 50% of stack action predictors used predictions from front-poke and side-poke predictors. We can suggest that differences in maturation order of different predictors caused differences in different structuring strategies.

For illustrative purposes, we weighted directed links in between predictors based on the frequencies provided in Fig. 10, and obtained the connectivity graph provided in Fig. 11(a). As shown in the plots, each low-level feature affects affordance predictions in different levels, and shape features are observed to be the first discovered distinctive features, i.e. selected input



(a) Connectivity graph with weighted links



(b) Connectivity graph when loose links are removed

Fig. 11. Discovered structure, i.e. input and output connections.

connections, especially in the initial phases of development for all actions. However, more important in the context of this paper, affordances are observed not to be used in the initial phases, and all of them are only found to be used in predicting effect of stack action, i.e. predicting stackability affordances. Note that stack predictor starts using single-object affordances after around 30 samples, probably because the single-object affordance prediction was not good enough before that time-point.

For illustration purpose, we removed the links that have weights below 25%. In other words, we removed the links if those links exist in less than 25% of the learning runs. When the graph is re-arranged, a loose but clear hierarchical prediction structure emerges as shown in Fig. 11(b).

C. Bootstrapping effect in stack learning

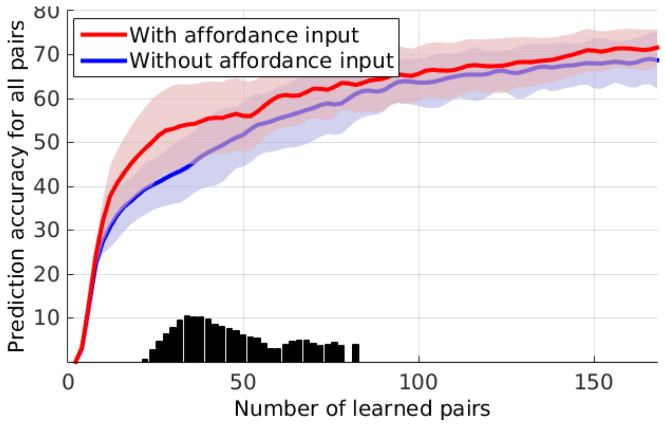
In this section, we analyzed the bootstrapping effect that is achieved by discovering use of single-object affordances for predicting paired-object affordances. In order to analyze this effect, we compared the learning speed with and without use of inputs from other effect predictions in predicting stack effects. The system, where effect predictions of other actions along with low-level features are used, is called 'with-affordance-input'. The system where only low-level features are used in learning and predicting effects of stack actions will be called 'without-affordance-input'.

When the learning speed of stack affordances is compared with- and without-affordance-inputs in an active learning setting where all affordances are learned as above, we noticed no significant difference between learning speeds. We argue that the bootstrapping is generally achieved in the beginning of learning the bootstrapped systems are shown to make better predictions with even small number of learned interactions. As soon as the number of samples get higher, the bootstrapping effect is reduced. In our system, stack predictor is trained along with other predictors, and in the initial phases of its training, it does not receive correct inputs from other predictors; therefore we could not achieve such bootstrapping effect.

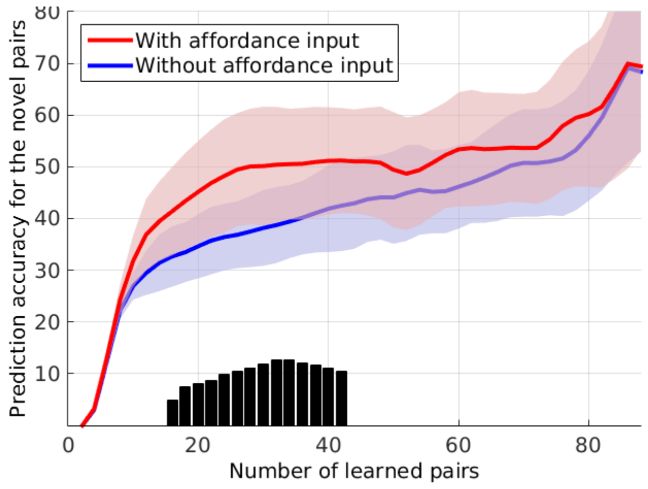
Next we analyzed the case, where stack predictor learning starts after learning all poke predictors. Here, we assume here that the system learned the learning order and structure of affordance predictors, and transfer the learned knowledge for new objects. Therefore, in this setting, the predicting poke actions is learned first, and the outputs of the learned predictors are used as potential inputs. The predictions from poke actions are also used in active object selection in order to compute the distance between objects and to maximize the diversity in training set. We again performed 50 learning runs both with- and without-affordance-inputs in learning stack effect predictions. In this setting, we can see a significant bootstrapping effect in learning as shown in Figs. 12(a) and 12(b). Mean and standard deviation of the group of trained predictors with- or without-affordance-inputs are shown with the bold line and the filled areas. As shown in the figures, initial and final mean accuracies and variances of two different cases are same. Still, the expected bootstrapping effect in the beginning of the learning steps are clearly visible in both figures. At the bottom of each figure, the difference in accuracies between with- and without-affordance-inputs is given, which is confirmed with double-side t-test with $p < 0.01$. The boxes at the bottom in Fig. 12(a) show that the accuracy of with-affordance-inputs case becomes 10% higher with 40 training samples, compared to without-affordance-inputs case with significance level of $p < 0.01$. As shown, the bootstrapping effect quickly increases in the beginning and remains same for some time. The bootstrapping effect is more significant when the accuracy is computed with novel pairs. Novel pairs of objects $\{(o_1, o_2)\}$ for stack action corresponds to objects, where $\{o_1\}$ and $\{o_2\}$ have never been experienced in bottom and top roles during stacking, respectively. The effect becomes visible after 18 pairs instead of 22 pairs; and increases to a maximum value of 13% compared to 10%. This also shows that use of affordances as inputs in learning and predicting other affordances provides significant generalization capabilities.

VI. Discussion

In this paper, the action and effect categories are known by the system. From a developmental point of view, these categories should be learned by the robot as it is not possible to predict and design the categorization that is suitable for the embodiment of the robot especially for unbounded set of complex actions. Therefore, we believe that it is necessary to discuss how such categorization can be achieved by interacting systems. In this section, we will summarize our previous



(a) Test with all pairs.



(b) Test with novel pairs.

Fig. 12. The evolution in prediction accuracy of stack effect predictor. The accuracy is computed by using all object pairs in (a) and by using only the novel object pairs in (b). The black bars show the amount of difference in prediction accuracies between learning systems with and without affordance inputs, in significance level of $p < 0.01$.

studies where categorization in action and effect spaces were self-discovered by a manipulator robot.

a) Action categorization: We previously showed that a robot that was initialized with a basic reach- and enclose-on-contact movement capability, can discover a set of action primitives by exploring its movement parameter space [38, 11]. In that work, the robot was assumed to have only two basic movement mechanisms: a basic finger enclose behavior akin to infants palmar grasp reflex, and one basic arm action, that generates simple arm movements to transport the hand to the vicinity of an object. The contact information, sensed during the approach and possibly after the finger enclosure, is used to cluster the executed movements, yielding a set of action primitives such as push, 'release', and grasp. In order to learn more complex actions, such as 'move-object-over-another', we realized an imitation learning strategy where the robot observed and encoded complex demonstrations into a series of previously learned primitives with the help of a cooperative tutor [11]. In that work, inspired from infant development and motionese framework [39], we equipped the robot with mechanisms that can detect motion related cues such as pauses in order to more easily segment the demonstrated behavior into chunks that can be replicated by previously discovered action primitives. We argue that simple action categories can be discovered through exploration, and more complex actions that are composed of simple categories can be learned through imitation learning.

b) Effect categorization: Effect categories are generally obtained in an unsupervised way by applying clustering methods to continuous effect space. We previously discussed that such clustering is sensitive to relative weighting of the effect features that are encoded in different units and channels. Furthermore, discovered effect categories should be instrumental for the latter predictions and reasoning; and completely unsupervised clustering does not provide any guarantee for this. Therefore, we proposed a 2-level clustering algorithm that takes into account the representational differences between different perceptual channels and utilizes a verification step that makes sure that the discovered effect categories can be predicted by the robot [38, 12]. In detail, in the lower level, channel-specific effect categories are found by clustering in the space of each channel, discovering separate categories for channels such as touch, position and shape. In order to ensure the predictability of the channel-specific effect categories, classifiers are trained. If a channel-specific effect category is found not to be predictable, the clustering is re-done with less number of desired clusters. After finding the final channel-specific effect categories, in the upper level these categories are combined to obtain all-channel effect categories using the Cartesian product operation. The final categories are only accepted if they are predictable. The discovered effect categories for push action were 'disappeared=rolled', 'grasped&disappeared', 'grasped', and 'pushed'. Similar ideas were also used by others, where categorization was also based on the ability to predict the outcome of action execution in Mugan and Kuipers [40], and categories were utilized if they appear in the learned rules in Pasula et al. [41]

With increasing complexity of actions, finding effect categories becomes more difficult. In finding effect categories of stack action for example, we observed that a naive clustering in continuous effect space was not effective as the interacting objects can generate various effects difficult to separate due to the complex interactions between them [16]. In that case, applying further exploratory actions on the objects after stacking helped the system to handle this complexity. For example, after a stack action, if the robot poked both of the objects one by one, effect categories were found to be distinguishable when observations obtained from the following poke actions were also taken into account. Some discovered effect categories for stack action were 'stacked', 'inserted-in', and 'tumbled' at the end.

VII. Conclusion

In this paper, we studied how interdependent affordance learning tasks can be autonomously structured along with the learning order of its components. In an online learning framework, we showed that intrinsic motivation mechanism, which select the next action to explore based on learning progress of the model of that action, can discover such a learning order where paired-object affordance learning follows maturation of single-object affordances learning. Next, we showed that by using the most discriminative features for affordance prediction, the expected hierarchical structure emerged autonomously where the learning system discovered that predictions of the single-object affordances are connected to the paired-object affordances. We validated our approach in a real dataset composed of 83 objects and pairs of these objects along with the effects of three poke actions and one stack action. The results show that hierarchical structure and learning order emerged from the learning dynamics that is guided by Intrinsic Motivation mechanisms and feature selection approach.

The results further show that a bootstrapping effect in learning speed of affordances could be achieved with such a hierarchical structure. This was probably achieved as simple affordances, which were used as inputs to complex affordances, provide more abstract and generalizable knowledge compared to low-level visual features. Bootstrapping effect was visible when generalization was required from learning of small number of samples, and disappeared in the latter steps as the system could learn equally well from low-level features with large number of training samples. Bootstrapping effect was also more significant in more difficult tasks such as when the predictors were tested in novel situations. In our system, the learning order of affordances was not strict, and determined based on the IM criteria with an approximate measure of future learning progress. Therefore, the hierarchical structure only appeared after initialization of learning progress through some exploration of all actions, and the initial bootstrapping effect was not observed in this emergent structuring setting. Other factors such as motor complexity of actions are also important in deciding which action to explore first, and can be used to achieve a more strict learning order, and therefore more significant bootstrapping effect.

In our framework, the system forms forward models [42] that enable it to predict the changes in the environment in terms of discrete effect categories. Generative models have been shown to be effective in learning (object, action, effect) relations and in making inferences on any element of the these relations. For example, they can infer the required action given the desired (object, effect) pair or they can predict the effect given (action, objects) pair [13]. We discuss that our 'discriminative' model still provides powerful mechanisms as it can effectively map the continuous object feature and behavior parameter spaces to the corresponding effects [43] possibly using complex non-linear functions without any initial categorization of object properties as in [13, 17]. Furthermore, while bi-directional relations are not explicitly encoded in our system, we showed that our robot was able to make predictions in different directions, and made plans that involved sequence of actions on automatically selected objects[12].

Our proposed framework, which exploits the idea of active selection of actions, objects, and connections, is very general and can be directly applied for a wide-variety of robot learning tasks and to different robotic platforms. It can also be extended to continuous actions where action parameters such as poke direction are used as inputs of the predictors, and to continuous action effects where powerful regressors are used for prediction.

ACKNOWLEDGEMENTS

This research was supported by European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreements no. 270273, Xperience, and no. 610532, Squirrel.

REFERENCES

[1] M. Lopes, P.-Y. Oudeyer *et al.*, "Guest editorial active learning and intrinsically motivated exploration in robots: Advances and challenges," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 2, pp. 65–69, 2010.

[2] P.-Y. Oudeyer and F. Kaplan, "What is intrinsic motivation? a typology of computational approaches," *Frontiers in neurorobotics*, vol. 1, pp. 1–6, 2007.

[3] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 265–286, 2007.

[4] A. Baranes and P.-Y. Oudeyer, "Robust intrinsically motivated exploration and active learning," in *Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on*. IEEE, 2009, pp. 1–6.

[5] S. Ivaldi, S. Nguyen, N. Lyubova, A. Droniou, V. Padois, D. Filliat, P.-Y. Oudeyer, and O. Sigaud, "Object learning through active exploration," *IEEE Transactions on Autonomous Mental Development*, 2013, accepted.

[6] E. J. Gibson, "Perceptual learning in development: Some basic concepts," *Ecological Psychology*, vol. 12, no. 4, pp. 295–302, 2000.

[7] J. J. Gibson, *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, 1986.

[8] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *International Journal of Robotics Research (IJRR)*, vol. 32, no. 8, pp. 951–970, 2013.

[9] A. Myers, C. L. Teo, C. Fermuller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[10] M. Schoeler and F. Worgotter, "Bootstrapping the semantics of tools: Affordance analysis of real world objects on a per-part basis," *IEEE Transactions on Autonomous Mental Development*, 2015, accepted, 10.1109/TAMD.2015.2488284.

[11] E. Ugur, Y. Nagai, E. Sahin, and E. Oztop, "Staged development of robot skills: Behavior formation, affordance learning and imitation with motionese," *IEEE Transactions on Autonomous Mental Development (TAMD)*, vol. 7, no. 2, pp. 119–139, 2015.

[12] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Robotics and Autonomous Systems*, vol. 59, no. 7–8, pp. 580–595, 2011.

[13] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory–motor maps to imitation," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 15–26, 2008.

[14] J. Sinapov, C. Schenck, K. Staley, and A. Stoytchev, "Grounding semantic categories in behavioral interactions: Experiments with 100 objects," *Robotics and Autonomous Systems*, vol. 62, no. 5, 2014.

[15] S. Griffith, J. Sinapov, V. Sukhoy, and A. Stoytchev, "A behavior-grounded approach to forming object categories: Separating containers from noncontainers," *IEEE Transactions on Autonomous Mental Development*, vol. 4, pp. 54–69, 2012.

[16] E. Ugur and J. Piater, "Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[17] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4373–4378.

[18] B. Moldovan and L. De Raedt, "Learning relational affordance models for two-arm robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Chicago, IL, USA: IEEE, 2014, pp. 2916–2922.

[19] S. Fichtl, D. Kraft, N. Kruger, and F. Guerin, "Bootstrapping the learning of means-end action preconditions," *IEEE Transactions on Autonomous Mental Development (TAMD)*, 2015, submitted.

[20] H. F. Harlow, M. K. Harlow, and D. R. Meyer, "Learning motivated by a manipulation drive." *Journal of Experimental Psychology*, vol. 40, no. 2, p. 228, 1950.

[21] G. Baldassarre, T. Stafford, M. Mirolli, P. Redgrave, R. M. Ryan, and A. Barto, "Intrinsic motivations and open-ended development in animals, humans, and robots: an overview," *Frontiers in psychology*, vol. 5, 2014.

[22] G. Baldassarre and M. Mirolli, *Intrinsically motivated learning systems: an overview*. Springer, 2013.

[23] J. Law, P. Shaw, K. Earland, M. Sheldon, and M. Lee, "A psychology based approach for longitudinal development in cognitive robotics," *Frontiers in neurorobotics*, vol. 8, 2014.

[24] S. M. Nguyen and P.-Y. Oudeyer, "Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation," *Paladyn Journal of Behavioural Robotics*, vol. 3, no. 3, pp. 136–146, 2012.

[25] S. Hart and R. Grupen, "Intrinsically motivated affordance learning," in *Workshop on Approaches to Sensorimotor Learning on Humanoids, IEEE Conference on Robotics and Automation (ICRA)*, 2009.

[26] S. Hart and R. Grpen, "Learning generalizable control programs," *IEEE Transactions on Autonomous Mental Development (TAMD)*, vol. 3, no. 3, pp. 216–231, 2011.

[27] J. Piaget, *The Origins of Intelligence in Children*. New York, USA: International University Press, 1952.

[28] A. G. Barto, S. Singh, and N. Chentanez, "Intrinsically motivated learning of hierarchical collections of skills," in *Proceedings of the 3rd International Conference on Development and Learning (ICDL 2004)*, 2004, pp. 112–19.

[29] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robotics and Autonomous Systems*, vol. 61, no. 1, pp. 49–73, 2013.

[30] E. Ugur and J. Piater, "Emergent structuring of interdependent affordance learning tasks," in *IEEE Intl. Conf. on Development and Learning and on Epigenetic Robotics (ICDL-Epirob)*, 2014.

[31] E. Ugur, S. Szedmak, and J. Piater, "Bootstrapping paired-object affordance learning with learned single-affordance features," in *IEEE Intl. Conf. on Development and Learning and on Epigenetic Robotics (ICDL-Epirob)*, 2014.

[32] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, September 1998.

[33] C.-C. Chang and C.-J. Lin, "Libsvm : a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 27, pp. 1–27, 2011.

[34] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *The Journal of Machine Learning Research*, vol. 6, pp. 1579–1619, 2005.

[35] R. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.

[36] A. W. Moore and M. S. Lee, "Efficient algorithms for minimizing cross validation error," in *Proceedings of the 11th International Conference on Machine Learning*, R. Greiner and D. Schuurmans, Eds. Morgan Kaufmann, 1994, pp. 190–198.

[37] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Point cloud library," *IEEE Robotics & Automation Magazine*, vol. 1070, no. 9932/12, 2012.

[38] E. Ugur, E. Sahin, and E. Oztop, "Self-discovery of motor primitives and learning grasp affordances," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3260–3267.

[39] R. J. Brand, W. L. Shallcross, M. G. Sabatos, and K. P. Massie, "Fine-Grained Analysis of Motionese: Eye Gaze, Object Exchanges, and Action Units in Infant-Versus Adult-Directed Action," *Infancy*, vol. 11, no. 2, pp. 203–214, 2007.

[40] J. Mugan and B. Kuipers, “Autonomous learning of high-level states and actions in continuous environments,” *Autonomous Mental Development, IEEE Transactions on*, vol. 4, no. 1, pp. 70–86, 2012.

[41] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, “Learning symbolic models of stochastic domains,” *Journal of Artificial Intelligence Research*, pp. 309–352, 2007.

[42] M. Haruno, D. Wolpert, and M. Kawato, “Hierarchical mosaic for movement generation,” in *Excepta Medica International Coungress Series*. Amsterdam, The Netherlands: Elsevier Science B.V., 2003, pp. 575–590.

[43] E. Ugur, E. Oztop, and E. Sahin, “Going beyond the perception of affordances: Learning how to actualize them through behavioral parameters,” in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4768–4773.