SEVENTH FRAMEWORK PROGRAMME

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

XPERIENCE.ORG

Deliverable Number:	D5.2.5
Deliverable Title:	Final Demonstration of Scenario 1
Type (Internal, Restricted, Public):	PU
Authors:	Mikkel Tang Thomsen, Dirk Kraft, Norbert Krüger, Emre Ugur, Justus Piater, Andrej Gams, Tadej Petrič, Aleš Ude, Giorgio Metta, David Schiebener, Tamim 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

In the last year of the Xperience project, we have merged scenario 1 and 2 into one major demo on ARMAR at KIT. This demonstration is described in D5.3.5. We describe the contributions of WP5.2 to this main demo in chapter 2. Moreover, we demonstrate structural bootstrapping in the full Xperience cycle on a robot platform at UIBK. This is described in chapter 3. In addition, we show a number of demonstrations performed in individual labs in chapter 4.

Chapter 2

Contributions to Main Demo

In this section, we describe the WP5.2 contributions to the main bootstrapping demo as outlined in the PPR in WP5.3 and in D5.3.5.

2.1 Segmentation and reactive grasping of unknown objects

The video "**Pushing-for-Segementation-and-Grasping-Armar-IIIb-final_render_0-x4.mpg**" demonstrates the integration of several skills that were developed within the Xperience project: Segmentation of unknown objects by physical interaction, their pre-grasp manipulation, and finally grasping employing reactive correction strategies based on haptic and visual perception.

The humanoid robot ARMAR-IIIb observes a scene consisting of unknown objects, and creates initial object hypotheses based on its visual perception. One of the object hypotheses is pushed, and the resulting rigid body motion is used to verify and improve the hypothesis, leading to a reliable segmentation even in cluttered scenes. The segmentation is used to estimate the objects' extent. By target-oriented pushing, it is brought to a position and orientation that facilitates grasping by orienting the shorter side of the object towards the expected approach direction of the hand.

Grasping is realized in a reactive manner. The robot approaches the object with an opened hand. If during the approach a collision with the fingers is detected by tactile, proprioceptive or visual feedback, the hand position and orientation are corrected to avoid this collision. This is repeated until the object is reached or contact is detected by tactile sensors in the palm of the hand. When this happens, the fingers are closed to conclude the grasp and the object is lifted. Based on haptic and proprioceptive feedback, the grasp success is verified, and in case of failure another attempt is initiated. See Figure 2.1 for an overview of this approach.

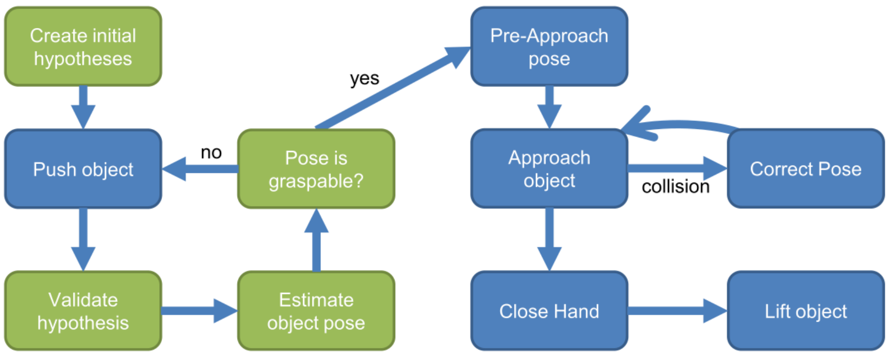


Figure 2.1: Overview of the object segmentation and grasping approach: An unknown object is segmented in clutter, using its motion that is caused by the robot's pushes. It is pushed to a favorable position and orientation, and then grasped in a reactive manner. Premature collisions of hand and object are detected using haptic and visual cues, and the hand pose is reactively corrected until the grasp succeeds.

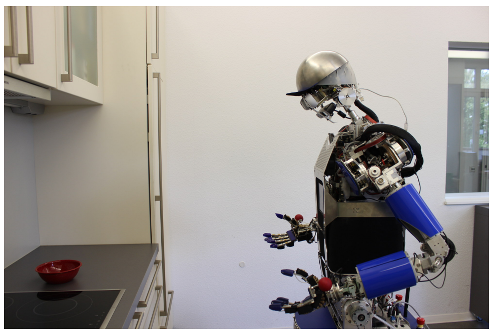
2.2 The PKS planner

High-level plans in Scenario 1 are generated by PKS (Planning with Knowledge and Sensing), the knowledge-level planner that UEDIN is developing as part of WP4. Details of the planner's operation have previously been reported in deliverables D3.2.1, D3.2.2, D3.2.3, and D3.2.4. The same planner is also being used as part of Scenario 2 on the ARMAR robot platform. Specific details about the PKS planning domain in Scenario 1 were previously described in D5.2.4.

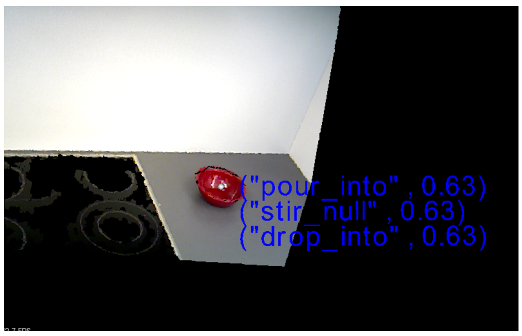
2.3 Action replacement using histograms and Joint SVM

Based on the relational histogram representation developed in Xperience, we apply a Joint SVM learning algorithm to associate affordances to objects. The sub-module is used in the main Xperience demo for object replacement in a complex planning problem, in which an object that is requested for executing the plan is not available and an alternative objects is sought for. The joint SVM—besides giving an actual affordance classification—also provides a confidence value that indicates the likelihood of a correct classification—that can be exploited in the actual choice of objects. Extensive quantifications as well as the integration on the ARMAR platform are described in [6].

The work presented here is part of a system that is capable of creating symbolic plans for a given task. The system has the ability to search for replacements of missing objects based on several replacement-strategies. Replacement based on affordance estimation is one of these strategies. Essentially, the robot creates an assumed memory state from previous experience, which serves as a basis for the planner. If the robot encounters that an object of a plan is missing, the replacement component is consulted for a valid replacement (see Figure 2.2).



(a) ARMAR-III encounters a small bowl while expecting the object in (b).



(b) Point cloud visualization (snapshot) of the scene, showing the predicted affordances of the unexpected object.

Figure 2.2: Affordance estimation in ARMAR III.

Chapter 3

Realization of the Xperience Cycle

The Xperience approach and cognitive architecture (see Figure 3.1) comprises a cycle of four processes: (1) categorization of perception-action dependencies in the form of object-action-effect representations that encode affordances, (2) generative modeling and forming symbolic representations, (3) mechanisms that enable internal simulation over formed symbols, and finally (4) closing the loop by enactive grounding. The key idea is to bootstrap generalized and transferable symbolic knowledge of perception-action associations and dependencies from specific perception-action contingencies of objects, associated actions, and consequent effects, and then to use these generative models in a process of internal simulation to predict the outcome of actions or plan actions to achieve desired outcomes. The results of these internal simulations are then enacted by grounding: validating them by observing the actual result and either adapting the generative model through accommodating new categories or assimilating the new experience to the previously learned model.

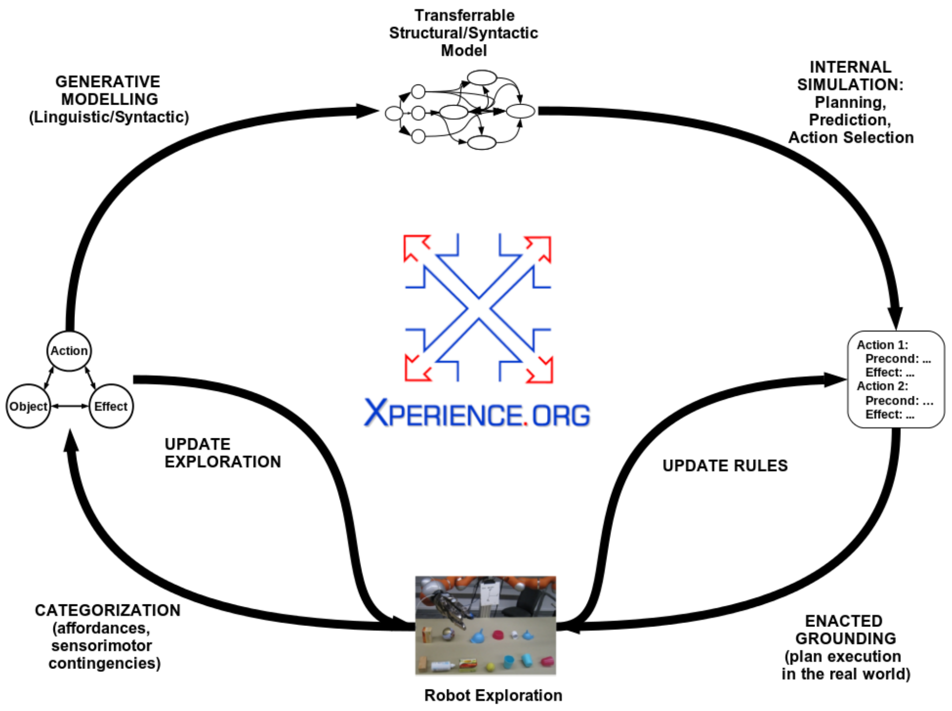

Transferrable Structural/Syntactic Model
GENERATIVE MODELLING (Linguistic/Syntactic)
INTERNAL SIMULATION: Planning, Prediction, Action Selection
Action
Object
Effect
XPERIENCE.ORG
UPDATE EXPLORATION
Action 1: Precond: ... Effect: ...
Action 2: Precond: ... Effect: ...
UPDATE RULES
CATEGORIZATION (affordances, sensorimotor contingencies)
ENACTED GROUNDING (plan execution in the real world)
Robot Exploration

Figure 3.1: Xperience learning cycle.

The video "**UIBK-Xperience-cycle-v3.mpg**" describes our effort to realize and implement the conceptual learning cycle summarized above. This cycle defines learning as a process that goes all the way from low-level sensorimotor exploration to symbolic planning. It generates progressively more complex and higher-level skills, abstractions and reasoning capabilities that are validated through further interactions.

The formed models also should influence the way the robot explores its environment in a more intelligent way.

The video first shows how the real and simulated robot explores its environment with the actions that involve single objects and pairs of objects, namely poke, grasp, release and stack actions.

- In the CATEGORIZATION stage, the robot monitors the consequences of its actions, and forms discrete effect and object categories from its continuous sensorimotor experience. Action effects such as rolled, tumbled, stacked, and inserted are discovered through unsupervised clustering. Object categories such as hollow, flat, round, and unstable are discovered by grouping objects that behavior similar in response to different actions. Finally, the relations between visual properties of objects and object categories are learned, providing affordance perception capability.
- The UPDATE EXPLORATION stage shows how learned categorization changes the way the robot explores its environment with two complementary active learning mechanisms. The first mechanism exploits the idea of intrinsic motivation, and enables the robot to automatically focus on easy-to-learn actions, and learn poking before the complex stacking action. The second mechanism enables the robot to detect and explore maximally different object categories, providing significant speed-up in learning [10].
- The GENERATIVE MODELING stage shows the hierarchical affordance learning structure that is autonomously constructed by the system. For this, the system discovers that re-using simple affordances (such as rollability) in learning complex affordances (such as stackability) speeds up learning. Because the experienced sensorimotor data is encoded with discovered symbols, the relations between this data, i.e. object and effect categories, can also be represented in logical form. For this, in the same stage, the system trains decision trees and learns logical rules that encode consequences of actions. This enables the system to reason on the symbolic level in the next stage.
- In the INTERNAL SIMULATION stage, these rules are automatically transferred to the STRIPS style Problem and Domain Definition Language (PDDL) format. Domain description includes a separate action for each learned rule along with the corresponding preconditions and effects. The initial state of the world, i.e. object categories and discrete relations between the objects, and the goal is defined in the problem description, in STRIPS notation as well. Given domain and problem descriptions, off-the-shelf symbolic planners are used to achieve the tasks [11].
- In the ENACTED GROUNDING stage, the validity of the learned symbols and operators are tested by executing the sequence of actions planned to achieve a real task such as building towers of different heights with different objects. While executing the plan, the robot encounters completely novel situations that cannot be perceived or reasoned about with the knowledge that the robot acquired in previous learning. Therefore, based on the observations from multi-sequence action executions, the robot either updates the previously learned rules or discovers new categories that appear in plan execution, and by that closing the loop [UP15].

Chapter 4

Individual Contributions

4.1 Accelerated motor learning in constrained domains

The three videos mentioned in this section demonstrate the application of bootstrapping for accelerated motor learning, developed within the Xperience project. The key components of the proposed bootstrapping mechanisms are demonstrated in the videos: autonomous learning of compliant movement primitives, their generalization, and finally learning in the space spanned by the previously acquired examples. Three distinct tasks are presented: learning of joint torques for reaching at different targets, learning of joint torques for accelerated whole-body squatting (both on the CoMan humanoid robot), and learning of joint torques for the execution of peg-in-hole tasks at different locations on KUKA LWR-4 robot. Note that the underlying parametric representation called compliant movement primitives (CMP), which was applied in these experiments, was developed within the EU FP-7 project Learning and Execution of Action Categories (ACAT) [1].

In the first video "**Coman_reaching.mp4**", the humanoid robot CoMan was commanded to reach a desired point in space. Since the robot is compliant, using low gains for trajectory tracking, it could not reach the desired goal configuration without knowing its dynamic model for the generation of the required feedforward torques. These were learnt using recursive regression. The learning of feedforward torques was repeated for a number of different reaching positions until the desired accuracy at the final configuration was achieved for all of them. The experiment of learning feedforward torques for reaching movements was then repeated, but this time using bootstrapping to speed up learning [2, 7]. In this experiment, once the desired accuracy of motion has been reached, the learned compliant movement primitive is added to the database of motion and used to bootstrap learning in the next step using statistical learning methods. As evident in the video, less and less executions are needed to add new samples into the database because the initial approximations become more and more accurate.

In the second video "**Coman_squating.mp4**", the same humanoid robot (CoMan) was commanded to perform squatting motions, but in simulation. The reflexive stability framework for the humanoid robot [3] was integrated into the learning of compliant movement primitives. Again, feedforward torques must be available to implement successful compliant squatting behavior. The video shows learning with and without bootstrapping for initial approximation until the desired accuracy of motion is reached. This video also demonstrates that learning with bootstrapping is much faster because in this case the learned compliant movement primitives are added to the database of motion, contributing to the improvement of initial approximation for the required squatting behavior.

In the last video "**LWR_PiH.mp4**" (see also Figure 4.1), a Kuka LWR-4 robot was commanded to perform a peg-in-hole (PiH) task at different locations. The difference to the learning of simple reaching movements is that in this case, interaction with the environment takes place, which makes the learning problem much more difficult. As in the previous two cases, the video shows learning with and without bootstrapping for initial CMP approximation. Again, adding the learned CMPs to the database and the application of statistical learning methods results in better initial approximations for subsequent PiH actions, which accelerates the learning of the desired behavior.

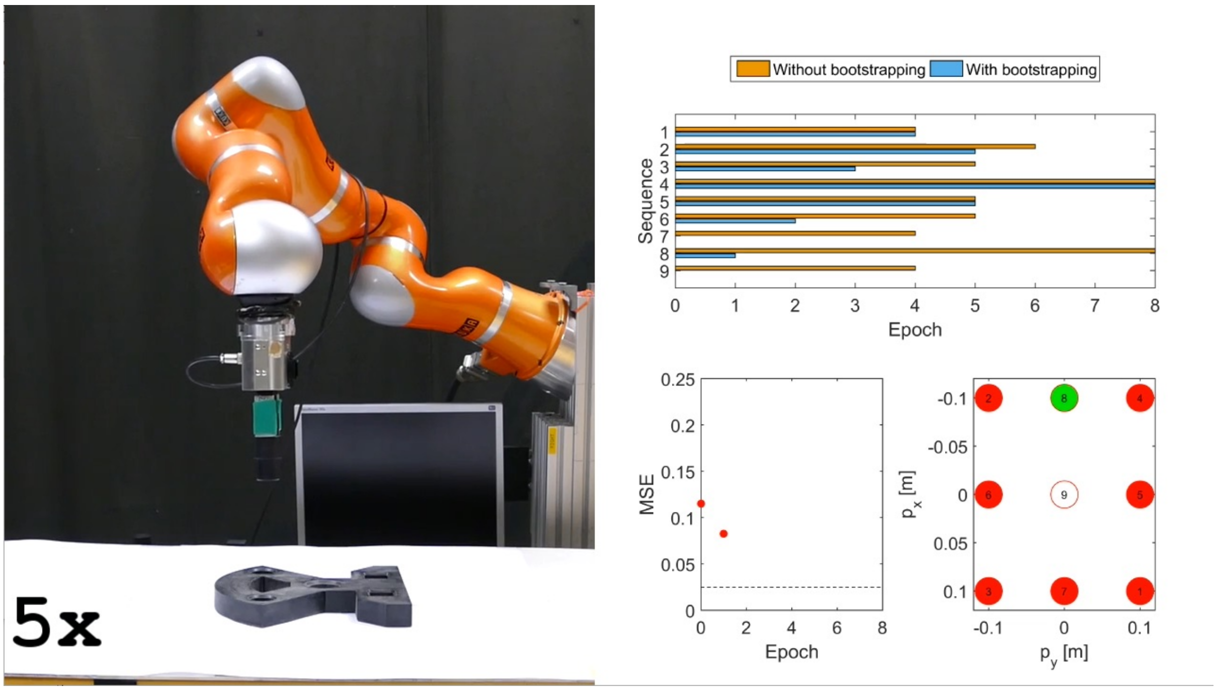


Figure 4.1: Image frame from the video depicting the peg-in-hole learning using bootstrapping mechanisms. The plots on the right-hand side show the number of learning epochs for a given position, defined by the red-green-empty circles in the bottom right. The red circles have finished learning, while the green one depicts learning in progress. In the top plot we can see that fewer learning epochs are needed when the initial estimate is bootstrapped with previous knowledge.

4.2 Self-supervised learning of grasp dependent tool affordances on the iCub humanoid robot

The video "**IIT_affordance_self.mp4**" demonstrates affordance learning and tool use. The main intention is to understand ways in which a robot can learn the correlation between the geometric characteristics of tools and the effect they can achieve upon the execution of certain actions. Such learning can allow the robot to select the best action to perform a desired task with a given tool, or eventually, to select the best tool to use for achieving a goal, even among tools previously unseen. Additionally, the way in which the grasp modulates the required action and the achievable effects has also been tackled.

This experiment, work published in [5], represents the geometric characteristics of tools as a set of 75 features extracted from their observed 2D contour after being grasped. These included features based on the contour's shape, moments, convex hull, skeleton, signatures and domain transform. The goal task was to pull an object closer to the robot, for which the appropriate action for each tool-pose had to be predicted. The experiment involved seven tools in simulation (Figure 4.2left) and four on the real robot (Figure 4.2right), each grasped in three possible ways, (left, front, right, as shown in Figure 4.3). In order to observe the pulling affordance of each of these tools, and the effect of the grasp, the pulling action was performed targeting points located from a few centimeters to the right of the object to a few centimeters to is left (Figure 4.4). The observed effects were clustered using K-means, which returned an index for each interaction trial. Finally, this index was then used as the target signal to train a classifier whose input was the 75 features of the tool used in each corresponding trial. Learning was tested by comparing the predicted effect class for each tool-pose to the actual recorded one, as well as by using the prototype effect vector of the predicted class to select the best action to pull the object. A full diagram of the performed experiment can be observed in Figure 4.5. Results showed that although there is still a lot of room for improvement, this or similar approaches are quite promising.

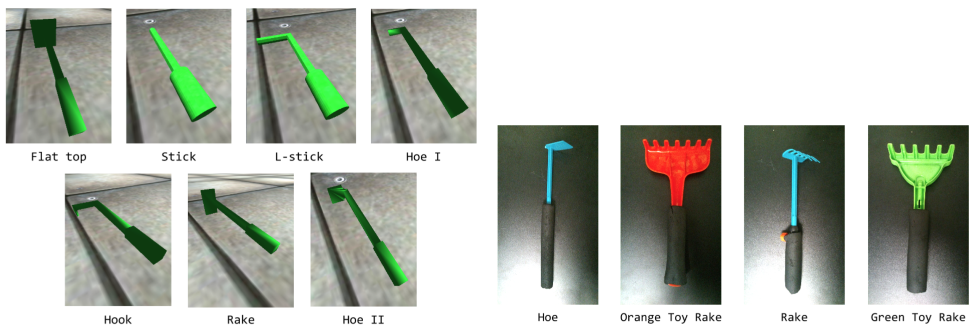


Figure 4.2: Illustration of virtual and real tools used for the affordance experiment.

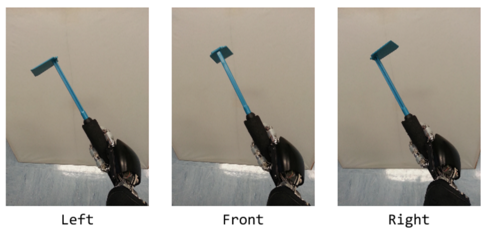


Figure 4.3: Considered tool grasps/ orientations.

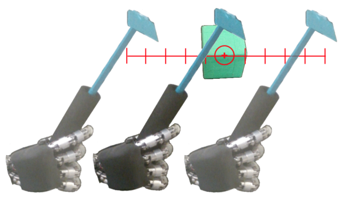


Figure 4.4: Approaches to target object for pull action. Each red mark represents 1 cm.

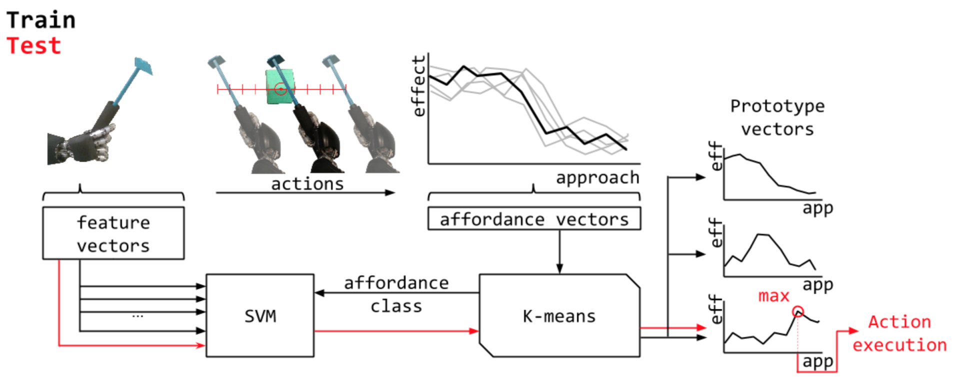


Figure 4.5: Full diagram of the experiment in [5]. Black lines correspond to training flow, while red corresponds to testing.

4.3 Multi-model approach based on 3D functional features for tool affordance learning in robotics

The video "**affordances.mp4**" presents a follow up experiment, published in [4], aimed at improving the approach described in the previous section by, on the one hand, applying features based on the 3D geometry of the tools rather than its contour, and on the other hand, employing actions which render possible the discrimination between more kinds of tools. Action, thus, instead of just pull, consisted in sliding the tool from a position on top of the object to each of the 8 main cardinal directions (i.e. in angles from 0 to 315 degrees on intervals of 45 degrees, as shown in (Figure 4.6).

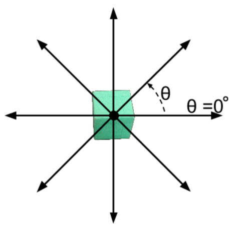


Figure 4.6: Possible slide actions to perform on the target object.

The feature set used to represent tools in this scenario was a concatenation of voxel-based normal spherical histograms, named Oriented Multi-scale Extended Gaussian Images (OMS-EGI for short). In order to represent not only the 3D geometry of the tool, but also the way in which it was grasped, voxels were obtained as octree divisions of the tool's bounding box aligned to the axis of the robot's hand reference frame. A visual representation of the feature extraction process can be seen in Figure 4.7.

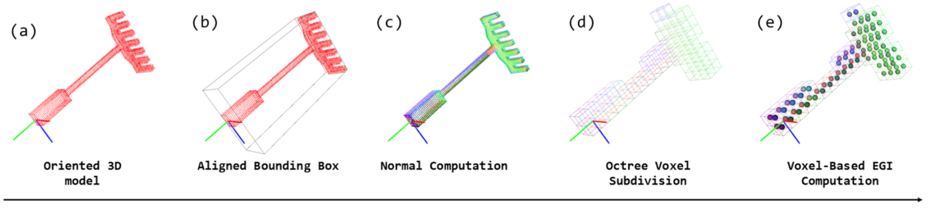


Figure 4.7: Visual representation of the OMS-EGI feature extraction process. Normal XYZ directions are color-mapped to RGB for visualization.

This set of features was obtained for 44 virtual tool 3D models, again in three different orientations each. The resulting dataset was clustered using a Self-Organizing Map K-Means, in order to discover the available tool-pose categories, based on their 3D geometry similarity as measured by the OMS-EGI. For each discovered category, a regressor was trained, linking the actions and effects corresponding to all tool-poses belonging to that category. A diagram showing this process can be observed in Figure 4.8. Testing was performed by comparing, for each tool-pose, the predicted effect in each possible direction to the recorded one. The predictive performance obtained was similar to that obtained in the previous experiment.

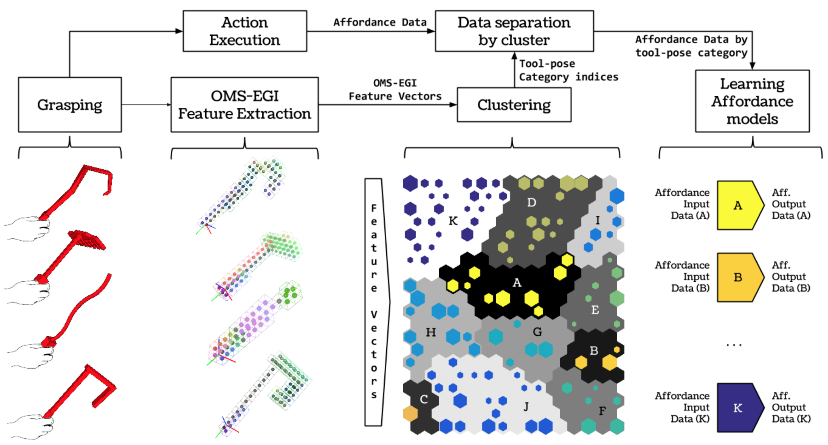

Action Execution
Affordance Data
Data separation by cluster
Affordance Data by tool-pose category
Tool-pose Category indices
Grasping
OMS-EGI Feature Extraction
OMS-EGI Feature Vectors
Clustering
Learning Affordance models
Feature vectors
K
D
I
A
E
H
G
B
C
J
F
Affordance Input Data (A)
A
Aff. Output Data (A)
Affordance Input Data (B)
B
Aff. Output Data (B)
...
Affordance Input Data (K)
K
Aff. Output Data (K)

Figure 4.8: Diagram of experiment published in [4].

This experiment [4] was performed solely in simulation, mainly because of the availability of the full 3D models of the tools for feature extraction, once they were built for the simulator. However, proper evaluation of the method requires testing on the real robot too. Therefore, during the last months we have carried out work (not published yet) in order to adapt the Humanoids 2015 experiment so that it can be carried out on the real robot. This required some modifications in the action execution, so that they are safer for the robot as well as reimplementation of the object localization for reaching and measuring effect, which was not needed in simulation, with a template tracker. Also, the clustering procedure was improved by steering it based on the predictive performance of the models based on each possible cluster scheme, rather than by cluster quality heuristic measures as before. Moreover, action selection based on the predicted effect of each possible action was also implemented, [5].

4.4 Learning peripersonal space representation through artificial skin for avoidance and reaching with whole body surface

This live demonstration (video also available here "**IIT_peripersonal-space.mp4**") will showcase a robot that learns a distributed representation of space around its body by exploiting a whole-body artificial skin and through physical contact with the environment. More specifically, every taxel has a spatial receptive field extending 20 cm along the normal to the skin surface. In this space, 'visual events' triggered by objects coming close to the robot are recorded. If they eventually result in physical contact with the skin, the activated taxels update their representation tracing back the oncoming object and increasing the stored probability that such an event is likely to contact the particular taxel. Other taxels on the body part that were not physically contacted also update their representations with negative examples. The spatial RF around every taxel is mediated by an initial kinematic model of the robot; however, it is adapted from experience, thus automatically compensating for errors in the model as well as incorporating the statistical properties of the oncoming objects. This representation naturally serves the purpose of predicting contacts with the whole body of the robot, which is of clear behavioral relevance. Furthermore, we will demonstrate a simple avoidance controller (Figure 4.10) that is triggered by this representation, thus endowing a robot with a 'margin of safety' around its body. Finally, simply reversing the sign in the controller we used gives rise to simple 'reaching' for objects in the robot's vicinity (Figure 4.9), which automatically proceeds with the most activated (closest) body part. An important asset of the proposed architecture is that learning is fast, proceeds in parallel for the whole body, and is incremental. That is, minutes of experience with objects coming toward a body part give already rise to a reasonable representation in the corresponding taxels that is manifested in the predictive activations prior to contact as well as avoidance behavior. The smoothing approach used (Parzen windows applied to the discrete representation) specifically contributes to this effect in the case of under-sampled spaces. The demonstrators avoidance and reaching are simply exploiting the Cartesian Controller to generate movements of a virtual point that is a result of voting of taxels activated by an object near the robot.

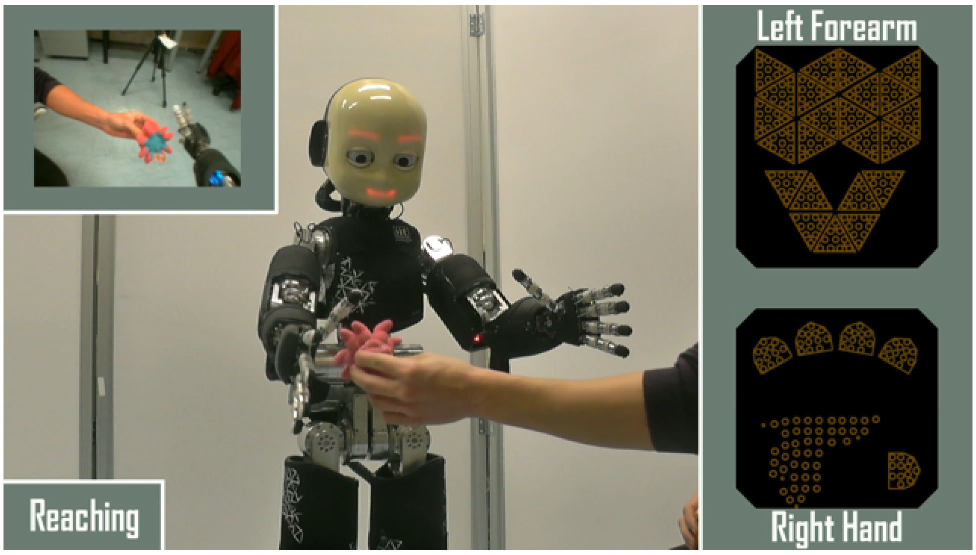


Figure 4.9: Reaching pps behaviour.

The direction of the movement is also a weighted average of the normals of the activated taxels. Avoidance

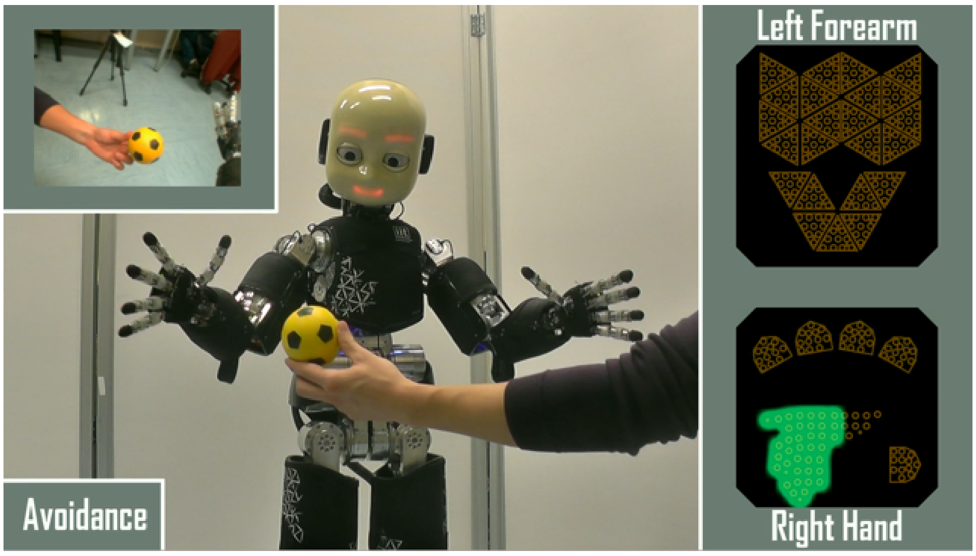


Figure 4.10: Avoidance pps behaviour.

differs from 'reaching' in the direction of this movement vector only. The response is thus local in a sense that there is only one averaged locus for the response. At the same time, the response is executed globally, since the Cartesian controller is employed, recruiting multiple joints in a coordinated fashion. However, this approach will not automatically scale to multiple skin parts activated at the same time (the averaging may produce counterintuitive locus and movement vectors in some configurations) or to the presence of multiple objects near the robot. The 'reaching' behavior is in fact rather a local 'magnet-like' response that will pull the skin parts close to an object toward it. No response will be elicited if the object leaves the 20 cm zone surrounding the body. Therefore, integrating the proposed representation with proper reaching in the robot's workspace in the presence of clutter, while utilizing the safety margin or the 'magnetizing margin' on the way, remains the topic of future work. In addition, the proposed representation could also be expanded by incorporating an additional variable next to the distance, namely the velocity or time to contact of the oncoming objects. Finally, the framework proposed is applicable also to other robots that are equipped with the key sensory modalities: vision (could be easily replaced by other sensors such as Microsoft Kinect or laser range finders), proprioception, and touch.

4.5 Demo on object and part segmentation using LCCP and CPC algorithms

The video "**UGOE_ObjSegmLCCP_PartSegmCPC.mp4**" shows a demo of objects, and parts of objects segmentation methods for the identification of functional parts of objects that are used to guide robotic actions. The demo presents how the system is able to identify handles of objects such as knives, cups, axes, and heat-guns. These handles are afterwards grasped using a KUKA arm platform. The object segmentation is basically used for object recognition while the part segmentation is used to identify the functional parts of the recognized objects. Object and part of object segmentations were implemented using the Local Concave Connected Patches (LCCP) and the Constrained Planar Cuts (CPC) methods, respectively.

4.6 Demo on learning how to grasp unknown objects

The video "**GraspAffordances.mp4**" demonstrates the execution of grasps of unknown objects with the approach described in ([9, 8], see also WP2.1 and WP2.3), where visual features and their combination are learned by a K-means clustering approach. By that discrete sets of prototypes of circular neighborhoods of local surface patches are found first by unsupervised learning techniques and are then combined with grasping actions, evaluated in simulation, to learn grasping affordances in a probabilistic way. By utilising a voting scheme that allows for multiple visual features to vote for a single grasp, the learned affordances

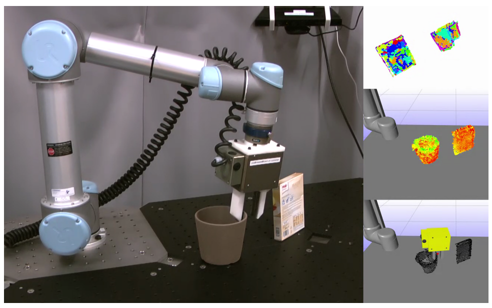


Figure 4.11: Grasping of unknown objects at the SDU platform.

are applied to grasping of novel objects. Grasps and a visual feature representation are learned in a simulated environment using RGB-D sensors for visual feature extraction and object dynamics for grasp simulation and are then applied on the robot set-up at SDU, see Figure 4.11.

References

[1] M. Deniša, A. Gams, A. Ude, and T. Petrič. Learning compliant movement primitives through demonstration and statistical generalization. *IEEE/ASME Transactions on Mechatronics*, 2015. doi: 10.1109/TMECH.2015.2510165.

[2] 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.

[3] A. Gams, T. Petrič, J. Babič, L. Žlajpah, and A. Ude. Constraining movement imitation with reflexive behavior: robot squatting. In *2011 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 294–299, Bled, Slovenia, 2011.

[4] Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Multi-model approach based on 3d functional features for tool affordance learning in robotics. In *Humanoid Robots (Humanoids), 2015 15th IEEE-RAS International Conference on*, 2015.

[5] Tanis Mar, Vadim Tikhanoff, Giorgio Metta, and Lorenzo Natale. Self-supervised learning of grasp dependent tool affordances on the icub humanoid robot. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 3200–3206. IEEE, 2015.

[6] Wail Mustafa, Mirko Waechter, Sandor Szedmak, Alejandro Agostini, Dirk Kraft, Tamim Asfour, Justus Piater, Florentin Worgotter, and Norbert Krüger. Affordance estimation for vision-based object replacement on a humanoid robot. In *ISR 2016 - 47th International Symposium on Robotics 2016.*, submitted.

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

[8] 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.

[9] 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.

[10] 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)*, pages 481–486, 2014.

[11] 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)*, pages 2627–2633, 2015.

Attached Articles

[UP15] E. Ugur and J. Piater. Refining discovered symbols with multi-step interaction experience. In *IEEE International Conference on Humanoid Robotics*, 2015.

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

Refining discovered symbols with multi-step interaction experience

Emre Ugur and Justus Piater

Abstract— **In our previous work, we showed how symbolic planning operators can be formed in the continuous perceptual space of a manipulator robot that explored the world with its single-step actions. In this paper, we extend our previous framework by enabling the robot to progressively update the previously learned concepts and rules in order to better deal with novel situations that appear during multi-step action executions. Our proposed system can infer categories of the novel objects based on previously learned rules, and form new object categories for these novel objects if their interaction characteristics and appearance do not match with the existing categories. Our system further learns probabilistic rules that predict the action effects and the next object states. There rules are automatically encoded in Planning Domain and Definition Language (PDDL), enabling use of powerful symbolic AI planners. Using this framework, our manipulator robot updated its reasoning skills from multi-step stack action executions. After learning, the robot was able to build stable towers in real world, exhibiting some interesting reasoning capabilities such as stacking larger objects before smaller ones, and predicting that cups remain insertable even with other objects inside.**

I. Introduction

Autonomous robots require high-level grounded cognitive capabilities to achieve general-purpose complicated tasks. In our previous research [1], a manipulator robot built symbolic planning concepts and operators from it's own continuous interaction experience with the world. Starting from low-level object percepts, the robot organized its sensorimotor space forming object and effect categories; and learned logical rules that encode the relations between these categories in a form suitable for direct utilization of off-the-shelf AI planners. With this, we argue that we closed the loop by going all the way from continuous sensorimotor experience to symbolic level, finally executing the plans in a real robot.

In the system described above, the symbols and rules were learned from robot's isolated action executions. However, when the robot executes the sequence of actions planned towards a goal, it would encounter with completely novel situations that cannot be perceived or reasoned about with the knowledge that the robot acquired in previous learning. Therefore, the previously learned perceptual and prediction mechanisms should be updated based on robot's further experience obtained from different sequences of actions.

In our case, the robot previously learned operators from isolated stacking actions, where individual objects were stacked on top of other individual objects. After learning,

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.

University of Innsbruck, Institute of Computer Science, IIS Innsbruck, Austria `firstname.lastname@uibk.ac.at`

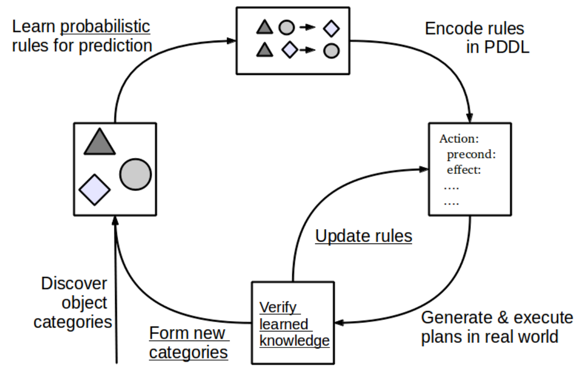


Fig. 1. The learning cycle. Adapted from the conceptual cycle defined in Xperience project: http://www.xperience.org/. The contributions in this paper are marked with underlined text.

we showed that the robot was able plan sequence of stack actions in order to build towers of arbitrary heights based on self-discovered object categories. However, the previously acquired knowledge would fail to capture the characteristics of a tower building task as the system cannot do reasoning over the compound objects formed during this task. Therefore, the robot is required to learn new concepts and rules related to the towers from its observation of sequential stack execution.

Fig. 1 provides an overview of our approach to progressive cognitive skill development. As we described above, our previous work went all the way from object category discovery to the execution of the symbolic plans, however it did not acquire further knowledge from the sequence of interactions. The contributions of this paper, which are marked in the figure, are as follows:

1) The categories of the novel complex objects, which are generated during interactions, are extracted from the learned rules, based on what kind of effects the objects generate in the following interactions. The previously learned rules are updated based on new information.
2) The novel objects are assimilated into the existing categories or new categories are accommodated for the novel objects depending on the visual properties of the novel objects. New object categories are formed if the visual appearance of components of this novel objects are not predicted to belong to the inferred category.
3) Probabilistic rules are learned from real-world interactions of objects, and symbolic planning is achieved using these learned operators.

Very few recent studies have addressed bottom-up construction of symbolic or sub-symbolic structures for planning in robotics. Ugur et al. and Pisoka and Nehmzow clustered the continuous sensory space of the robot and generated multi-step plans in continuous perceptual space with learned effect predictions [2], [3]. Mugan and Kuipers proposed a system that learns qualitative representations of states and predictive models in a bottom-up manner by discretizing the continuous variables of the environment [4]. Konidaris et al. studied construction of symbols that are directly used as preconditions and effects of actions for generation of deterministic [5] and probabilistic [6] plans in simulated environments. Pasula et al. [7] and Lang et al. [8] studied learning of symbolic operators using pre-defined predicates in simulated blocks world domains. Our framework, on the other hand, learns non-linear relations between the **discovered** discrete symbols and the continuous percept of a **real** manipulation robot.

II. Methods

In this section, we first give the manually designed perceptual and motor capabilities of the robot. In Section II-B, we summarize what kind of structures and rules were autonomously learned in and transferred from our previous study [1]. Finally, in Section II-C, we provide the main contribution of this paper, where the robot detects categories of novel objects, forms new categories, develops new prediction capabilities and probabilistic rules from observations of object interactions generated by sequence of actions.

A. Built-in knowledge

The robot is equipped with a number of manually-coded actions (a_j) that enable single- and multi-object manipulation. It can push a single object from different sides, pick-up, and release it; and also stack one object onto another. The single-object actions were used to find action-grounded object categories in previous stages, and not further explored in this paper.

The robot has the built-in functionality of detecting objects and extracting a number of visual features from them. The list of these features represent the objects in continuous sensory space and is denoted as $\boldsymbol{f}_o$ throughout the text. The continuous effect created in object features during action executions are also observed by the robot to find the discrete effect categories, as detailed in the next section.

B. Capabilities transferred from previous learning stages

In previous learning stages [1], the robot executed actions that involve single objects and pairs of objects, and progressively learned the following information. Effect space was discretized. For this, effect categories (ε_o^a) were formed by applying unsupervised clustering methods to the set of observed continuous effects for each action. The formed effect categories for different actions were as follows:

$$\begin{aligned}
\varepsilon^{\text{pick-up}} &\in \{\textsc{grasped}\} \\
\varepsilon^{\text{release}} &\in \{\textsc{stable, tumbled}\} \\
\varepsilon^{\text{front-poke}} &\in \{\textsc{rolled, pushed}\} \\
\varepsilon^{\text{side-poke}} &\in \{\textsc{rolled, pushed}\} \\
\varepsilon^{\text{top-poke}} &\in \{\textsc{inserted, obstructed}\} \\
\varepsilon^{\text{stack}} &\in \{\textsc{stacked, inserted, tumbled}\}
\end{aligned} \tag{1}$$

Action-grounded object categories ($\{c_o\}$) were formed. For this, objects that were affected similarly from the robot actions were grouped together. Therefore, object categories were encoded as the collection of effect categories generated by the five single-object actions ($c_o = (\varepsilon^{a_1}, \varepsilon^{a_2}, ..\varepsilon^{a_5})_o$). The formed object categories were as follows:

$$c_o \in \{\textsc{hollow, solid, rollable, unstable}\} \tag{2}$$

where

$$\begin{aligned}
\textsc{hollow} &= (\textsc{grasped, stable, pushed, pushed, inserted}) \\
\textsc{solid} &= (\textsc{grasped, stable, pushed, pushed, obstructed}) \\
\textsc{rollable} &= (\textsc{grasped, stable, rolled, rolled, obstructed}) \\
\textsc{unstable} &= (\text{-}, \textsc{tumbled}, \text{-}, \text{-}, \text{-})
\end{aligned}$$

Prediction of object categories from their visual appearance was acquired. For this, the mapping from object features to the corresponding object categories was learned by training non-linear classifiers (SVMs):

$$c_o = \mathcal{C}(\boldsymbol{f}_o) \tag{3}$$

Finally, predicting more complex action effects, i.e. effects of stack action, was learned. For this purpose, decision tree learners were trained to find logical rules that return the stack effect category given categories of the involved objects and their relations :

$$\varepsilon^{\text{stack}} = \mathcal{R}(c_b, c_r, rel(o_b, o_r)) \tag{4}$$

Table I gives the results of decision tree learning along with the corresponding rules. Given categories of objects and discrete relations between them, the effect category is predicted according to this table [1].

TABLE I

The deterministic rules learned from the robot simulator.

	Below = HOLLOW		Below = SOLID
	.. Rel-Width = below-smaller	10	.. Above = HOLLOW: STACKED
01	.. .. Above = HOLLOW: STACKED	11	.. Above = SOLID: STACKED
02	.. .. Above = SOLID: STACKED	12	.. Above = ROLLABLE: STACKED
03	.. .. Above = ROLLABLE: INSERTED	13	.. Above = UNSTABLE: TUMBLED
04	.. .. Above = UNSTABLE: INSERTED	14	Below = ROLLABLE: TUMBLED
	.. Rel-Width = same-width		
05	.. .. Above = HOLLOW: STACKED		
06	.. .. Above = SOLID: STACKED		
07	.. .. Above = ROLLABLE: INSERTED		
08	.. .. Above = UNSTABLE: INSERTED		
09	.. Rel-Width = below-bigger: INSERTED		

C. Learning from sequence of actions

After learning logical rules on how to stack pairs of objects, the robot could build towers of arbitrary size using the available objects in the environment by planning and executing sequence of stacking actions. However, as the rules were learned from isolated single stacking interactions, the

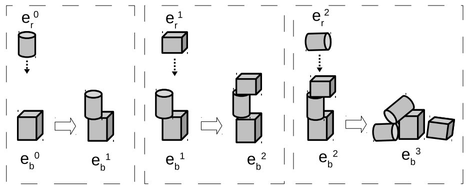


Fig. 2. One hypothetical episode used in learning from action execution sequence. This episode is composed of three interactions.

sequential effects of the successive action executions are not represented in these rules. In this section, we detail the methods that aim to enhance the learned rules by directly learning from the experience of sequence of stacking interactions.

The learning is achieved in episodes that are composed of successive stacking interactions. Each episode starts with stacking two objects on top of each other, continues with putting new objects on top of the stack, and finishes when the object tower collapses. Fig. 2 shows one hypothetical episode, which lasts three interactions, where e_r and e_b correspond to the *released* and *base* entities, respectively. Note that the perception system cannot distinguish between touching objects, therefore the concept 'entity' will be used to refer to both individual and composite objects. If the entity e corresponds to an individual object o, than $e = o$, otherwise the entity is encoded as the list of the objects included: $e = \{o_0, o_1, ..\}$.

The robot observes, computes, and stores the following information in each interaction i:

- Visual features of the detected entities before and after the interaction. The point clouds in beginning and at the end of the interaction are processed to compute visual features of the entities $(\boldsymbol{f}_{e_r^i}, \boldsymbol{f}_{e_b^i})$.
- Object category of the released entity, i.e. $c_{e_r^i}$. The classifier introduced in (3) is used to find the object category based on visual features of the object.
- Object category of the base entity, i.e. $c_{e_b^i}$. If this is the first interaction of the episode, $c_{e_b^i} = c_{o_b^i} = \mathcal{C}(\boldsymbol{f}_{o_b^i})$ (recall the classifier in (3)). If this is not the first interaction, i.e. the base entity is a composite object, the category of this entity is assigned to be unknown.One major aim of this paper is to reveal the categories of these composite objects by observing how they affect the subsequent stack interactions.
- The list of the objects that compose the base entity:
$$e_b^i = \{o_r^{i-1}, \dots o_r^1, o_r^0, o_b^0\} \tag{5}$$
- The result of the stack interaction (ε^i). It is assigned to one of the three effect categories ({STACKED, INSERTED, or TUMBLED}), which were provided in (1).

From above, one can notice that the interaction instances lack the information regarding the category of the entity on the ground (e_b), if this entity is a composite object. As shown in Fig. 2, entities on the ground are composed of several objects with different categories, and which category this composite object belongs to is ambiguous. As the objects might be inserted in others or remain on the top of the stack, the resulting category might be one of the categories that constitute the composite object. If the orientation of the released object change or the top surface is combination of several objects, the resulting category of the base object might be something completely different.

Algorithm 1 Inferring category of complex base entities

```
1: for all episodes do
2:     for all interactions $i$ do
3:         for all objects $o' \in e_b^i$ do
4:             if $\varepsilon^i = \mathcal{R}(c_{o'}, c_{e_r^i}, rel(o', e_r^i))$ then
5:                 $e_b^i \leftarrow o'$
6:                 assimilate $\boldsymbol{f}_{e_b^i}$ into $c_r^i$
7:                 go to 2
8:         for all possible categories $c_j$ do
9:             if $\varepsilon^i = \mathcal{R}(c_j, c_{e_r^i}, rel(*, e_r^i))$ then
10:                $c_{e_b^i} \leftarrow c_j$
11:                if $\mathcal{C}(\boldsymbol{f}_{e_b^i}) = c_j$ then
12:                    assimilate $\boldsymbol{f}_{e_b^i}$ into $c_j$
13:                else
14:                    accommodate $\boldsymbol{f}_{e_b^i}$ into $c_j^{new}$
15:                go to 2
```

1) Inferring categories of complex entities: Categories of base entities are inferred using Algorithm 1. This algorithm finds the category of the base entity in each interaction (i), given the category of the released object (c_r^i), the effect observed in that interaction (ε^i), and the set of previously learned rules that predict the effect category given the categories of the objects and their relations (See (4) and Table I). The algorithm first checks if any possible interaction with one of the objects that is included in the base entity is represented in the set of rules (lines 3-7). These objects are checked starting from the last one as the latter added objects have higher probability to influence the category of the composite base entity. If no object satisfies the conditions expressed in rules, then the system checks all possible object categories, even if they were not used previously in the current episode (lines 8-15). If the conditions of any rule is satisfied with this possible category, i.e. if the observed effect can be generated based on the rules with this category, then the base category is assumed to be this one. Please note that there is an important distinction between assigning a previous object to the base entity (line 5), and assigning a possible category (line 10). In the first case, the other features of the entity (such as width and height) are also known and stored, whereas in the latter one, the features of the base entity are left unknown.

2) Assimilation/accommodation mechanism: In Algorithm 1, we also described our assimilation and accommodation mechanism. The novel complex entity is assimilated into an existing category either if an object of the same category is a part of the complex entity (line 6) or the objects in the inferred category are visually similar to the complex entity (line 12). If these conditions are not satisfied, a new category is forked from the inferred category, and novel similar objects that are assigned to this new category.

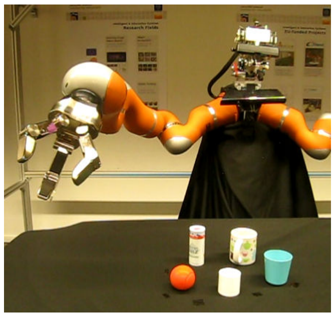


Fig. 3. The robot arm and gripper are used for manipulation, and Kinect is used to compute object features.

3) Rule learning: After categories of base entities are inferred, the system learns to predict the effect and the category of the formed entity given the categories of the released and base entities, and their relations:

$$(c_{e_r}, c_{e_b}, rel(e_r, e_b)) \rightarrow (\varepsilon, c_{e'_b}) \tag{6}$$

4) Symbolic planning: The robot builds symbolic domain and problem descriptions based on the object categories and the learned rules. The predicates correspond to the automatically discovered object categories and relations. Actions correspond to the learned rules with following fields:

- Preconditions: The list of the predicates that should be valid in order to apply the action. This corresponds to the object categories and their discrete relations (left part of (6)) for each learned rule.
- Effects: The list of the predicates that change if the preconditions are satisfied, and the action is executed. The predicted effect categories (right part of (6)) are provided in this field.

The initial state of the world, i.e. object categories and discrete relations between the objects, and the goal is defined in the problem description, in STRIPS notation as well. Given domain and problem descriptions, off-the-shelf symbolic planners are used to acquire the desired tasks.

The learned rules are represented probabilistically, however planning directly in probabilistic domain is inherently more complex compared to deterministic domain because of the high computation complexity [9]. In this paper, we take a path in between, compute deterministic plans using rules with highest probabilities, and re-plan with other rules if no plan is formed. Depending on the joint probability of the actions, the plan might or might not be executed - however we did not further analyze this verification step.

III. EXPERIMENTS

A. Robot Platform and Interactions

Our experimental setup is composed of a KUKA Light Weight Robot arm and a Schunk gripper for manipulation, a Kinect sensor for environment perception, and a number of objects that are placed on the table for exploration (Fig. 3). The workspace consists of several objects and a table. First, the point cloud is segmented in order to detect the existing objects. Next, continuous object state is found by computing the following features:

$$\boldsymbol{f}_o = (vis_o, pos_o, shape_o, dim_o, dist_o)$$

where vis feature encodes the knowledge regarding the existence of the object, $shape$ encodes the distribution of local surface normal vectors from the object surface. pos and dim correspond to the center and size of the object, respectively. Finally, $dist$ features encode the distribution of the local distance of all pixels to the neighboring pixels [1].

The robot is equipped with a manually-coded stack action, where the vertically-aligned gripper grasps an object using built-in spherical grip first, carries it on top of the another object, and releases it.

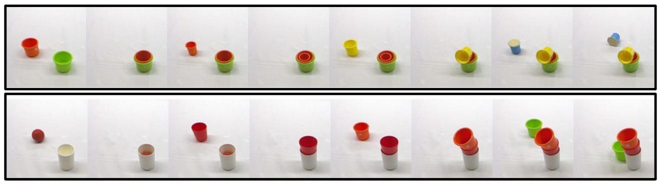
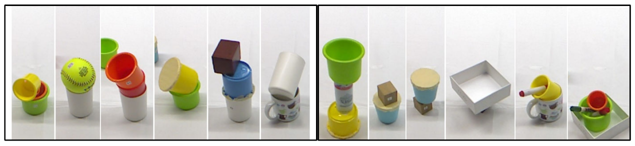


Fig. 4. Sample snapshots from stack interactions. The objects on the left are always stacked on top of the objects on the right.



(a) Assimilate (b) Accommodate

Fig. 5. (a) Assimilate the entities into ROLLABLE category. (b) Accommodate a new category ROLLABLE-NEW for the entities.

B. Interactions

In order to collect the interaction dataset, a human imitated stack action of the robot. The initial and final point clouds of each interaction are stored for later processing. The dataset contains 26 episodes and 66 interactions in total. A number of sample interactions are provided in Fig. 4[1]

C. Learned categories and category predictions

In this section, we analyze the learned rules that predict the next category of the base entity given the current categories of the base and released entities, and their relations.

First of all, the system formed a new object category, which has similar dynamics with ROLLABLE objects under stacking interactions, but different visual appearance. As shown in Fig. 5b, the entities represented by this new category generally correspond to towers of objects which

[1]In order to imitate the noise in perception and the crude stacking skill of the robot, we introduced small offsets while dropping objects; and instead of carefully and gently placing the objects, we dropped the objects from the air similar to the release behavior of the robot. While ideally learning should be achieved through robot's own exploration, we discuss that it might be acceptable to make such simplifications in exploring object-object interactions as long as the learning results are verified with the real robot. In future, we will use robot's own interaction experience in order to better capture uncertainty in robot actions.

TABLE II

THE PROBABILITIES OF GENERATING THE NEXT BASE ENTITY CATEGORY GIVEN INTERACTING ENTITIES.

#	Base	Released	Rel-Width	HOLLOW	SOLID	ROLLABLE	UNSTABLE	ROLL-NEW	UNDEFINED
01	HOLLOW	*	Base-small	0.3	0.0	0.3	0.00	0.0	0.3
02	HOLLOW	HOLLOW	Base-big	0.7	0.0	0.2	0.0	0.2	0.0
03	HOLLOW	SOLID	Base-big	0.0	1.0	0.0	0.0	0.0	0.0
04	HOLLOW	ROLLABLE	Base-big	0.6	0.0	0.4	0.0	0.0	0.0
05	HOLLOW	UNSTABLE	Base-big	0.5	0.0	0.2	0.0	0.2	0.0
06	SOLID	HOLLOW	Base-small	0.0	0.0	0.0	0.0	0.7	0.3
07	SOLID	HOLLOW	Base-big	0.7	0.0	0.3	0.0	0.0	0.0
08	SOLID	SOLID	Base-small	0.0	0.2	0.0	0.0	0.2	0.6
09	SOLID	SOLID	Base-big	0.0	0.7	0.1	0.0	0.1	0.0
10	SOLID	ROLLABLE	*	0.0	0.0	0.0	0.0	0.0	1.0
11	SOLID	UNSTABLE	*	0.0	0.0	0.0	0.0	0.0	1.0
12	ROLLABLE	*	*	0.0	0.0	0.0	0.0	0.0	1.0
13	ROLL-NEW	*	*	0.0	0.0	0.0	0.0	0.0	1.0

are not stable or which do not allow further stacking. Not allowing further stacking is a similar characteristic with ROLLABLE objects, but as shown, the visual appearances of these entities are very different from ROLLABLE objects.

Table II provides the rules that are obtained from decision tree learning. The system revealed several different important underlying characteristic of the sequential stacking operations. In a number of cases, the next category of base entity is UNDEFINED, which corresponds to TUMBLED effect. In some situations, the category of the released entity is transferred to the formed base entity (rules 03, 07, 09). For example, if a HOLLOW entity is stacked on top of a bigger SOLID entity (rule 08), the new formed base entity becomes HOLLOW with higher probability. In some other situations, the category of the base entity is preserved. For example, when a small ROLLABLE object or an UNSTABLE object is stacked on top of a HOLLOW entity (rules 04 and 05), then base entity remains HOLLOW. Finally, a new category is formed from different categories. For example, when a HOLLOW object is stacked on top of a smaller SOLID object (rule 06), the category of the generated base entity becomes NEW-ROLLABLE.

We argue that these rules reveal some interesting underlying characteristics of sequential stacking actions such as HOLLOW objects filled with small objects remain HOLLOW. However, these rules are neither inclusive nor all correct. For example, we can argue that rule 06 is not really intuitive: a HOLLOW object that is stacked on top of smaller SOLID object should generate a HOLLOW or UNDEFINED entity, but it generates ROLLABLE-NEW or UNDEFINED instead. When investigated in detail, this rule was extracted from instances, where these two objects are stacked on other HOLLOW objects, whose walls prevent the objects from falling, but also creating a structure which is not stackable anymore. We still believe that even the learned information is not perfect, it will enable construction of more safe and conservative plans. Rule 07, for example, will try to avoid stacking of large HOLLOW object on smaller SOLID objects, which would create unstable towers.

TABLE III

THE PROBABILITIES OF GENERATING AN EFFECT GIVEN OBJECT CATEGORIES AND THEIR RELATIONS.

#	Base	Released	Rel-Width	STACKED	INSERTED	TUMBLED
01	HOLLOW	*	Base-small	0.22	0.44	0.33
02	HOLLOW	*	Below-big	0.00	1.00	0.00
03	SOLID	HOLLOW	*	0.83	0.00	0.16
04	SOLID	SOLID	Base-small	0.40	0.00	0.60
05	SOLID	SOLID	Below-big	1.00	0.00	0.00
06	SOLID	ROLLABLE	*	0.00	0.00	1.00
07	SOLID	UNSTABLE	*	0.00	0.00	1.00
08	ROLLABLE	*	*	0.00	0.00	1.00
09	ROLL-NEW	*	*	0.00	0.00	1.00

D. Learned effect prediction

We used C4.5 decision tree learning algorithm to find the set of rules with the corresponding probabilities from real interactions. Table III provides the results of decision learning. The results can be interpreted as follows:

- (01-02): If the base entity is HOLLOW and has a larger width, then always an insertion occurs. However, if the base entity is smaller, then STACKED and TUMBLED also become possible. TUMBLED was not learned in the original rules that had been trained in the simulator.
- (04-05): If both entities are SOLID, then they would be STACKED unless the released object is bigger. If the released object is bigger, the objects might be STACKED or TUMBLED with similar probabilities.
- (03): HOLLOW objects released over base objects would be probably STACKED, but there is small probably of observing TUMBLED effect.
- (06-07): The ROLLABLE or UNSTABLE entities that are released on SOLID objects generate TUMBLED effect. In the rules learned from the robot simulator, the ROLLABLE objects was creating STACKED effect when released on the SOLID objects, however in reality we observed that they roll off the objects after released.
- (08-09): If the entity on the base belongs to ROLLABLE or ROLLABLE-NEW category, then the effect is always TUMBLED.

The set of rules above (Table III) make more realistic predictions in the real world compared to the set of rules learned from the robot simulator (Table I).

E. Generated STRIPS rules

We used C4.5 decision tree learners to predict both the effect of the action and the category of the formed base object. The set of rules (in Table IV) are then automatically transferred to PDDL. An explanatory sample action that corresponds to rule 04 is provided in Fig. 6. Lines 2-3 state preconditions and 4-6 state effects of the action. Predicate 'U', which is 'U0' for all objects in the beginning, encodes the order of the object in stacking. 'N' and 'H' predicates correspond to the number of objects in the stack and height of the stack, and they change based on the predicted effect.

F. Real World Experiments

Given objects, the robot first finds the object categories from visual features of objects using the trained non-linear

TABLE IV
THE PREDICTED (EFFECT, ENTITY) GIVEN STACKED ENTITIES.

	Below = HOLLOW
01	— Rel-Width = below-smaller: **TUMBLED (0.3), UNDEFINED (0.3)**
	— Rel-Width = below-bigger
02	— — Above = HOLLOW: **INSERTED (1.0), HOLLOW (0.6)**
03	— — Above = SOLID: **INSERTED (1.0), SOLID (1.0)**
04	— — Above = ROLLABLE: **INSERTED (1.0), HOLLOW (0.6)**
05	— — Above = UNSTABLE: **INSERTED (1.0), HOLLOW (0.5)**
	Below = SOLID
	— Above = HOLLOW:
06	— — Rel-Width = below-smaller: **STACKED (0.6), ROLLABLE-NEW (0.6)**
07	— — Rel-Width = below-bigger: **STACKED (1.0), HOLLOW (0.6)**
	— Above = SOLID:
08	— — Rel-Width = below-smaller: **TUMBLED (0.6), UNDEFINED (0.6)**
09	— — Rel-Width = below-bigger: **STACKED (1.0), SOLID (0.7)**
10	— Above = ROLLABLE: **TUMBLED (1.0), UNDEFINED (1.0)**
11	— Above = UNSTABLE: **TUMBLED (1.0), UNDEFINED (1.0)**
12	Below = ROLLABLE : **TUMBLED (1.0) , UNDEFINED (1.0)**
13	Below = ROLLABLE -new: **TUMBLED (1.0) , UNDEFINED (1.0)**

```
(:action stack :parameters (?Below ?Above) ;;; rule no: 04
:precondition (and (Hollow ?Below) (Rollable ?Above)
(Below-bigger ?Below ?Above) (U0 ?Above)(U2 ?Below) (N2) (H0)
:effect (not (U0 ?Above)) (U3 ?Above) (U2 ?Below)
(N3) (not (N2)) (not (Hollow ?Below)) (not (Rollable ?Above))
(Hollow ?Above))
```

Fig. 6. Automatically generated sample action in PDDL that corresponds to rule 04 in Table IV. ?Above variable takes the role of formed base entity in the effect field.

classifiers, and generates the domain description based on object categories. Next, it runs the Blackbox off-the-shelf planning software [10], setting the 'S' predicate to the number of objects, and the 'H' predicate to the minimum number initially. If no plan is generated, the goal 'H' predicate is increased by one in a loop. For plan generation, we use the Fig. 7 provides snapshots from a representative plan generated and (blindly) executed for the goal of building tower that include all the objects independent of any constraint on compactness or height. As seen, given various objects with different affordances, the robot planned the sequence of actions that generate a stable tower. Due to the the learned probabilistic rules that favor stacking small objects on larger objects (e.g. rules 04 and 05 in Table III), the robot implicitly learned building stable towers from objects of different sizes. Note that this knowledge was not encoded in the rules that were transferred from previous stages (Table I), which shows the advantages of further learning from real interactions. One can also observe that the robot correctly reasoned about the properties of the containers by planning to insert several board-markers (detected as UNSTABLE objects) into the cup.

During the experiments, we observed that the robot sometimes suffered from incorrect categorization of objects based on visual features obtained from Kinect. The second reason for the failures was the inaccuracies in perception of the release location and execution of the stack action. Finally, there were some new categories that do not conform to any extracted rules, therefore not learned by the system.

We argue that the problems addressed above can be avoided by better perception, more training, and more advanced action representations. However, there are other more major limitations which are not straightforward to address with our current approach. For example, while our system

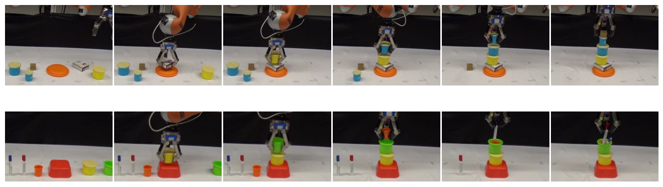


Fig. 7. Stack of objects with mixed categories. Snapshots show that the robot builds the stacks starting from larger objects, and can reason about inserting several small objects into the containers. The robot execution videos are available at http://emreugur.net/humanoids2015/.

provides the capability to detect whether the objects remain HOLLOW or not from their visual perception during action executions; it does not have the capability to reason about how many objects can fit into a HOLLOW objects. This kind of reasoning probably requires other concepts (such as volume of the hole instead of width and height of the object) and learning of more complex capabilities such as arithmetic processing over learned concepts.

IV. CONCLUSION

In this paper, we showed how the robot can further develop previously learned concepts and reasoning skills in order to better represent multi-step interaction characteristics and to make better plans. As we partially discussed at the end of the previous section, our system is currently limited to predicting the next state based on small number of object categories of only the current state with simple object relation information. In future, we plan to investigate how the robot can discover other task dependent and potentially hidden concepts and variables such as stability and height of the towers, and how it can learn more challenging rules that can make reasoning with higher-level knowledge.

REFERENCES

[1] E. Ugur and J. Piater, "Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning," in *ICRA*, 2015, pp. 2627–2633.

[2] 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.

[3] J. Pisokas and U. Nehmzow, "Experiments in subsymbolic action planning with mobile robots," in *Adaptive Agents and Multi-Agent Systems II, Lecture Notes in AI*. Springer, 2005, pp. 80–87.

[4] 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.

[5] G. Konidaris, L. P. Kaelbling, and T. Lozano-Perez, "Constructing symbolic representations for high-level planning," in *28th AAAI Conf. on AI*, 2014.

[6] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Symbol acquisition for probabilistic high-level planning," in *International Joint Conference on Artificial Intelligence*, 2015.

[7] 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.

[8] T. Lang and M. Toussaint, "Planning with noisy probabilistic relational rules," *Journal of Artificial Intelligence Research*, vol. 39, no. 1, pp. 1–49, 2010.

[9] A. L. Blum and J. C. Langford, "Probabilistic planning in the graphplan framework," in *Recent Advances in AI Planning*. Springer, 2000, pp. 319–332.

[10] H. Kautz and B. Selman, "Unifying sat-based and graph-based planning," in *IJCAI*, vol. 99, 1999, pp. 318–325.