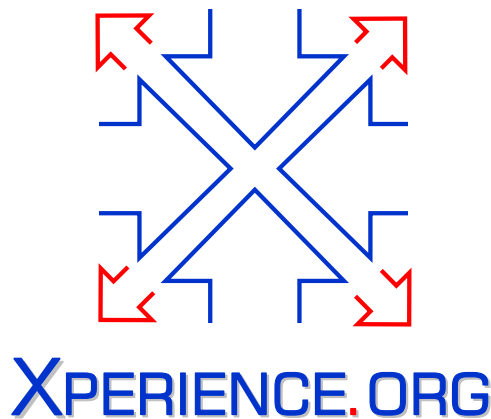




Project Acronym:	Xperience
Project Type:	IP
Project Title:	Robots Bootstrapped through Learning from Experience
Contract Number:	270273
Starting Date:	01-01-2011
Ending Date:	31-12-2015



Deliverable Number:	D3.1.2
Deliverable Title:	Structural bootstrapping on sensorimotor experience (II): Report or scientific publication on introspection and predication, based on internal simulation and accumulated experience, can be applied for purposes of learning and planning
Type (Internal, Restricted, Public):	PU
Authors:	Justus Piater, Tamim Asfour, Christopher Geib, Aleš Ude, Mirko Wächter, Martin Do, Eren Aksoy, Minija Tamosiunaite, Rüdiger Dillmann and Florentin Wörgötter
Contributing Partners:	ALL

Contractual Date of Delivery to the EC: 31-01-2014  
Actual Date of Delivery to the EC: 08-02-2014

# Contents

<b>1</b>	<b>Executive Summary</b>	<b>3</b>
<b>2</b>	<b>Structural Bootstrapping over Different Levels</b>	<b>5</b>
2.1	Introduction . . . . .	5
2.2	Setup - Overview . . . . .	6
2.3	Methods - Short Summary . . . . .	8
2.4	Implementing Structural Bootstrapping at different levels . . . . .	11
2.5	Robotic Implementation . . . . .	18
2.6	Benchmark Experiments . . . . .	18
2.7	Discussion . . . . .	21
2.8	Conclusions . . . . .	23
<b>3</b>	<b>Structural Bootstrapping Examples at Different Levels</b>	<b>25</b>
3.1	Semantic Event Chain-based Assimilation and Accommodation of actions . . . . .	25
3.2	Learning Object-Action Relations . . . . .	25
3.2.1	Active Learning with Knowledge Propagation . . . . .	25
3.2.2	Bootstrapped Affordance Learning . . . . .	26
3.2.3	Learning Object-Action Relations with Homogeneity Analysis . . . . .	26
3.2.4	Generative learning of correlations between object properties and action parameters	26
3.2.5	Object Categorization with Knowledge Propagation . . . . .	27
<b>4</b>	<b>Conclusion</b>	<b>28</b>

# Chapter 1

## Executive Summary

This deliverable shows solutions of the consortium for structural bootstrapping of robotic actions. The contributions reported here focus on two of the five *major scientific questions* defined by the Xperience project:

- We address the second major scientific question by implementing an algorithm that allows the agent to extend its knowledge in a systematic way, and to accelerate learning significantly beyond what an exploration-based approach can achieve.
- We address the third major scientific question by realizing an active learning procedure on the top of the data acquisition loop.

The main deliverable material is presented in two chapters. In Chapter 2 we present an aggregated structural bootstrapping solution extending over all levels of the Xperience system: planning level, mid-level and sensorimotor level. This is the summary and the extension of the work that had been presented at the second review meeting's introductory talk and is now being submitted to IEEE TAMM as shared work of the consortium [WGT<sup>+</sup>14]. The bootstrapping approach in the planning and mid-level is similar, as we rely on sequence-wise similarity of actions at those two levels. When two (planning or mid-level) sequences are similar enough we say we can bootstrap. At the sensorimotor level we rely on trajectory similarities as well as object clustering in the object-action parameter space. As the above summarized contribution extends over the whole Xperience system, the approach is presented in detail in the deliverable.

In Chapter 3 we present a summary of complementary approaches to structural bootstrapping at different levels of the Xperience system, where progress in individual components has exceeded the state of the art. Full-text of the summarized contributions can be found in the attached papers. Here we introduce a Piaget-like action assimilation and accommodation framework based on Semantic Event Chains [ATV<sup>+</sup>13]. A method for knowledge propagation is introduced for the learning object-action relations [SP14] where from an initial set tested in a real robot environment consequent cases for testing can be deduced in an active learning procedure and used for bootstrapped affordance learning [Ugu14]. Homogeneity analysis is introduced as yet another method to learn Object-Action relations [XSP13]. In [DSEA14] an experiential cycle is presented for learning the association between object properties (softness and height) and action parameters for a wiping task and for building generative models from sensorimotor experiences.

In Chapter 4 conclusions and future work are described.



Figure 1.1: Example of structural bootstrapping performed by a human.

Before we start the discussion we would like to show one quite amazing and somewhat entertaining example of human structural bootstrapping (Fig. 1.1), which came from a person unrelated to the Xperience project (a friend of a student). Apparently this person did indeed use this contraption to prepare dough!



## Chapter 2

# Structural Bootstrapping over Different Levels

### 2.1 Introduction

In this chapter we present structural bootstrapping solution extending over different levels of the Xperience system. This had been a core contribution of the Xperience group in the last year and its first germs had been presented at the second year review. A paper based on the ideas presented here will be submitted to the IEEE Transactions of Autonomous Mental Development (TAMD). We give the title page of the paper as an attachment to this deliverable as the complete text is presented next anyhow [WGT<sup>+</sup>14].

It has been a puzzling question how small children at the age of three to four are suddenly able to very quickly acquire the meaning of more and more words in their native language, while at a younger age language acquisition is much slower. Two interrelated processes are being held responsible for this speeding-up. The primary process is semantic bootstrapping where the child associates meaning from observing their world with co-occurring components of sentences. For example, if the word "fill" is consistently uttered in situations where "filling" occurs, then the meaning of the word can be probabilistically guessed from having observed the corresponding action again and again [50, 56]. Once a certain amount of language has been acquired, a second process – named syntactic bootstrapping – can speed this up even more and this is achieved by exploiting structural similarity between linguistic elements. This process can take place entirely within language and happens in a purely symbolic way without influence from the world. For example, if a child knows the meaning of fill the cup and then hears the sentence fill the bowl, it can infer that a bowl denotes a thing that can be filled (rather than a word meaning the same thing as fill) without ever having seen one ([50, 29, 23, 44, 22, 30, 66, 24] see [14] for a comparison between semantic and syntactic bootstrapping). Thus, the most probable meaning of a new word is being estimated on the basis of the prior probability established by previously encountered words of the same semantic and syntactic type in similar syntactic and semantic contexts.

These two generalization mechanisms – semantic and syntactic bootstrapping – are very powerful and allow young humans to acquire language without explicit instruction. It is arguable that bootstrapping is what fuels the explosion in language and conceptual development that occurs around the third year of child development ([66, 65]).

In general "the trick" seems to be that the child possesses at this age already enough well-ordered knowledge (grammar, word & world knowledge) which allows him/her to perform guided inference without too many unknowns. Grammar and word-knowledge are highly structured symbolic representations and can, thus, provide, a solid scaffold for the bootstrapping of language. Symbolic representations, however, do not stop short at human language. For robots, planning, planning operators, and planning languages constitute another (non-human) symbolic domain with which they need to operate. Thus, it seems relatively straightforward to transfer the idea of semantic and syntactic bootstrapping to the planning domain for robot actions. The current paper will first address this problem.

The question, however, arises whether related mechanisms might also play a role for the acquisition of

other, non-linguistic cognitive concepts, for example the properties of objects and tools. Briefly, if you know how to peel a potato with a knife, would there be a way to infer that a potato peeler can be used for the same purpose? This example belongs to the second set of problems addressed in this study: How can a cognitive agent infer role and use of different objects employing the knowledge of previously seen (and used) objects, how can it infer use patterns (for example movement and force patterns, etc.)?

The goal of the current study is to address one complex scenario all the way from the planning- down to sub-symbolic sensorimotor levels and implement (different) bootstrapping processes for the fast acquisition of action knowledge. The only requirement for all these different bootstrapping mechanisms is that there exists a well-structured scaffold as a basis from where on different inference processes can take place. The different scaffolds, thus, form the structures upon which bootstrapping can be built. Hence, we call these processes "structural bootstrapping".

One can consider structural bootstrapping as a type of semi-supervised probabilistic learning, where an agent uses an internal model (scaffold) to quickly slot novel information (obtained for example by observing a human) into appropriate model categories. The advantage of such a bootstrapping process is that the agent will be able to very quickly perform these associations and grounding needs only to take place afterwards by experimenting in a guided way with the new piece of knowledge. Evidently, as this is based on probabilistic guesswork, bootstrapping can also lead to wrong results. Still, if the scaffold is solid enough all this can be expected to be much faster than the much more unstructured and slow process of bottom-up exploration learning or than full-fledged learning from demonstration. (For a comparison to the state of the art see Discussion).

Here we will show that one can implement structural bootstrapping across different levels of our robotics architecture in the humanoid robot ARMAR-III ([6, 7]) trying to demonstrate that bootstrapping appears in different guises and will, thus, possibly not be limited to the case studies presented in this paper. As a major aspect this work is meant to advocate structural bootstrapping as a way forward to a more efficient extension of robot-knowledge in the future. Early on we emphasize that the complexity of the here-shown aspects prevents exhaustive (e.g. statistical) analyses. After all we are dealing with very complicated and possibly human-like cognitive generative (inference) processes for which children and adults need years of experience to reach their final efficiency.

The paper is structured in the following way. First we provide an overview of system, processes, and methods; next we show six different types of structural bootstrapping at different levels. This will be followed by some benchmarks and a discussion section which also includes the state of the art in robot knowledge acquisition.

## 2.2 Setup - Overview

In the following we will give a course overview over the structures and processes used for this study. Some entities will shine up without much explanation but details will follow.

### Scenario (task)

ARMAR operates in a kitchen scenario. The task for the robot is to pour two ingredients (e.g. flour and water) and mix them together to obtain batter. For this the robot has the required knowledge to do it in one specific way (by using an electric mixer), but will fail whenever it should react flexibly to a changed situation (e.g. lack of the mixer). The goal of this work is to show that bootstrapping will quickly provide the required knowledge to successfully react to such a change. This process is based on observing a human providing an alternative solution (stirring with a spoon) where bootstrapping lead to the "understanding" of the meaning of objects and actions involved.

### Structures

For our experiments we use the humanoid robot ARMAR-III in conjunction with humans who demonstrate different aspects of the scenario as required for the bootstrapping processes which we implement.

We employ in the robot a (rather traditional) 3-layer architecture: Planning level, Mid-level, and sensorimotor Level (see e.g. [5]).

As data structure we define the so-called *Executable* consisting of:

1. planning operators forming a "plan" together with its
2. mid-level descriptors and
3. all perception and/or control information from the sensorimotor level for executing an action.

Some of these aspects are to some degree embodiment specific (most notably the control information), some others are not. Note, the structure of an Executable is essentially an extended version of an Object-Action-Complex (OAC), extended by the planning operator, where OACs had been introduced earlier by us to capture the relations between actions and objects [69, 39].

Essential to this work is that we use the concept of bootstrapping now in the same scenario at these three levels. The syntactic representations used to compute aspects of a given level are level-dependent where we have the following syntactic representatives:

1. planning operators,
2. syntactic structure of mid-level descriptors, and
3. perceptual (sensor) and control (motor) variables.

Therefore, we employ different (grammatical) scaffolds for the bootstrapping:

1. planning language,
2. semantic event chains (SECs<sup>1</sup> [2, 1]), and
3. sensorimotor feature/parameter regularity

from where the bootstrapping commences.

The general process of bootstrapping, however, is at all levels identical: Similarity at the level of the scaffold is used to infer, which entities can take the role of (can be replaced by) which other entities at the syntactic level. In this paper we will focus mainly on planning operators, objects, and motion trajectories as "entities for replacement", but we will discuss (see Discussion section) that structural bootstrapping is not limited to these aspects, but can be extended (for example) also to the bootstrapping of action-relevant attributes, etc.

## Prior knowledge

As bootstrapping relies on existing knowledge we have provided the robot with several (pre-programmed) Executables and we assume that the robot knows how to:

- pick up an object;
- put down an object;
- pour an ingredient;
- mix with an electric mixer.

In addition, robot has learned earlier to execute one apparently unrelated action, namely:

- wipe a surface with a sponge [DSEA14, 20, 25].

---

<sup>1</sup>Semantic Event Chains (SECs) act as a sequencer for actions and encode the sequence of touching and untouching events that happen until an action concludes. A more detailed description is given in the Methods section below.

Furthermore the robot has a certain type of object memory where it has stored a set of objects together with their roles, called the *repository of objects&attributes with roles (ROAR)*. This prior knowledge can be inserted by hand or by prior experience. It allows objects to be retrieved by their attributes, and attributes of novel objects to be inferred, based on proximity in a low-dimensional, Euclidean space in which both, objects and attributes, reside [XSP13].

The following entries exist in the ROAR:

- Sponge, rag, brush = objects-for-wiping with outcome: clean surface
- Mixer tool ends, whisks, sticks = objects for mixing with outcome: batter or dough.

Furthermore we have endowed the machine with a few recognition procedures:

- The robot can generate and analyze the semantic event chain (SEC) structures of observed (and own) actions by monitoring an action sequence using computer vision. Thus, the machine can recognize known actions at the SEC level [2, 1].
- The robot can recognize known objects (tools, ingredients, batter) using computer vision [9, 8, 10].
- The robot can explore unknown object haptically [12] and and extract object features such as deformability and softness [45, 55, DSEA14]

## Problem definition

The problem(s) to be solved by structural bootstrapping are defined by several stages as spelt out next:

Normal System Operation: If all required entities are present (mixer, ingredients, containers, etc.) the robot can make a plan of how to make batter and also execute it.

System Break-Down: Planning and execution will fail as soon as there is no mixer.

Alternative: The robot observes a human making batter by stirring the dough with a spoon.

Goal: The robot should find a way to understand the newly observed action and integrate it into its knowledge base and finally be able to also execute this.

Problem: The robot has no initial understanding of

- the planning requirements,
- the objects involved, and
- the movement patterns seen

in the newly observed stirring action. For example the robot does not know how to parameterize the rhythmic trajectory. Also, it does not know what a spoon is. Furthermore, the robot does not have any planning operator for stirring with a spoon in its plan-library.

Requirement (for the purpose of this study): The process of understanding the new action should happen without in-depth analysis of new actions constituents (hence without employing exploration based processes) but instead by using bootstrapping.

## 2.3 Methods - Short Summary

To not extend this paper unduely, methods are only described to the details necessary to understand the remainder of this paper. References to specific papers are provided where more details can be found.

EXECUTABLE		
Level	Name	Methods
1	Planning	CCG[58], PKS[48]
2	Mid-Level	SEC[1]
3	Sens.Mot.Level	ROAR[XSP13], DMP[35, 20, 67]

Table 2.1: Structure of an Executable

## Planning Methods

In this project, we are using the so-called Combinatory Categorical Grammars (CCGs) [58] to address the planning problem. CCGs are in the family of *lexicalized* grammars. As such they push all domain specific information into complex categories and have domain independent combinators that allow for the combination of the categories into larger and larger categories. As we have already alluded to, at the planning level, structural bootstrapping is a specialized form of learning new syntactic categories for known actions. A number of different methods have been suggested for this in the language learning literature [34, 41] for this project however we will be applying a variant of the work by Thomford [63]. However, we note that to do the kind of learning that we will propose it will be critical that the same syntactic knowledge, which is used by the system to plan for action, is also used to recognize the plans of other agents when observing their actions. This is not a new idea, however, there are very few AI planning and plan recognition systems that are able to use the exact same knowledge structures for both tasks.

Imagine that, as in our example, the high level reasoner knows about a plan to achieve a particular goal. It knows all of the actions that need to be executed, and for each action has encoded as CCG categories the knowledge necessary to direct its search for the plan. Further we suppose the same knowledge can be used to parse streams of observations of actions in order to recognize the plan being executed by others.

Now suppose the agent sees the execution of another plan that achieves the same goal. Let us assume that this new plan differs from the known plan in exactly one action. That is, all of the actions in the new plan are exactly the same as the actions in the known plan except for one action. Since the agent knows that the plan achieved the same goal, and it knows the CCG categories for each action that would be used to recognize the original plan, it is not unreasonable for the agent to assume that the new action should be assigned the same CCG category as its opposite action in the known plan.

If this addition is made to the grammar the agent now knows a new plan to achieve the goal and will immediately know both how to recognize others executing the plan and how to build the new plan for the goal itself (at the higher “abstract” level). The system will have performed structural bootstrapping at the planning level.

In this case, the system will have leveraged knowledge about the outcome of the observed plan being the same as the previously known plan, along with syntactic knowledge about how the previously known plan was constructed to provide new syntactic knowledge about how to construct and recognize the new plan.

## sensorimotor Methods

Sensory Aspects: Visual scenes are analysed to recognize objects and their attributes, measure movement trajectories, and record object poses.

Basic object and pose recognition is performed in a straight-forward way using pre-defined classes of the different objects which occur during the actions of “stir”, “wipe”, and “mix” and in addition adding some distractor objects (e.g., cups, knives, etc.). Any suitable method can be used for object detection, recognition, and pose estimation; such as edge-based, statistical shape representations [9, 8, 10, 62].

Another important aspect is object recognition for the construction of the repository of objects&attributes with roles (ROAR).

Our primary input for the ROAR consists of a table such as the one shown in Table 2.2.

	Attribute 1	Attribute 2	Attribute 3
Object A	$Value_{A,1}$	$Value_{A,2}$	$Value_{A,3}$
Object B	$Value_{B,1}$	$Value_{B,2}$	$Value_{B,3}$

Table 2.2: ROAR encoding

Objects and attributes are (discrete) labels; values can be categorical, discrete or continuous. Examples of objects are "bowl" or "knife"; examples of attributes are "cuts", "food", "is elongated", "gripper orientation for grasping", "fillable", etc. We then use Homogeneity Analysis to project objects and (attribute) values into the same, low-dimensional, Euclidean space (the *ROAR space*) [XSP13]. This projection is set up such that:

- Objects that exhibit similar attribute Values are located close together,
- Objects that exhibit dissimilar attribute Values are located far apart,
- Objects-as-such are close to their attribute Values.

Euclidean neighborhood relations allow us to make the following general types of inference:

- Attribute value prediction: Say, we have an object of which we know some but not all attribute Values. We can predict missing attribute Values by projecting the object into the ROAR and examining nearby attribute Values.
- Object selection: Say, we have a set of required attribute values. We can find suitable objects in the vicinity of these Values in the ROAR.

Note we cannot generally expect that very complex object/attribute relations will be faithfully represented in a low-dimensional Euclidean space. While we are currently working on more powerful representations for such relations, this is a complex research issue [XSP13, 28, 42, 43, 64]. For us the ROAR is at the moment just a viable way forward, which allows us to demonstrate different aspects of structural bootstrapping.

Motor Aspects: Trajectory information is encoded by Dynamic Movement Primitives (DMPs), which were proposed as an efficient way to model goal-directed robot movements [35]. They can be applied to specify both point-to-point (discrete) and rhythmic (periodic) movements. A DMP consists of two parts: a linear second order attractor system that ensures convergence to a unique attractor point and a nonlinear forcing term. The forcing term is normally given as a linear combination of basis functions that are defined along the phase of the movement. The basis functions are either periodic or nonzero only on a finite phase interval. The type of basis functions decides whether the DMP defines a discrete or a periodic movement. DMPs have many favorable properties, e.g. they contain open parameters that can be used for learning without affecting the overall stability of the system, they can control timing without requiring an explicit time representation, they are robust against perturbations and they can be modulated to adapt to external sensory feedback [35, 54].

## Methods for the Mid-Level: Semantic Event Chains (SECs)

Semantic Event Chains encode in an abstract way the sequence of events (see below) that occur during a complex manipulation. They are used for two purposes: (1) Every event provides a specific temporal anchor point, which can be used to guide and temporally constrain the above described scene and motion analysis steps. And (2) the SEC-table itself (see Fig. 2.1 b), is used to define the mid-level of an Executable.

Fig. 2.1 shows the corresponding event chains extracted for a *stirring* action. SECs basically make use of image sequences (see Fig. 2.1 a, top) converted into uniquely trackable segments. The SEC framework first interprets the scene as undirected and unweighted graphs, nodes and edges of which represent image segments and their spatial touching or not-touching relations, respectively (see Fig. 2.1 a, bottom). Graphs hence become semantic representation of the relations of the segments (i.e. objects, including

hand) presented in the scene in the space-time domain. The framework then discretizes the entire graph sequence by extracting only the main graphs, which are those where a relation has changed (e.g. from not-touching to touching). Each main graph, thus, represents an essential primitive of the manipulation. All extracted main graphs form the core skeleton of the SEC which is a sequence table (the SEC-table), where columns correspond to main graphs and rows to the spatial relations between each object pair in the scene (see Fig. 2.1 b). SECs consequently extract only the naked spatiotemporal relation-patterns and their sequentiality, which then provides us with the essence of an action, because SECs are invariant to the followed trajectory, manipulation speed, or relative object poses.

Columns of a SEC represent transitions between touching relations. Hence, they correspond to decisive temporal moments of the action and, consequentially, they allow now to specifically pay attention "at the right moment when something happens" to additional action relevant information (such as objects, poses, and trajectories). Fig. 2.1 (c-d)) illustrate syntactic elements of the manipulation. Manipulated objects, e.g. spoon and liquid, are extracted from the rows of event chains, i.e. from the nodes of the main graphs. Temporal anchor points provided by SECs can also be used to segment the measured hand-trajectory into parts for further analysis.

## 2.4 Implementing Structural Bootstrapping at different levels

Figure 2.2 shows a schematic representation of the bootstrapping processes implemented here. A known plan A (left) consists of a set of planning operators (black bars) and each has attached to it an Executable consisting of the planning operator itself, a mid level descriptor and sensorimotor level information. The plan, being executed, also has a certain "outcome", which can be considered as the goal of this action sequence. An observed plan B (right) of a similar action (with similar goal), will normally consist of many planning operators which are identical or highly similar to the ones of the known plan and also the outcome will be similar or same. Still some planning operators may be dissimilar and hence unknown to the agent (white bars). In the same way, individual newly observed Executables (right) may contain unknown components (white). The goal of bootstrapping is to fill in all this missing information. To the end, first (1) the respective entities, Plans (1A) or Executables (1B), will be compared at an "outer", grammatical level to find matching components. This way, in the second step one can try to infer the respective missing entities, planning operators (2A) or components of the Executables (2B).

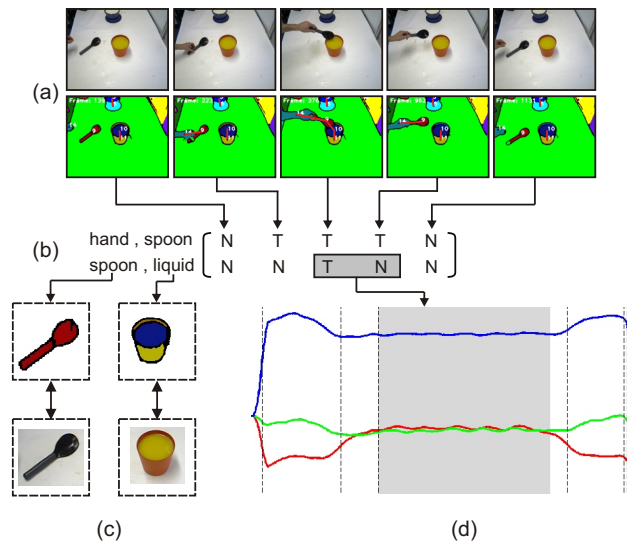


Figure 2.1: A real action scenario: "Stirring liquid with a spoon". (a) Sample original key frames with respective segments and graphs. (b) Corresponding SEC where each key frame corresponds to one column. Possible spatial relations are N, T, and A standing for "Not-touching", "Touching", and "Absence", respectively (*A* does not happen here). Shaded box shows a sample relational transition. (c) Object identities derived from segments (d) Complete trajectory information for the hand. Trajectory segment for the time-chuck covered by shaded box in (c) is indicated in gray color.

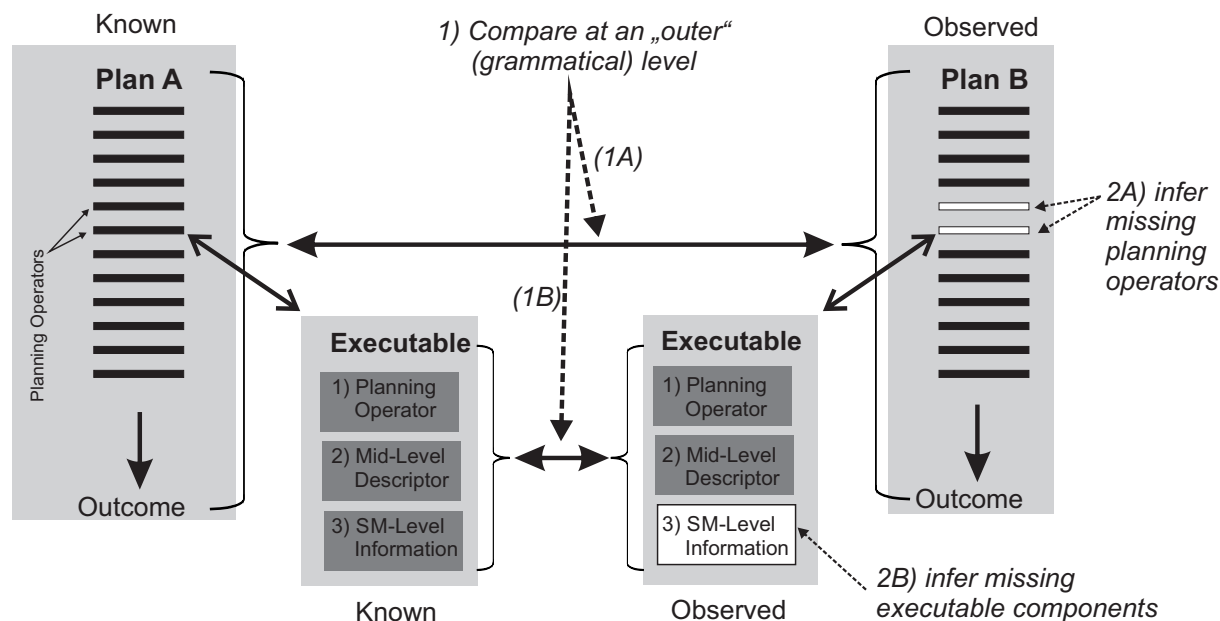


Figure 2.2: Schematic of structural bootstrapping.

Hence, a central statement is that structural bootstrapping always "propagates downward". It uses type-similarities of entities from one level above to define the missing syntactical elements of the currently queried (lower) level! Plan similarities are used to infer planning operators, Executable similarities to infer Executable parameters such as objects, trajectories, forces, poses, and possibly more.

The main difficulty for implementing structural bootstrapping is to define appropriate scaffolds on which the bootstrapping can be based where – as described – the goal is to create novel information by generative processes which compare existing knowledge with newly observed one, without having to perform an in-depth analysis.

## Planning Level

The existing plan of making batter with a mixer is compared to the observed sequence of actions during making batter with a spoon. Due to the fact that all sub-actions, but one, are identical between known-action and new-action the agent can infer that the unknown sub-action (stirring with a spoon) is of the same type as its equivalent known sub-action (mixing with a mixer). Hence the grammatical comparison of known with unknown action renders a new (syntactic) planning operator entry for the unknown sub-action. This process is very similar to syntactic bootstrapping as observed in child language acquisition. A semantic element enters here due to the same outcome of both actions being recognized as batter. We use CCG as our planning language and we employ the PKS planner [48] for the actual planning processes of ARMAR-III.

The actual inference process makes use of the similarity of known plan with newly observed plan, where in our example all but one action are identical.

Figure 2.3 shows the comparison between a known (and executable) plan on the left and an observed new one (right). Structural (grammatical) one-by-one comparison shows that there is just one unknown planning operator present. When the plan recognizer is run on the observed plan it would result in the following explanation of those observations with the highest probability:

```
[ addIngC(left, liquid1, beaker, mixingbowl),
  addIngC(left, liquid2, cup, mixingbowl),
```



Original Plan	Observed Plan
<pre> testName: xpermix; initialState: [ ]; observations: [   pickA( left, beaker, t ),   pourA( left, liquid1, beaker, mixingBowl ),   placeA( left, beaker, t ),   pickA( left, cup2, t ),   pourA( left, liquid2, cup2, mixingBowl ),   placeA( left, cup2, t ),   pickA( right, mixer1, t ),   mixA( mixer1, liquid1, liquid2, mixingBowl ) ]; </pre>	<pre> testName: xpermixnew; initialState: [ ]; observations: [   pickA( left, beaker, t ),   pourA( left, liquid1, beaker, mixingBowl ),   placeA( left, beaker, t ),   pickA( left, cup2,t ),   pourA( left, liquid2, cup2, mixingBowl ),   placeA( left, cup2, t ),   pickA( right, UNKNOBJ, t),   UNKNACT( UNKNOBJ, liquid1, liquid2, mixingBowl ) ]; </pre>

Figure 2.3: Comparing known with observed plan. The arrow indicates where there is a novel, unknown planning operator found in the new plan. This is also associated with an, as yet, unknown object (the spoon).

```

pickC(left, UNKNOBJ, table),
UNKNACT(left, UNKNOBJ, liquid1, liquid2, mixingbowl)]

```

Note, the category name for the previously unseen action is simply denoted as UNKNACT. This is a special purpose category used to complete the explanation when we have an action that has never been seen before.

Now the agent has been told (or can observe) that the observed plan is a plan that achieves makeBatterC (making batter), and we will assume that all of the actions in the observed plan are relevant to the plan. The agent's job is to infer a category to replace UNKNACT that allows the completing of the parse. If the agent wants to build a category to assign to the unknown action that will result in a complete plan with the goal of makeBatterC, all it needs to do is walk the explanation from right to left collecting the categories and adding them to the complex category in order. This will result in the unknown action being given the following category:

```

action: UNKNACT(hand, UNKNOBJ, ingredient, ingredient, bowl)
  [ ((( makeBatterC( 2, 3, 4 )))\
    {addIngC( 0, 2, obj(1), 4)}\
    {addIngC( 0, 3, obj(2), 4)}\
    {pickC( 0, 1, table(1)) } ] ];

```

Note the agent also infers the types and co-reference constraints for the basic categories arguments from the plan instance. In the above definitions we have denoted those arguments to the basic categories by numbers indicating when an argument is bound to the same argument as the action. (i.e. All references to "0" in the category refer to the hand used in the action because it is the zeroeth argument for the action. Likewise all reference to "4" in the category refer to the bowl argument of the action since it is the fourth argument.)

This category would represent the most restrictive hypothesis about the plan structure since it will require both that the actions be executed in the same order (and we know the ingredients can be added to the plan in either order) and that all of the arguments that co-refer in the example plan must co-refer in future instances. In this case, it would require that the same hand be used for all of the ingredient adding and mixing which we know to be overly restrictive.

If we compare the new category to the category for the known mix action (mixA), we can see that the only differences are exactly in these overly restrictive areas:

<p>A) <b>Picking up</b></p> <table border="0"> <tr><td>Hand, Beaker</td><td>0</td><td>1</td><td>1</td></tr> <tr><td>Beaker, Table</td><td>1</td><td>1</td><td>0</td></tr> </table>	Hand, Beaker	0	1	1	Beaker, Table	1	1	0	<p>B) <b>Putting down</b></p> <table border="0"> <tr><td>Hand, Beaker</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>Beaker, Table</td><td>0</td><td>1</td><td>1</td></tr> </table>	Hand, Beaker	1	1	0	Beaker, Table	0	1	1																				
Hand, Beaker	0	1	1																																		
Beaker, Table	1	1	0																																		
Hand, Beaker	1	1	0																																		
Beaker, Table	0	1	1																																		
<p>C) <b>Pouring</b></p> <table border="0"> <tr><td>Hand, Beaker</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr> <tr><td>Beaker, MixBowl</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>Beaker, Liquid2</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>MixBowl, Liquid2</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr> </table>	Hand, Beaker	1	1	1	1	1	Beaker, MixBowl	0	1	1	1	0	Beaker, Liquid2	1	1	1	0	0	MixBowl, Liquid2	0	0	1	1	1	<p>E) <b>Mix (with Mixer)</b></p> <table border="0"> <tr><td>Hand, Mixer</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>Mixer, Dough</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> </table>	Hand, Mixer	0	1	1	1	0	Mixer, Dough	0	0	1	0	0
Hand, Beaker	1	1	1	1	1																																
Beaker, MixBowl	0	1	1	1	0																																
Beaker, Liquid2	1	1	1	0	0																																
MixBowl, Liquid2	0	0	1	1	1																																
Hand, Mixer	0	1	1	1	0																																
Mixer, Dough	0	0	1	0	0																																
<p>D) <b>Wipe (with Sponge)</b></p> <table border="0"> <tr><td>Hand, Sponge</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>Sponge, Surface</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> </table>	Hand, Sponge	0	1	1	1	0	Sponge, Surface	0	0	1	0	0	<p>F) <b>Stir (was UNKNACT) with Object*</b> <b>Unknown SEC</b></p> <table border="0"> <tr><td>Hand, Object</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> <tr><td>Object, Dough</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr> </table>	Hand, Object	x	x	x	x	x	x	Object, Dough	x	x	x	x	x	x										
Hand, Sponge	0	1	1	1	0																																
Sponge, Surface	0	0	1	0	0																																
Hand, Object	x	x	x	x	x	x																															
Object, Dough	x	x	x	x	x	x																															
<p>G1) <b>Stir (was UNKNACT) with Object*</b> <b>SEC from one observation</b></p> <table border="0"> <tr><td>Hand, Object</td><td>0</td><td>1</td><td>1</td><td>1</td><td style="background-color: #cccccc;">1</td><td style="background-color: #cccccc;">0</td></tr> <tr><td>Object, Dough</td><td>0</td><td>0</td><td>1</td><td>0</td><td style="background-color: #cccccc;">1</td><td style="background-color: #cccccc;">0</td></tr> </table>	Hand, Object	0	1	1	1	1	0	Object, Dough	0	0	1	0	1	0	<p>G2) <b>Stir (was UNKNACT) with Object*</b> <b>SEC from two observations</b></p> <table border="0"> <tr><td>Hand, Object</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>Object, Dough</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr> </table> <p><small>*Object = "UNKNOBJ", before object specification Object = "spoon" after object specification</small></p>	Hand, Object	0	1	1	1	0	Object, Dough	0	0	1	0	0										
Hand, Object	0	1	1	1	1	0																															
Object, Dough	0	0	1	0	1	0																															
Hand, Object	0	1	1	1	0																																
Object, Dough	0	0	1	0	0																																

Figure 2.4: Several important SECs, which occur during the different actions. Headlines (bold lettering, like "Picking up", etc.) denote the type-specifiers of the different SECs. Note, sometimes objects can change. E.g. "Beaker" can be replaced by "Cup2". **A-E)** error-free archetypical SECs from known actions. **F)** So-far unspecified SEC. **G)** SECs from the unknown action extracted from observation of the human performing it. Hence these SECs might contain errors. **G1)** one observed case, **G2)** two observed cases. (In human terms: G1 corresponds to a case where the spoon had intermittently been pulled out from the dough (grey box), whereas for G2 it always remained inside until the action terminated.)

1. The ordering of the categories for the add ingredient steps. The known category is more general allowing the ingredients to be added in any order while the new learned category has a required order.
2. The co-reference constraints are less restrictive in the known category. (Note the numbers indicating, which hand is to be used in the addIngC, are not the same so the plan would not enforce that the same hand be used.)

At this point, on the basis of the structural information provided by the parse and the action grammar, the agent has inferred that "UNKNACT" is equal to (or at least very similar to) "mixA" and the information can be entered directly into the planning grammar of the agent and forms the top-level of the corresponding new executable. We will, for convenience, from now on name it: "stir", hence we set:

UNKNACT:=stir.

While we have now added a new action to the planning grammar, still there is massive information lacking for designing the complete (new) executable for "stir", for example there is as yet no understanding existing about the UNKNOBJ (the spoon) and nothing is known about several other mid- and low-level descriptors.

## Mid-Level

At the mid-level, we need to define the correct SEC for "stir". Figure 2.4 shows SECs for several actions where (F) represents the so-far unknown SEC for "stir". Please, ignore panels (G) for a moment.

Structural bootstrapping at the mid-level uses as the "outer", grammatical scaffold the type-similarity of the planning operators (here "stir" and "mix") ascertained above. Hence we know that UNKNACT=stir.

Following this conjecture the agent can now with a certain probability assume that so-far unknown SEC for "stir" ought to be identical (or very similar) to the known one from "mix" and use the "mix"-SEC to define the mid-level (the SEC) for the Executable of "stir". The arrow indicates that the SEC from panel (E) should just be transferred to fill the unknown SEC in (F) with the same entries.

There is a second line of evidence which supports this. Panels (G1) and (G2) represent the actually observed SECs of the stirring action here from a total of three observations of a human performing this. The SEC in panel (G1) had been observed once and the other twice. By comparing these SECs, the robot can with some certainty infer that the transfer of (E) to (F) was correct, because the more often observed SEC in (G2) corresponds to it, while the SEC from panel (G1) might be noisy as it is a bit different. As shown in an earlier study [2, 1], more frequent observations are likely to confirm this even more, but were not performed with the current setup.

## Sensorimotor Level

Bootstrapping at the control level is used by the agent to find out how stirring is actually done (motion patterns), what the meaning of "UNKNOBJ" is, and which other objects might have a similar meaning. Before going into details we can quickly state that at the sensorimotor level several bootstrapping processes can be triggered. We note that bootstrapping is a probabilistic process and things can go wrong, too. One such example is, hence, also included. We find that the following processes are possible:

### 1. Motion

- (a) Bootstrapping from SEC-similarities ["wipe" and "stir"] to define the motion patterns for "stir".

### 2. Objects

- (a) Bootstrapping from SEC-similarities ["wipe" and "stir"] into the new action. Here arriving at a false conjecture that "sponges" could be used for mixing.
- (b) Bootstrapping from SEC-similarities ["mix" and "stir"] from the repository of objects&attributes with roles (ROAR) into the new action seeking different objects that could potentially be used for mixing.
- (c) Bootstrapping from SEC-similarities ["mix" and "stir"] from the new action into the ROAR, entering the "spoon" into the category of objects for mixing.

To address the sensorimotor level the agent has to bootstrap from the mid-level downwards. It can do this by comparing the *type-similarities* of the different SECs. For this essentially one calculates a sub-string comparison of the rows and columns between one SEC and any other [2, 1]. We obtain that "stir" and "mix" as well as "stir" and "wipe" are 100% type-similar (compare panels D, E, and G2 in Figure 2.4), whereas "stir" and "pour" are only 52% similar, etc. Thus, the agent can infer that syntactical elements from "mix" and "wipe" might be used to define missing entities at the sensorimotor level of the Executable.

### 1a) Motion: Bootstrapping from SEC-similarities "wipe" and "stir" into the new action for completing motor information

Here we make use of the fact that the SEC for stir is very similar to the known one from wipe. Figure 2.5 shows the SECs and the different trajectories recorded from human observation for both actions. Note that for "wipe" the complete motor encoding is known and provided by the respective DMP parameters.

We have in our data-base the following description for "wipe": Since wiping is essentially a rhythmic movement, we use periodic dynamic movement primitives to specify the required behavior [25]. Periodic DMPs are defined by the following equation system [35]

$$\dot{z} = \Omega\alpha_z(\beta_z(g - y) - z) + f(\phi), \quad (2.1)$$

$$\dot{y} = \Omega z, \quad (2.2)$$

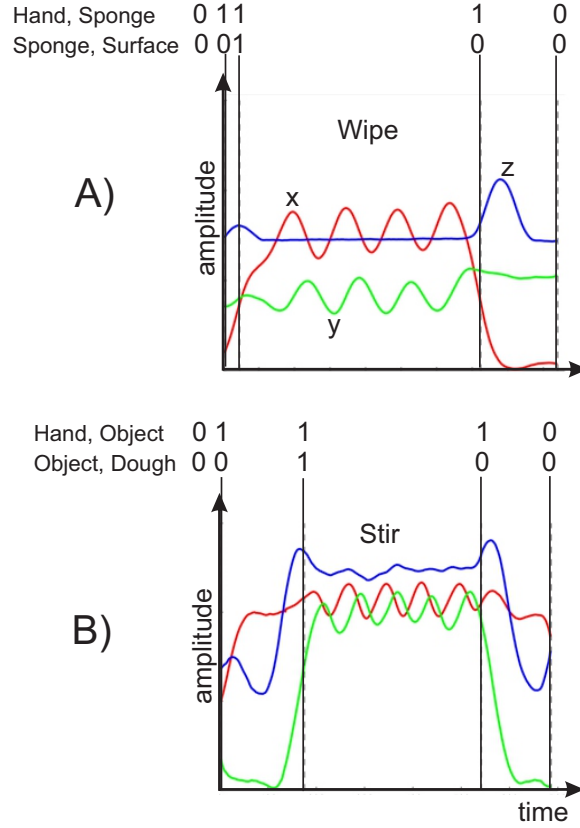


Figure 2.5: Bootstrapping motor information. SECs (top) and trajectories (bottom) for  $x$ ,  $y$ , and  $z$  coordinates in task space are shown for (A) wipe and (B) stir.

In the above equations,  $g$  is the anchor point of the periodic movement. The nonlinear forcing term  $f$  is defined as

$$f(\phi, r) = \frac{\sum_{i=1}^N w_i \Psi_i(\phi)}{\sum_{i=1}^N \Psi_i(\phi)} r, \quad (2.3)$$

$$\Psi_i(\phi) = \exp(h_i \cos(\phi - c_i) - 1),$$

where the phase  $\phi$  is given by

$$\dot{\phi} = \Omega. \quad (2.4)$$

Here we assume that a complete parameterization of the DMP for wiping has been learnt from earlier experiences. Given this the DMP can be easily modulated by changing:

- the anchor point  $g$ , which translates the movement,
- the amplitude of oscillation  $r$ ,
- the frequency of oscillation  $\Omega$ .

These variables can be used to immediately adapt the movement to sensory feedback.

Bootstrapping progresses by using the concept of temporal anchor points, which are those moments in time when a touching relation changes (from 0 to 1, or vice versa). These anchor points divide the trajectories in a natural way (shown by the vertical lines in the figure.)

Bootstrapping now *just copies* the complete DMP information from "wipe" to the Executable of "stir" between the respective anchor points only leaving the constraint-parameters (e.g. amplitude) open as those are given by the situation (mainly the size of the bowl wherein to stir). Thus, the agent assumes that it can use the motor encoding from "wipe" in an unaltered way to also perform "stir". We know from own

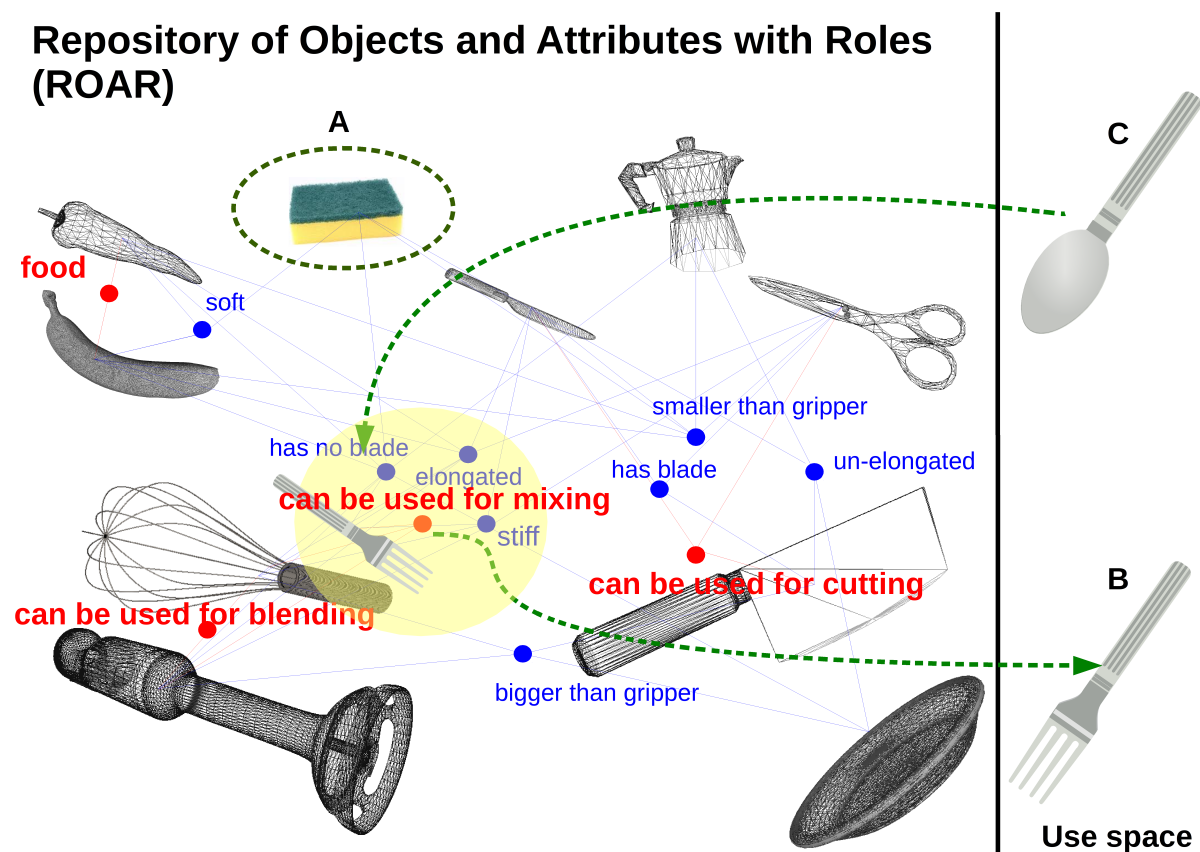


Figure 2.6: Bootstrapping object information. Graphical rendering of the repository of objects&attributes with Roles (ROAR). Depicted are the metric distances between the different objects and the attribute values that describe their respective roles. **A)** The sponge is located far from the attribute value "can be used for mixing". **B)** Bootstrapping allows inferring that a fork, found close to the "mixing" attribute value, could be used also for "stir", as "mix" and "stir" are at the SEC-level type-similar. **C)** Following this SEC-similarity, a novel object (spoon) with unknown "mixing" attribute may be hypothesized useful for mixing by the ROAR and also due to other, known attribute values (such as shape, stiffness, and SEC characteristics of known, observed actions).

experience that this largely holds true. Here we can also clearly see the advantages of bootstrapping: we do not need any process that *extracts and generalizes* motor information from the observed example(s) of "stir" (a process which could be more tediously performed by methods from imitation learning [13, 4, 17]. Instead we just copy! Clearly, the agent - like any young child - will have to ground this by trying out the stirring action. It will possibly then have to adjust the force profile, which is likely to be much different for wipe and stir. Still, all this is faster than learning the required motor pattern in any other way. The benchmark experiments below show this clearly.

#### 2a) Objects: Bootstrapping from SEC-similarities "wipe" and "stir" into the new action for object use

The SEC-similarities between "wipe" and "stir" allow the agent to also (wrongly!) infer that the object for wiping (sponge) should be suitable for stirring, too. Note this may seem unexpected but can happen during any bootstrapping process due to its probabilistic nature. The use of just one single scaffold (here the SECs) is not strong enough to allow rigorously excluding such false conjectures. For this the agent needs to integrate additional information and, due to the fact that there is a repository of known objects&attributes with roles (ROAR), it can indeed obtain evidence that there has been an error.

The agent knows that "stir" and "mix" are at the mid-level (SEC) type-similar action. It finds, however,

that sponges are clearly outside the cluster of objects for mixing (Figure 2.6 A). This lowers the probability substantially that sponges should be used for mixing/stirring actions.

Interestingly, children will many times indeed over-generalize and use "unsuitable" objects for an intended action [32]. It is unknown how the brain represents this, but – clearly – their representation does apparently not yet contain the fine grained-ness of an adult representation.

### **2b) Bootstrapping from SEC-similarities "mix" and "stir" from the ROAR to find other suitable objects**

Here the agent falls back (again!) on the similarity of the new SECs of "stir" with the known one of "mix". Due to this similarity, the agent knows that appropriate objects for the novel action might be found in the cluster of "objects for mixing" in the repository of objects&attributes with roles. Hence it can ask the repository for a tool suitable for mixing and maybe locate it somewhere else on the table. Clearly this process will lead to an action relevant result only in those cases where the agent actually find such an object within reach. Then it can try to use this object for stirring, too. Again we can draw similarities to our own behavior. Generally this type of tool-replacement is found for a wide variety of actions where we "define" the tool according to its planned use. Our own generalization properties may here go far beyond what the ROAR offers to our artificial robotic agent, which is evident from situations where we "abuse" objects for entirely different purposes.

### **2c) Bootstrapping from SEC-similarities "mix" and "stir" from the new action into the ROAR to create a new entry**

In the last step, the agent can perform one more bootstrapping procedure to augment the repository of objects&attributes with roles. For this it analyzes the outcomes of the actions realizing that batter is obtained from "mixing" and also from the unknown action of "stirring".

Thus, the agent can enter the new observed tool (spoon) into the ROAR and can then – by virtue of its resulting position in the ROAR – infer other, unobserved attribute values (uses), which is a bootstrapping effect. This way the repository will be extended by a novel entry following a single-shot experience. This step, however, does require a parametrization of the new object according to the features used for the ROAR.

## **2.5 Robotic Implementation**

Note, the actual bootstrapping processes happen "inside the machine" and any demonstration will, thus, only show that "the robot can do it now". Hence, it is far more useful to provide quantitative results on performance gain by bootstrapping, which will be presented in the next sections, below.

Still, a complete robotic implementation of these processes is currently being performed using the ARMAR-III robot. For brevity, we will here show one central part of this implementation demonstrating the required transfer of human action knowledge (Fig. 2.7 A) onto the robot. This is the initial step needed to set up action knowledge in the machine before any bootstrapping can happen. The robot acquires here the knowledge to perform mixing with a mixer.

To better be able to extract object relations we have here used the marker-based motion capture system (VICON) from which we immediately get error-free Semantic Event Chains (Fig. 2.7 B). The complete action relevant information is extracted at the respective key frames and encoded into the required Executables (Fig. 2.7 C), which can be used by the robot to reproduce this action (Fig. 2.7 D). The complete experiment is described in [68].

## **2.6 Benchmark Experiments**

In the following we will show some experiments from our scenario demonstrating the power of structural bootstrapping for example the speed-up as compared to conventional, exploration based learning methods but also the accuracy of the object attribution methods used in the bootstrapping process.





Figure 2.7: Transfer of action knowledge from human to robot. **A)** Human demonstration, **B)** SEC depicted by ways of its key frame graphs, which show which objects touch which other objects (edges) during human execution. **C)** Abbreviated Executables **D)** Robot execution.

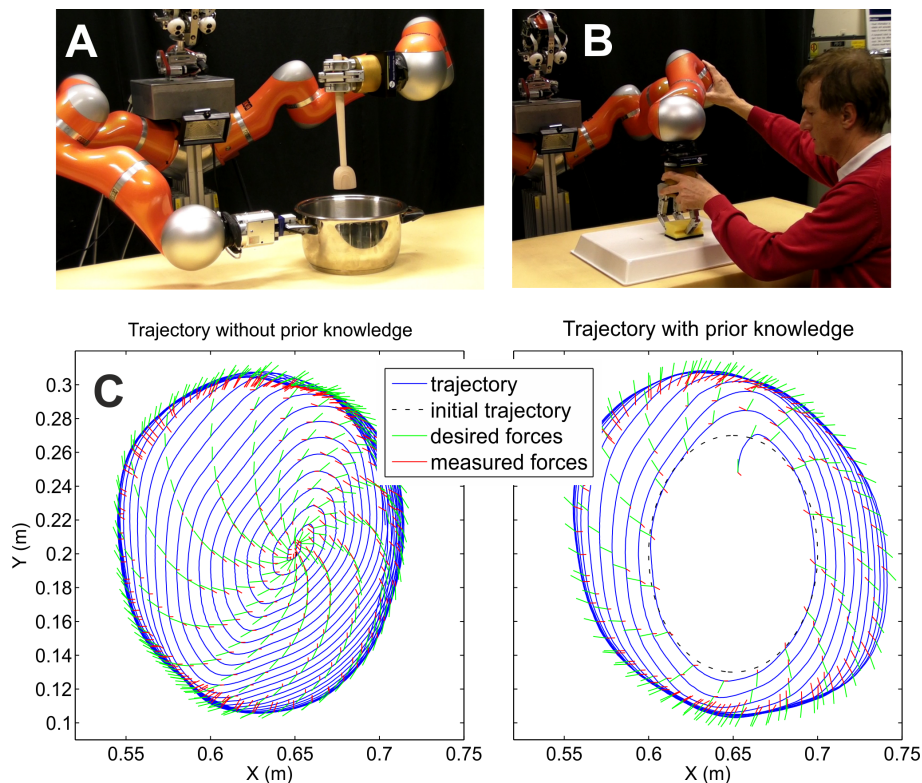


Figure 2.8: Benchmark experiment demonstrating the gain of learning speed when bootstrapping motion trajectories. **A)** Experimental setup and **B)** demonstration of wiping. **C)** Learning of stirring behavior without prior knowledge and **D)** adaptation of wiping to stirring. The desired and actual forces are shown with red and green vectors.

## Bootstrapping Motion - Measuring Learning Speed

Our setup for learning of stirring behavior is shown in Fig. 2.8 A. It is composed of two KUKA LWR robots, both equipped with Barred hands. The task is to learn how to stir in a metal pad of diameter of 21 cm using wooden spoon. The position, size and shape of the pad are not known in advance. To define a criterion function for motion learning, we specify the force  $\mathbf{F}_d$  with which the robot should move along the edge of the pot.

We considered two cases: 1) learning without any prior knowledge about the stirring trajectory and 2) learning where the adaptation process is initialized with wiping trajectory. The wiping trajectory is obtained by imitation learning (Fig. 2.8 B). We used periodic DMPs to represent the movement [26] and apply a Repetitive Control (RC) algorithm [16, 27]. The RC algorithm iteratively adapts the current estimate of the stirring behavior to achieve the desired outcome as defined by the desired contact force. Task performance is improved with each repetition and eventually, the required behavior is achieved regardless of the initial estimate of the stirring trajectory.

Fig. 2.8 C,D show the progress of learning in  $x$ - $y$  plane for both cases. The robot learned the policy in approximately 15 cycles without any prior knowledge about the trajectory and in approximately 7 cycles with prior knowledge taken from wiping motion. This demonstrates in a practical example that low-level sensorimotor learning can significantly benefit from the initialization provided by the semantic understanding of the task.

Note that in the specified scenario, the direction of adaptation is provided by the information about the desired contact force. We can expect that the difference between the two approaches would be even bigger if model-free methods such as reinforcement learning were used.



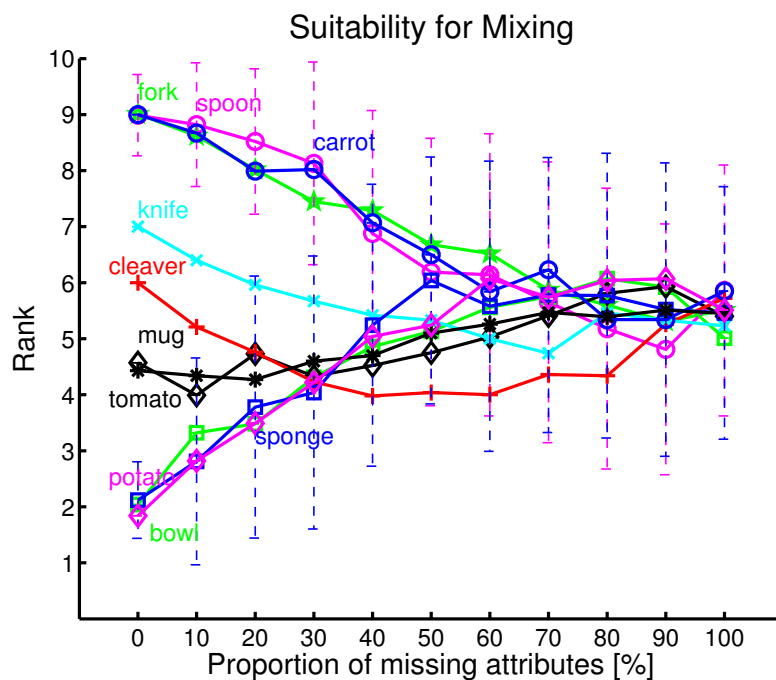


Figure 2.9: Estimated suitability of 10 different objects for mixing, ranked by the ROAR. With fully known attributes, the ROAR consistently considers the spoon, the fork and the carrot as useful mixing tools, and the sponge, potato and bowl as useless. This consistency degrades gracefully with increasing percentage of missing attributes. Each column of the graph represents ranks averaged over 100 runs. Error bars give standard deviations for our two objects of interest (sponge and spoon); those of the other objects are similar but not shown to reduce clutter.

## Bootstrapping Objects - Measuring Success

Trivially immediate object replacement, using the ROAR as suggested for cases 2a, b, and c above, will always be faster than finding an appropriate object by exploration, but will the ROAR find the correct object?

We evaluate the capacity of the ROAR to predict the suitability of given objects for mixing, similar to the scenario above. To this end, we created a database of 10 objects as listed in Fig. 2.9. Each object is characterized by 10 binary attributes describing its properties (such as shape and stiffness) and usage categories (such as “container” or “mixing tool”), some of which may be unknown. The ROAR ranks objects according to their estimated suitability for mixing. Fig. 2.9 shows the suitability of the 10 objects as a function of the proportion of missing attribute values. Each column of the graph represents results averaged over 100 runs.

For each run and for each proportion of missing attributes, the designated proportion of object attribute values is randomly chosen from the complete database; these are set to **unknown**. On the resulting copy of the database with missing attribute values, homogeneity analysis is performed (see “Methods”), producing a ROAR. Objects are ranked by the ratio of their Euclidean distances to the “mixing tool”=**true** vs. “mixing tool”=**false** attribute values [XSP13].

With fully known attributes, the ROAR consistently ranks the spoon, the fork and the carrot as the most useful mixing tools, while the sponge, potato and bowl rank last. This consistency degrades gracefully with increasing percentage of missing attributes.

## 2.7 Discussion

A central issue for the development of autonomous robots is how to quickly acquire new concepts to be used for planning and acting, for example learning to stir in a pot, which is a relatively complex

manipulation sequence. Reinforcement learning or association-based learning methods have been applied to this and related problems but are usually too slow to achieve this efficiently. Thus, one often combines them with supervised learning. Here, especially the paradigm of learning from demonstration has often been successfully employed [53, 13, 47, 19, 15, 17] also because we (humans) are rather good at this. Still none of these methods is generative in the sense that it would take existing knowledge to generalize it into novel unexplored domains. At best one finds in-domain generalization, such a generalizing across different trajectories of the same action-type [67, 46, 38, 37].

This may not make us wonder, though! After all, generative learning is clearly an advanced cognitive trait and the gap between human performance and the current capabilities of machines is exceedingly wide. The central problem seems to be that – on the one hand – one has clear evidence that such processes do indeed happen during human (infant) learning [50, 66, 14, 65, 49], but – on the other hand – no one knows how; let alone, no one seems to have convincing ideas of how to do this with artificial agents either.

This was also the main challenge which we faced in this study: How can one develop a set of generative processes that use an "outer", grammatical representation to bootstrap missing "inner", syntactic elements, preferably at different levels of a cognitive architecture (planning, mid-level, and sensorimotor level). Furthermore our goal was to define such processes in a rigorous, algorithmically implementable way, to actually allow a robot to do this.

Language development did offer us a useful analogon on which we could build in this study. Semantic and syntactic bootstrapping [50, 56, 29, 23, 44, 22, 30, 66, 24], by which a child infers the meaning of unknown words using prior knowledge both rely on a general principle which we also used here: Grammar provides a solid scaffold for the probabilistic reasoning required for such inferences. While this was a helpful notion, still it remained unclear what the grammatical elements of an action sequence are (see [33] for a set of articles related to action-verb learning in children).

## Bootstrapping at the planning level

Planning languages and planning operators can be rather directly linked to the "language of action". Since the earliest days of AI research on symbolic planning, the ideas of abstraction and hierarchy and the decomposition of high level plans into lower level plans has been seen as central to efficiently building plans [61, 52]. Many current researchers view knowledge of such plan hierarchies as "domain specific control knowledge", that is knowledge of how to construct plans that is specific to individual domains. This kind of knowledge has traditionally been encoded in Hierarchical Task Networks (HTNs) [21]. A formal relationship has been shown between HTNs and other similar plan structures and Context Free Grammars (CFGs) that are used extensively in natural language processing, formal grammar theory and theory of computation [21]. Here essentially we were representing our search control knowledge as a grammar and thereby it becomes quite clear how to extend the idea of syntactic&semantic bootstrapping to the symbolic planning domain. In this case, our objective was to learn the "syntactic knowledge" that encodes how to effectively build a new from an old plan.

Thus, for us it was relatively straight forward to implement structural bootstrapping at the planning level. The similarities of two plans allows inferring missing planning operator information (Fig. 2.3). But this addresses only the highest, the symbolic, level of an action sequence. It is for robotics totally useless to utter commands like "pour liquid", without also providing the required, complex sub-symbolic information of how to actually do this.

## The problem of mid-level scaffolds

Hence, more was needed to bridge the gap from symbols all the way down to the control signals of the robot motors. In some earlier studies we had introduced the Semantic Event Chain (SEC) as a possible mid-level descriptor for manipulation actions [2, 1, ATV<sup>+</sup>13]. The SEC framework analyzes the sequence of changes of the relations between the objects that are being manipulated by a human or a robot. Consequently, SECs are invariant to the particular objects used, the precise object poses observed, the actual trajectories followed, or the resulting interaction forces between objects. All these aspects are allowed to change and still the same SEC is observed which, thus, captures the essence of the action as demonstrated in several action classification tests performed by us [1, ATV<sup>+</sup>13].

It turned out that SECs offer two important aspects which make them good scaffolds for the bootstrapping of lower-level sensorimotor information.

1. SECs provide temporal anchor points, annotating in an action when "something decisive" has happened. This allows the chunking of an action and thereby provides the agent with a means to perform motor-pattern replacement (here wipe for stir), because "it knows" when to do the replacement.
2. Above we stated that SECs are invariant to the particular objects used. This is also essential for the bootstrapping. Only through this, object replacement is immediately permitted as the scaffold (the SEC) is not bound to particular objects as long as the chosen-one performs the same role (performs the same NT, TN transitions).

Ideas to utilize (spatial) relations to approach the semantics of actions first appeared in 1975. Badler [11] used directed scene graphs where each node identifies one object. Edges represent spatial information (e.g., LEFT-OF, IN-FRONT-OF, etc.) between the objects. Based on the object's motion patterns, events are defined. Taken together this then represents an action. This, approach came to a stand-still, though because only now powerful enough image processing methods are available to provide the required information.

Even by now there are still only a few approaches towards semantic action understanding [57, 36, 72], often based on very complex activity graphs [57]. In [36], segmented hand poses and velocities are used to classify manipulations based on a histogram representation and using support vector machine classifiers for categorization of the manipulated objects. Others [72] introduced a visual semantic graph to recognize the action consequences based on changes in the topological structure of the manipulated objects.

In the context of the current study, potentially all these different approaches could be used as mid-level scaffolds, because they are based on the fact that the human action space is rather limited [70] and we are in fact not restricted by the here used SECs.

## Bootstrapping low-level information

Any of these mid-level scaffolds could thus be used to guide bootstrapping at the control level, where we had shown 4 different examples (bootstrapping headlines 1a, 2a-c, see above). Here mainly visual information is used. This is done by linking shape similarities to action affordances into categories. These categories create the links in the repository of objects&attributes with roles.

The learning of perception and action categories requires quite some time during human development because large scale statistics on perceptual data need to be acquired and evaluated to sufficiently ground the categories. This learning process is working along two tracks. On a behavioral track, a rather small set of archetypical behaviors (as outlined in [32]) ensures the early association of objects with actions. The general execution of an action generates the required low-level sensorimotor experience later to be used for structural bootstrapping and facilitates a model building by creating internal world knowledge. This – in turn – can be used by older children and adults to perform mental simulations of potential action contingencies thereby creating the second track.

The fundamental problem of these processes is the dimensionality of the potential sensorimotor contingencies (e.g. think of the visual input space, [31] leading to a level of complexity that generates a very difficult learning/simulation task. To handle this complexity, an appropriate representation of sensorimotor information is required. Analysis of the visual representation in the vertebrate brain suggest that this takes place in form of a deep hierarchy which potentially allows for providing search spaces with different degree of granularity, different order of feature combinations and different levels of semantic abstraction [40]. This may lead to the required complexity reduction and could lead to the emergence of new structures in the internal world model of the agent further speeding up structural bootstrapping.

## 2.8 Conclusions

The main contribution of this study was to introduce a novel concept for the fast acquisition of actions-relevant information (by a robot), called Structural Bootstrapping. By ways of one complex, multi-layered

experiment, we have tried to demonstrate that this concept can be successfully used across all levels of our cognitive architecture (planning-, mid-, and sensorimotor level). The concept reaches into aspects of cognitive action understanding and learning by humans and its rigorous application will require more experiments in the future.

## Chapter 3

# Structural Bootstrapping Examples at Different Levels

### 3.1 Semantic Event Chain-based Assimilation and Accommodation of actions

We have introduced a Piaget-like approach including assimilation and accommodation processes [49] for developing a robot action library. We base the approach on the structure as provided by Semantic Event Chains (SECs), however for our approach we needed an extension of the SEC approach. As a SEC we consider a sequence of touching and un-touching events between objects. What we call extended Semantic Event Chain, is this sequence of the touching and un-touching events with poses, trajectories and forces, specific to each individual transition, as well as the object names attached to each line in the SEC (i.e. we talk not about object1 and object2 participating in the relation, but we talk, e.g., about hand and knife participating in the relation). The ideas of assimilation is as follows: if the SECs of two actions are similar, we can perform action assimilation, if not accommodation. However, at the assimilation step we consider extended SEC entities, and we consider which of those entities match and which entities shall be indicated as different. E.g., for assimilating a chop action into the cut-category, we indicate that we keep the pose relation between knife and objects to be cut (e.g. vegetables) of the initially known action cut, but we add a trajectory that is typical for chop into the description. Now we know that we can divide vegetables in pieces by "cut-like" actions, where pose of the knife shall be perpendicular to the vegetable and where trajectories can be of two types: either the cyclic cutting-like trajectory or the abrupt knife-down chop-like trajectory. For details see [ATV<sup>+</sup>13].

### 3.2 Learning Object-Action Relations

In this section we provide examples on bootstrapping processes for learning object-action relations.

#### 3.2.1 Active Learning with Knowledge Propagation

In the previous Deliverable D3.1.1 we introduced our general framework for learning object-action relations via knowledge propagation [60]. Now we present extensions that address the problems arising from the complexity in the learning interactions, i.e., connections between actions and objects, since testing great numbers of such interactions on a real robot is generally infeasible. To overcome these complexity issues, we propose an algebraic, statistical active learning strategy [SP14]. It starts by building an acceptably small but expressive starting training set covering a specially-constructed subset of all possible interactions that can be tested in a real robot environment. This set is expressive in that it allows us to predict the outcome of all untested cases and to provide confidence estimates of the predictions. These confidence estimates are subsequently used to actively select and incorporate untested cases into the training set. The new items improve the prediction capabilities by increasing the accuracy

and extending the coverage of the domain containing the possible interactions. The combination of the specially-designed initial training set and its extension by active learning provides a feasible, but at the same time effective, structural bootstrapping framework. We currently employ this learning approach to create an object-action interaction data base for bootstrapped affordance learning [Ugu14].

### 3.2.2 Bootstrapped Affordance Learning

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.

We designed and implemented a proof-of-concept system for an affordance-learning task [Ugu14]. At the first level, the robot learns to predict the effects of certain actions on single objects from a given set based on category information and shape features. At the second level, the robot learns to predict the effects of actions on pairs of objects, in two different ways: Without bootstrapping, learning is based on the same, low-level features as the first-level learning; with bootstrapping, it uses the affordances learned by the first-level learner as additional features. We show that bootstrapped learning is more effective than learning without bootstrapping.

### 3.2.3 Learning Object-Action Relations with Homogeneity Analysis

Our knowledge propagation framework [60] is potentially very powerful due to its exploitation of kernelized maximum-margin methods, but these very methods can make its properties nonobvious.

We designed a complementary method for learning object-action relations that addresses an overlapping space of problems. It is based on straightforward Euclidean reasoning, which brings with it the advantage of providing useful visualizations of the intricate dependency structures our learning problems exhibit, but which also limits its expressive power and flexibility compared to knowledge propagation.

Our method [XSP13] is based on homogeneity analysis, a tool for categorical multivariate statistical analysis. In this paper, we show for an example scenario how to learn object-action relations with homogeneity analysis, and devise methods for reasoning tasks such as predicting the effect of actions on novel objects, and selecting objects such that a given action has a desired effect.

### 3.2.4 Generative learning of correlations between object properties and action parameters

In [DSEA14], we address the question of generative knowledge construction from sensorimotor experience, which is acquired by observation and exploration. We show how to build generative models of actions and their effects on objects, together with perceptual representations of the objects, which then can be used in internal simulation to predict the outcome of actions. Specifically, we address the the experiential learning of association between object properties (softness and height) and action parameters for the wiping task where a generative model is acquired from sensorimotor experience resulting from wiping experiments. Object and action are linked to the observed effect to generate training data for learning a nonparametric continuous model using Support Vector Regression. In subsequent iterations, this model is grounded and used to make predictions on the expected effects for novel objects which can be used to constrain the parameter exploration. The complete learning cycle and together with the required skills have been implemented on the humanoid robot ARMAR-IIIb. Experiments with set of wiping objects differing in softness and height (size) demonstrate efficient learning and adaptation behaviour of the action of wiping to unknown objects.

The cycle incorporates two learning stages. In the first stage, the learning process is triggered by the observation of a human wiping action. The captured demonstration is encoded as a periodic Dynamic Movement Primitive (see [20] and Deliverable 2.2.1) yielding a movement policy which is mainly controlled by the amplitude and the frequency of the desired movement. Adapted to the scene the encoded action is executed whereas the observation of its effect provide information based on which the action representation is grounded and refined. In the second stage, models are generated which encode the relationship between

object properties, action parameters and effects. For this purpose, the robot explores a given object haptically in order to determine its properties such as softness and height ([12, 55]). Through execution of the wiping action with this specific object and observation of the resulting effect, the optimal action parameter, in our case the amplitude, is specified. Based on this experiential data a Support Vector Regression is applied to train models which represent the object-action-effect relations. In subsequent iterations of this cycle, these models are used for the prediction of action parameters and to accelerate the learning process for novel objects where the robot becomes able to select suitable action parameters in case of unknown/unexplored objects.

Future work will deal with the question of how to make better predictions by an enriched object representation which contains information about the object's weight and the geometry. Furthermore, we will extend this work towards more versatile wiping behaviour by considering additional action parameters such as different hand orientations during wiping and the integration of interaction forces in the learning cycle. For more details see [DSEA14].

### 3.2.5 Object Categorization with Knowledge Propagation

In D3.1.1 we reported on an application of our knowledge propagation framework [60] to hierarchical classification [59], both written up as technical reports. In the meantime we developed this work further and applied it to classification tasks using highly-competitive 3D features we developed within the EU FP7 ICT project IntellAct, which is currently under review for a major journal [51].

## Chapter 4

# Conclusion

By the third year of the project the Xperience consortium has started connecting the individual contributions of structural bootstrapping into a unified framework, as presented in Chapter 2. However, fast progress on the level of individual components of structural bootstrapping is happening in parallel and this progress was summarized in Chapter 3. In the remaining years of the Xperience project the newly developed as well as improved components will be integrated into the now started framework. Also more benchmarking results of how structural bootstrapping allows to make robot action acquisition faster will be obtained.



# References

- [1] E. E. Aksoy, A. Abramov, J. Dörr, N. KeJun, B. Dellen, and F. Wörgötter. Learning the semantics of object-action relations by observation. *Int. J. Robot. Res., Special Issue on Semantic Perception for Robots in Indoor Environments*, 2011, in press.
- [2] E. E. Aksoy, A. Abramov, F. Wörgötter, and B. Dellen. Categorizing object-action relations from semantic scene graphs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 398–405, may 2010.
- [3] E. E. Aksoy, M. Tamosiunaite, R. Vuga, A. Ude, C. Geib, M. Steedman, and F. Wörgötter. Structural bootstrapping at the sensorimotor level for the fast acquisition of action knowledge for cognitive robots. In *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, Osaka, Japan, August 2013.
- [4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.
- [5] T. Asfour, P. Azad, N. Vahrenkamp, K. Regenstein, A. Bierbaum, K. Welke, J. Schröder, and R. Dillmann. Toward Humanoid Manipulation in Human-Centred Environments. *Robotics and Autonomous Systems*, 56:54–65, January 2008.
- [6] T. Asfour, K. Regenstein, P. Azad, J. Schröder, N. Vahrenkamp, and R. Dillmann. ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control. In *Humanoids*, pages 169–175, Genova, Italy, December 2006.
- [7] Tamim Asfour, Nikolaus Vahrenkamp, David Schiebener, Martin Do, Markus Przybylski, Kai Welke, Julian Schill, and Rüdiger Dillmann. ARMAR-III: Advances in Humanoid Grasping and Manipulation. *Journal of the Robotics Society of Japan*, 31(4):341–346, 2013.
- [8] P. Azad, T. Asfour, and R. Dillmann. Accurate shape-based 6-dof pose estimation of single-colored objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2690–2695, 2009.
- [9] P. Azad, T. Asfour, and R. Dillmann. Combining harris interest points and the sift descriptor for fast scale-invariant object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4275–4280, 2009.
- [10] P. Azad, D. Münch, T. Asfour, and R. Dillmann. 6-dof model-based tracking of arbitrarily shaped 3d objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 0–0, 2011.
- [11] N.I. Badler. *Temporal Scene Analysis: Conceptual Descriptions of Object Movements*. PhD thesis, University of Toronto, Canada, 1975.
- [12] Alexander Bierbaum, Matthias Rambow, Tamim Asfour, and Rüdiger Dillmann. Grasp Affordances from Multi-Fingered Tactile Exploration using Dynamic Potential Fields. pages 168 – 174, Paris, France, 2009.
- [13] A. Billard, S. Calinon, and F. Guenter. Discriminative and adaptive imitation in uni-manual and bi-manual tasks. *Robot. Auton. Syst.*, 54:370–384, 2006.
- [14] Gennaro Chierchia. Syntactic bootstrapping and the acquisition of noun meanings: the mass-count issue. In Barbara Lust and John Whitman Margarita Suner, editors, *Heads, Projections and Learnability Volume 1*, pages 301–318. Hillsdale, New jersey, 1994.

- [15] R. Cubek and W. Ertel. Learning and Execution of High-Level Concepts with Conceptual Spaces and PDDL. In *3rd Workshop on Learning and Planning, ICAPS (21st International Conference on Automated Planning and Scheduling)*, 2011.
- [16] L. Cuiyan, Z. Dongchun, and Z. Xianyi. A survey of repetitive control. In *IEEE/RSJ International Conference on Robots and Systems (IROS)*, pages 1160–1166, Sendai, Japan, 2004.
- [17] Rüdiger Dillmann, Tamim Asfour, Martin Do, Rainer Jäkel, Alexander Kasper, Pedram Azad, Aleš Ude, Sven R. Schmidt-Rohr, and Martin Lösch. Advances in robot programming by demonstration. *KI - Künstliche Intelligenz*, 24(4):295–303, 2010.
- [18] M. Do, J. Schill, J. Ernesti, and T. Asfour. Learning how to wipe: A case study of structural bootstrapping from sensorimotor experience. In *IEEE International Conference on Robotics and Automation (ICRA)*, accepted, 2014.
- [19] Staffan Ekvall and Danica Kragic. Robot learning from demonstration: a task-level planning approach. *International Journal of Advanced Robotic Systems*, 5(3):223–234, 2008.
- [20] Johannes Ernesti, Ludovic Righetti, Martin Do, Tamim Asfour, and Stefan Schaal. Encoding of periodic and their transient motions by a single dynamic movement primitive. In *IEEE-RAS International Conference on Humanoid Robots*, pages 57–64, Osaka, Japan, 2012.
- [21] Kutluhan Erol, James A. Hendler, and Dana S. Nau. HTN planning: Complexity and expressivity. In *AAAI*, pages 1123–1128, 1994.
- [22] C. Fisher and L.R. Gleitman. Language acquisition. In *Pashler HF and Gallistel CR (eds.), Steven's Handbook of Experimental Psychology, Vol 3: Learning and Motivation*, pages 445–496. New York: John Wiley & Sons, 2002.
- [23] Cynthia Fisher. Structural limits on verb mapping: the role of analogy in childrens interpretation of sentences. *Cogn Psychol*, 31:41–81, 1996.
- [24] Cynthia Fisher, Yael Gertner, Rose M. Scott, and Sylvia Yuan. Syntactic bootstrapping. *WIREs Cognitive Science*, 1:143–149, 2010.
- [25] Andrej Gams, Martin Do, Aleš Ude, Tamim Asfour, and Rüdiger Dillmann. On-line periodic movement and force-profile learning for adaptation to new surfaces. In *2010 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 560–565, Nashville, TN, 2010.
- [26] Andrej Gams, Auke Ijspeert, Stefan Schaal, and Jadran Lenarčič. On-line learning and modulation of periodic movements with nonlinear dynamical systems. *Autonomous Robots*, 27(1):3–23, 2009.
- [27] Andrej Gams, Jesse van den Kieboom, Massimo Vespignani, Luc Guyot, Aleš Ude, and Auke Ijspeert. Rich periodic motor skills on humanoid robots: Riding the pedal racer. In *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.
- [28] Mustansar Ghazanfar, Adam Prügel-Bennett, and Sandor Szedmak. Kernel-Mapping Recommender System Algorithms. *Information Sciences*, 208:81–104, 11 2012.
- [29] L.R. Gleitman. The structural sources of verb meanings. *Language Acquisition*, 1:3–55, 1990.
- [30] L.R. Gleitman. Hard words. *Language Learning and Language Development*, 1:23–64, 2005.
- [31] G.H. Granlund. The complexity of vision. *Signal Processing*, 74, 1999.
- [32] F. Guerin, N. Krüger, and D. Kraft. A survey of the ontogeny of tool use: from sensorimotor experience to planning. *IEEE TAMD*, 5:18 – 45.
- [33] Kathy Hirsh-Pasek and Roberta Michnick Golinkoff, editors. *Action Meets World: Now Children Learn Verbs*. Oxford University Press, 2006.
- [34] Julia Hockenmaier and Mark Steedman. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396, 2007.
- [35] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computations*, 25(2):328–373, 2013.

- [36] Hedvig Kjellström, Javier Romero, and Danica Kragić. Visual object-action recognition: Inferring object affordances from human demonstration. *Comput. Vis. Image Underst.*, 115(1):81–90, jan 2011.
- [37] J. Kober, A. Wilhelm, E. Oztop, and J. Peters. Reinforcement learning to adjust parametrized motor primitives to new situations. *Auton. Robots*, 33(4):361–379, 2012.
- [38] K. Kronander, M.S.M. Khansari-Zadeh, and A. Billard. Learning to control planar hitting motions in a minigolf-like task. In *Proc. 2011 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 710–717, 2011.
- [39] Norbert Krüger, Christopher Geib, Justus Piater, Ronald Petrick, Mark Steedman, Florentin Wörgötter, Aleš Ude, Tamim Asfour, Dirk Kraft, Damir Omrčen, Alejandro Agostini, and Rüdiger Dillmann. Object-action complexes: Grounded abstractions of sensorimotor processes. *Robotics and Autonomous Systems*, 59:740–757, 2011.
- [40] Norbert Krüger, Peter Janssen, Sinan Kalkan, Markus Lappe, Aleš Leonardis, Justus Piater, Antonio J. Rodríguez-Sánchez, and Laurenz Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE PAMI*, 35(8):1847–1871, 2013.
- [41] Tom Kwiatkowski, Sharon Goldwater, Luke S. Zettlemoyer, and Mark Steedman. A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. In *EACL*, pages 234–244, 2012.
- [42] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and José Santos-Victor. Learning Object Affordances: From Sensory Motor Maps to Imitation. *IEEE Transactions on Robotics*, 24(1):15–26, 2 2008.
- [43] Wail Mustafa, Nicolas Pugeault, and Norbert Krüger. Multi-view object recognition using view-point invariant shape relations and appearance information. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.
- [44] L.R. Naigles. The use of multiple frames in verb learning via syntactic bootstrapping. *Cognition*, 58:221–251, 1996.
- [45] S. Navarro, N.Gorges, H. Wörn, J. Schill, T. Asfour, and R. Dillmann. Haptic object recognition for multi-fingered robot hands. In *IEEE Haptics Symposium*, pages 497–502, 2012.
- [46] B. Nemeč, R. Vuga, and A. Ude. Exploiting previous experience to constrain robot sensorimotor learning. In *Proc. 11th IEEE-RAS Int. Conf. Humanoid Robots*, pages 727–732, 2011.
- [47] Michael Pardowitz, Steffen Knoop, Rüdiger Dillmann, and Raoul D. Zöllner. Incremental Learning of Tasks From User Demonstrations, Past Experiences, and Vocal Comments. *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics*, 37(2):322–332, 2007.
- [48] R. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *International Conference on Artificial Intelligence Planning and Scheduling (AIPS)*, pages 212–221, 2002.
- [49] Jean Piaget. *The Origins of Intelligence in the Child*. Routledge, London, New York, 1953.
- [50] Steven Pinker. *Language Learnability and Language Development*. Cambridge University Press, Cambridge, 1984.
- [51] Antonio J Rodríguez-Sánchez, Sandor Szedmak, and Justus Piater. An object-curvature and viewpoint-centered 3D descriptor sets new standards in classification tasks. *Pattern Analysis and Machine Intelligence*, 2014. Submitted.
- [52] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artif. Intell.*, 5(2):115–135, 1974.
- [53] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3:233–242, 1999.
- [54] Stefan Schaal, Peyman Mohajjerian, and Auke Ijspeert. Dynamics systems vs. optimal control – a unifying view. *Progress in Brain Research*, 165(6):425–445, 2007.

- [55] J. Schill, J. Laaksonen, M. Przybylski, V. Kyrki, T. Asfour, and R. Dillmann. Learning continuous grasp stability for a humanoid robot hand based on tactile sensing. In *IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechanics (BioRob)*, pages 1901–1906, Rome, Italy, June 2012.
- [56] Jesse Snedeker. Cross-Situational Observation and the Semantic Bootstrapping Hypothesis. In *E. Clark (ed.), Proc. 13th Ann. Child Language Research Forum. Stanford, CA: Center for the Study of Language and Information*, pages 445–496. New York: John Wiley & Sons, 2002.
- [57] M. Sridhar, G. A. Cohn, and D. Hogg. Learning functional object-categories from a relational spatio-temporal representation. In *Proc. 18th European Conference on Artificial Intelligence*, pages 606–610, 2008.
- [58] Mark Steedman. *The Syntactic Process*. MIT Press, 2000.
- [59] Sandor Szedmak. Shape-based object categorization. Technical report, University of Innsbruck, 2013.
- [60] Sandor Szedmak and Justus Piater. Learning object-action relations via knowledge propagation. Technical report, University of Innsbruck, 2012.
- [61] Austin Tate. Generating project networks. In *IJCAI*, pages 888–893, 1977.
- [62] Damien Teney and Justus Piater. Continuous Pose Estimation in 2D Images at Instance and Category Levels. In *Tenth Conference on Computer and Robot Vision*, pages 121–127. IEEE, 5 2013.
- [63] Emily Thomforde and Mark Steedman. Semi-supervised CCG lexicon extension. In *EMNLP*, pages 1246–1256, 2011.
- [64] M. Thomsen, L. Bodenhausen, and N. Krüger. Statistical identification of composed visual features indicating high-likelihood of grasp success. In *Workshop 'Bootstrapping Structural Knowledge from Sensory-motor Experience. IEEE International Conference on Robotics and Automation (ICRA)*, 2013.
- [65] F. Tracy. The language of childhood. *Am. J. Psychol.*, 6(1):107–138, 1893.
- [66] J. Trueswell and L. Gleitman. Learning to parse and its implications for language acquisition. In *Oxford Handbook of Psycholinguistics*, pages 635–656. Oxford, 2007.
- [67] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Trans. Robot.*, 26(5):800–815, 2010.
- [68] Mirko Wächter, Sebastian Schulz, Tamim Asfour, Eren Aksoy, Florentin Wörgötter, and Rüdiger Dillmann. Action sequence reproduction based on automatic segmentation and object-action complexes. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 189–195, Atlanta, Georgia, 2013.
- [69] F. Wörgötter, A. Agostini, N. Krüger, N. Shylo, and B. Porr. Cognitive agents – A procedural perspective relying on predictability of object-action complexes (OACs). *Robotics and Autonomous Systems*, 57(4):420–432, 2009.
- [70] F. Wörgötter, E. E. Aksoy, N. Krüger, J. Piater, A. Ude, and M. Tamosiunaite. A simple ontology of manipulations: Towards representations for manipulation actions in robotics. *IEEE Transactions on Autonomous Mental Development (TAMD)*, 5(2):117–134, 2013.
- [71] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Homogeneity Analysis for Object-Action Relation Reasoning in Kitchen Scenarios. In *2nd Workshop on Machine Learning for Interactive Systems*, pages 37–44. ACM, 8 2013. Workshop at IJCAI.
- [72] Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos. Detection of manipulation action consequences (mac). In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2563–2570, 2013.

# Attached Articles

- [ATV<sup>+</sup>13] E. E. Aksoy, M. Tamosiunaite, R. Vuga, A. Ude, C. Geib, M. Steedman, and F. Wörgötter. Structural bootstrapping at the sensorimotor level for the fast acquisition of action knowledge for cognitive robots. In *IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, Osaka, Japan, August 2013.
- [DSEA14] M. Do, J. Schill, J. Ernesti, and T. Asfour. Learning how to wipe: A case study of structural bootstrapping from sensorimotor experience. In *IEEE International Conference on Robotics and Automation (ICRA)*, accepted, 2014.
- [SP14] Sandor Szedmak and Justus Piater. An active learning based sampling design for structural bootstrapping. Technical report, University of Innsbruck, 2014.
- [Ugu14] Emre Ugur. Bootstrapping multi-object affordance learning using learned single-affordance features. Technical report, University of Innsbruck, 2014.
- [WGT<sup>+</sup>14] Florentin Wörgötter, Chris Geib, Minija Tamosiunaite, Eren Erdal Aksoy, Justus Piater, Hanchen Xiong, Ales Ude, Bojan Nemeč, Dirk Kraft, Norbert Krüger, Mirko Wächter, and Tamim Asfour. Structural bootstrapping – a novel concept for the fast acquisition of action-knowledge. *Transaction of Autonomous Mental Development*, 2014. Submitted.
- [XSP13] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Homogeneity Analysis for Object-Action Relation Reasoning in Kitchen Scenarios. In *2nd Workshop on Machine Learning for Interactive Systems*, pages 37–44. ACM, 8 2013. Workshop at IJCAI.

# Structural bootstrapping at the sensorimotor level for the fast acquisition of action knowledge for cognitive robots

E. E. Aksoy<sup>1</sup>, M. Tamosiunaite<sup>1</sup>, R. Vuga<sup>2</sup>, A. Ude<sup>2</sup>, C. Geib<sup>3</sup>, M. Steedman<sup>3</sup>, and F. Wörgötter<sup>1</sup>

**Abstract**—Autonomous robots are faced with the problem of encoding complex actions (e.g. complete manipulations) in a generic and generalizable way. Recently we had introduced the Semantic Event Chains (SECs) as a new representation which can be directly computed from a stream of 3D images and is based on changes in the relationships between objects involved in a manipulation. Here we show that the SEC framework can be extended (called “extended SEC”) with action-related information and used to achieve and encode two important cognitive properties relevant for advanced autonomous robots: The extended SEC enables us to determine whether an action representation (1) needs to be newly created and stored in its entirety in the robot’s memory or (2) whether one of the already known and memorized action representations just needs to be refined. In human cognition these two processes (1 and 2) are known as accommodation and assimilation. Thus, here we show that the extended SEC representation can be used to realize these processes originally defined by Piaget for the first time in a robotic application. This is of fundamental importance for any cognitive agent as it allows categorizing observed actions in *new* versus *known* ones, storing only the relevant aspects.

## I. INTRODUCTION

A central issue for the development of autonomous robots is how to quickly acquire new concepts for planning and acting, for example learning a relatively complex manipulation sequence like cutting a cucumber. Association-based or reinforcement learning methods are usually too slow to achieve this in an efficient way. They are therefore most often used in combination with supervised learning methods. Especially the Learning from Demonstration (LfD) paradigm seems promising for cognitive learning ([1], [2], [3], [4], [5]) because we (humans) employ it very successfully. The problem that remains in all these approaches is how to represent complex actions or chains of actions in a generic and generalizable way allowing to infer the “meaning” (semantics) of an action irrespective of its individual instantiation.

In our earlier studies we introduced the “Semantic Event Chain” (SEC) as a possible descriptor for manipulation actions [6], [7], [8]. The SEC framework analyzes the sequence of changes of the *relations* between the objects that are being

The research leading to these results has received funding from the European Communitys Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

<sup>1</sup>Inst. Physics-3 & BCCN, University of Göttingen, Friedrich-Hund Platz 1, D-37077, Germany [eaksoy,minijs,worgott] at physik3.gwdg.de

<sup>2</sup>Jožef Stefan Institute, Department of Automatics, Biocybernetics and Robotics, Jamova 39, Ljubljana, Slovenia [rok.vuga,ales.ude] at ijs.si

<sup>3</sup>School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh, Scotland [cgeib,steedman] at inf.ed.ac.uk

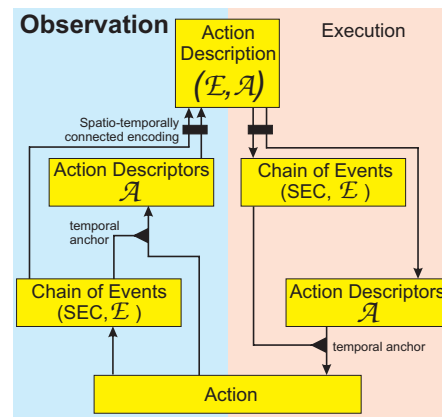


Fig. 1: Schematic representation of the acquisition of action information by observation using SECs and Action descriptors (left) as well as execution (right).

manipulated by a human or a robot. Consequently, SECs are invariant to the particular objects used, the precise object poses observed, the actual trajectories followed, or the resulting interaction forces between objects. All these aspects are allowed to change and still the same SEC is observed and captures the “essence of the action” as demonstrated in several action classification tests performed by us [6], [7], [8].

The goal of this paper is to extend the SECs with action related information (action descriptors) and to use the obtained structure for the assimilation of novel information into the existing schemata or for the creation of novel schemata (accommodation) in a Piagetian sense [9]. The first happens when an agent finds that a newly observed action is compatible with an already memorized SEC, but there are some elements present in the new action which are truly novel. These can then be stored (assimilated) together with the known ones into the existing schema. The second happens when the agent realizes that the new action does not compare to any of its known schemata and requires a novel schema to be created (accommodation). This way agent’s cumulative memory of actions can be developed. The main contribution of this paper is therefore the enrichment process of event chains to further use memory in a more efficient way. With respect to our previous approaches ([6], [7], [8]), enriched SECs also lead to extraction and comparison of action descriptors such as trajectory segments, pose, and object information.

The whole process of action representation using SECs and action descriptors is summarized in Fig. 1. In this paper we are concerned only with the observation phase (left side). The



Fig. 2: Real action scenarios. (a), (c), (e) Sample original key frames, (b), (d), (f) corresponding segments and graphs for the following actions: *Cutting*, *Chopping*, and *Stirring*.

execution stage (right side) is described in [10] which shows the possibility of imitating actions with robots by directly using the here introduced representations.

The paper is organized as follows. We start with the description of the extended SEC representation and then provide an example for assimilation as well as accommodation using this framework. We call the developed algorithm Structural Bootstrapping. In the discussion section we embed these results into the state of the art and provide also a comparison to child language development, from where the concept of Bootstrapping originates.

## II. DATA AND DATA REPRESENTATIONS

### A. Data and Pre-processing

Data structures and algorithms we developed are generic and do not depend on the actual input data. Nonetheless, it is best to first describe some example experiments, which should make it easier to understand all components of our representation. We have investigated three different manipulation actions: *Cutting*, *Chopping*, and *Stirring*. In the *Cutting* action,

a hand is cutting a cucumber by moving a knife back and forth. In the *Chopping* action, a cleaver follows a straight trajectory to cut a carrot. The *Stirring* action represents a scenario in which a spoon is used to stir milk in a bucket. We recorded these three manipulation sequences with Microsoft Kinect. The developed system first pre-processes all movie frames by a real-time image segmentation procedure ([11], [12]) to uniquely identify and track objects (including hands) in the observed actions. Each segmented image is represented by a graph: nodes represent segment centers and edges indicate whether two objects touch each other or not (in 3D). Fig. 2 (a-f) depict sample original images with extracted segments (regions) and graphs for each scenario.

While recording each action sequence, the trajectories of the hand and manipulated objects as well as their poses are measured by the 3D motion capture system Optotrak. Fig. 5 illustrates the measured trajectories for the knife and cleaver as used in the *Cutting* and *Chopping* scenario. These trajectories were measured by attaching a set of 3 active markers to each of the objects involved in the action.



1) *Semantic Event Chain*  $\mathcal{E}$ : By using an exact graph matching technique the framework discretizes the entire graph sequence into decisive main graphs. A new main graph is identified whenever a new node or edge is formed or an existing edge or node is deleted. Thus, each main graph represents a “key frame” in the manipulation sequence. All extracted main graphs form the core skeleton of the SEC  $\mathcal{E}$ , which is a matrix where rows (index  $i$ ) are possible pairwise object relations (e. g. between the hand and knife or the knife and cucumber) and columns (index  $j$ ) describe the scene configuration at time  $j$  when a new main graph has occurred. Fig. 3 (a) indicates the SEC with sample main graphs derived from the *Cutting* action shown in Fig. 2 (a).

Let  $\mathcal{E}$  be a semantic event chain with size  $n \times m$ . Then it can be written as:

$$\mathcal{E} = \begin{bmatrix} R(o_{a_1}, o_{b_1}) \\ R(o_{a_2}, o_{b_2}) \\ \vdots \\ R(o_{a_n}, o_{b_n}) \end{bmatrix} = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,m} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{n,1} & r_{n,2} & \cdots & r_{n,m} \end{bmatrix}, \quad (1)$$

where  $r_{i,j}$  represents a spatial relation  $R$  between an object pair  $o_{a_i}, o_{b_i}$  at time  $j$ . Thus all pairs of objects need to be considered once, where rows that do not contain any changes in object-object relations are deleted. The maximum total number of rows  $n$  is defined as  $n = \lambda(\lambda - 1)/2$ , where  $\lambda$  is the total number of segments. However, the total number of columns  $m$  depends on the action and can vary.

Relations are given by:

$$r_{i,j} \in \{\text{not touching (N), touching (T), absence (A)}\}, \quad (2)$$

where N means that there is no edge between two segments, i.e. graph nodes corresponding to two spatially separated objects, T represents objects that touch each other, and absence of an object yields A.

2) *Action Encoder Matrix*  $\mathcal{A}$ : A central advantage of our framework is that we can extract temporal anchor points from a SEC. These points tell us when to “pay attention to the action”, because action-relevant details occur at or near the *transitions* between the relations recorded in the SEC. These transitions are encoded by  $\mathcal{T}_{i,j}$ , defined as:

$$\mathcal{T}_{i,j} = \begin{cases} 0 & \text{if } r_{i,j} = r_{i,j-1}, j > 1 \\ [\mathcal{T}.\{d^1, d^2, \dots, d^k\}]_{i,j} & \text{else} \end{cases} \quad (3)$$

The variables  $\mathcal{T}_{i,j}$  correspond to the respective transition and its  $k$  action descriptors  $d$  (described later). One could think of each  $\mathcal{T}$  as a derivative-like “change-encoder”, which is non-zero whenever there is a change in the scene graph (“something has happened with any of the objects”). For improved readability transitions are given in plain-text (e.g. NT, AT, AN, etc.) using the corresponding pairs of relations  $r$  from the event chain to encode this. For example, entries  $(r_{4,4}, r_{4,5})$  and  $(r_{4,7}, r_{4,8})$  represent transitions from N to T and from T to N as depicted in shaded boxes in Fig. 3 (a). Hence in these cases we would write specifically  $\mathcal{T}_{4,5} = [\text{NT}]_{4,5}$  and  $\mathcal{T}_{4,8} = [\text{TN}]_{4,8}$ . For these (and all others where  $r_{i,j} \neq r_{i,j-1}$ ) we have  $\mathcal{T}_{i,j} \neq 0$ . Descriptors  $d$  need to be computed next.

Note that for the first column ( $j = 1$ ) there are no transitions and we write  $\mathcal{T}_{i,1} = X_i, \forall i$ , where descriptors  $d_{i,1}$  define the initial state of the corresponding objects before the action progresses. Fig. 3 (b) depicts the corresponding transitions derived from the SEC given in Fig. 3 (a). In the following we will abbreviate  $\mathcal{T}.\{d^1, d^2, \dots, d^k\}$  with  $\mathcal{T}.d$  where possible.

The resulting structure will, thus, be a matrix describing the action  $\mathcal{A}$ :

$$\mathcal{A} = \begin{bmatrix} X_1 & [\mathcal{T}.d]_{1,2} & \cdots & [\mathcal{T}.d]_{1,m} \\ X_2 & [\mathcal{T}.d]_{2,2} & \cdots & [\mathcal{T}.d]_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ X_n & [\mathcal{T}.d]_{n,2} & \cdots & [\mathcal{T}.d]_{n,m} \end{bmatrix}. \quad (4)$$

Each descriptor  $d$  contains information about the objects involved, their relative poses at time  $j$  of the event, as well as their trajectories until time  $j$  and the forces involved. In our implementation the variable  $k$  from Eq. (3) is set to 4, but this can be changed if more action-relevant attributes are needed. We define the descriptor set as:

- $d_{i,j}^1 = \{o_{a_i}, o_{b_i}\}_i$  is a set containing two object identifiers of those objects that are involved in the given event. Note that by definition of the event chain there are always exactly two objects for each event. Objects do not change along the rows of an event chain, thus index  $j$  is irrelevant.
- $d_{i,j}^2 = \{p\}_{i,j} = \{x, y, z, \alpha, \beta, \gamma\}_{i,j}$  is a set containing *relative* pose information *between* two object identifiers. The  $x, y, z$  and  $\alpha, \beta, \gamma$  values hold corresponding translation and rotation values, respectively.
- $d_{i,j}^3 = \{t\}_{i,j} = \{\mathbf{s}, \mathbf{g}, \tau, \mathbf{w}_{1,\dots,6}\}_{i,j}$ ,  $\mathbf{s}, \mathbf{g} \in \mathbb{R}^6$ , is

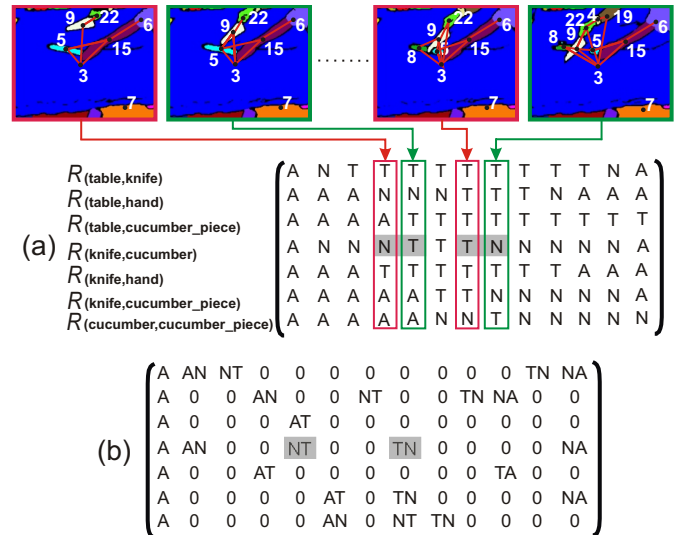


Fig. 3: The SEC ( $\mathcal{E}$ ) and transition matrix extracted from the *Cutting* action given in Fig. 2 (a). (a) SEC with corresponding sample main graphs and segments. (b) Transition matrix showing the respective relational transitions between object pairs. First column defines the initial relations. Shaded boxes show two sample transitions:  $\mathcal{T}_{4,5} = [\text{NT}]_{4,5}$  and  $\mathcal{T}_{4,8} = [\text{TN}]_{4,8}$ .



a set of parameters containing trajectory information in the Cartesian task space. In this study we use the modified Dynamic Movement Primitives (DMPs, [13]) to encode movement trajectories because they have faster convergence at the end points compared to the standard DMPs and allow smooth joining of movement sequences. Variables  $\mathbf{s}$  and  $\mathbf{g}$  denote start and goal (end) points of the DMP, respectively, and  $\tau$  is the time constant modulating the speed of movement. Vectors  $\mathbf{w}_l$ ,  $l = 1, \dots, 6$ , hold the shape parameters of the DMP given as weights for about 5-20 Gaussian kernels (see [13] for a description of the DMP parameterization). DMPs offer the advantage that they are robust to perturbations, can generalize to different start- and end-points, and also allow online modification of the movement by ways of sensory coupling ([14], [15], [13], [16]).

- $d_{i,j}^4 = \{\mathbf{f}\}_{i,j} = \{f_x, f_y, f_z, \tau_x, \tau_y, \tau_z\}_{i,j}$  is a 6D vector containing the Cartesian space force and torque information. Force information cannot directly be obtained from human demonstration and but requires own exploration (similar to the situation for a human child).

Thus, derived from the event chain  $\mathcal{E}$  and using additional information encoded with descriptors  $d$ , we have now obtained a new matrix  $\mathcal{A}$ . The event chain becomes obsolete by this. Still, it makes sense to keep both,  $\mathcal{E}$  and  $\mathcal{A}$ , to make the next step, the description of the bootstrapping algorithm, easier.

### B. Algorithm: Structural Bootstrapping

Sensorimotor structural bootstrapping consists of four main steps: (1) Initial memory formation, (2) observation, (3) comparison, and (4) generalization via Assimilation or Accommodation (Fig. 4). In the very first step, the framework analyzes and stores an action in the specific format described above. Let this first observed and stored action be  $\langle (\mathcal{E}, \mathcal{A})^{a1} \rangle$  where in the pseudo-code below we denote memory storage by brackets  $\langle \cdot \rangle$ . In the second step, a new action is observed  $(\mathcal{E}, \mathcal{A})^{a2}$ . In the third step, comparison, we determine whether both, stored and newly observed, actions are semantically the same (for example cutting and chopping have the same SEC, whereas stirring has a different one). Similarity between two event chains,  $\mathcal{E}^{a1}$  and  $\mathcal{E}^{a2}$  is measured using the definition of spatiotemporal similarity  $\zeta(\mathcal{E}^{a1}, \mathcal{E}^{a2})$  for SECs given in [6]. The last step, generalization, is divided into two aspects. If the semantic similarity  $\zeta(\mathcal{E}^{a1}, \mathcal{E}^{a2})$  is below a certain threshold  $\tau_{\mathcal{E}}$ , actions are not type-similar and we will store the complete newly observed action as a new schema (Accommodation, lower path in Fig. 4). Otherwise, i. e. for type-similar actions, we perform a comparison of the two descriptor sets  $[\mathcal{T}.d]_{i,j}^{a1}$  and  $[\mathcal{T}.d]_{i,j}^{a2}$ . Below we describe how DMP descriptors  $d_{i,j}^3$  can be compared. The comparison of other descriptors (object identity, relative pose information, and torques at the contact point) can be done using standard metrics.

The acquisition of new action information is completed by extending the memory either by storing the new action or by additionally storing those descriptors that are different compared to those already stored in the memory of a known action.

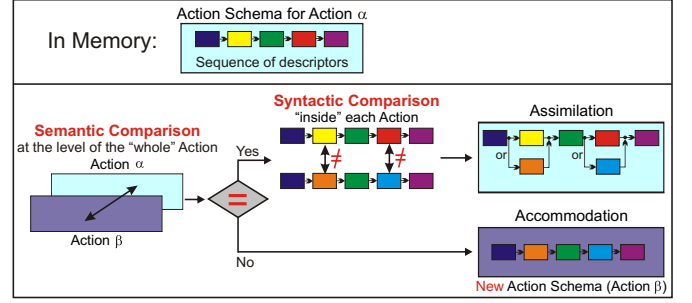


Fig. 4: Schematic representation of the required steps for Structural bootstrapping. For explanation see text.

In the latter case, individual entries in matrix  $\mathcal{A}$  turn into tuples of descriptor sets  $[\mathcal{T}.(d^{a1}, d^{a2})]_{i,j}$ . This whole procedure can be repeated as soon as more actions are observed. A concise description of the entire structural bootstrapping framework is given in Algorithm 1.

---

### Algorithm 1 Sensorimotor Structural Bootstrapping

---

```

Store first action in memory  $\langle \cdot \cdot \cdot \rangle$ .
 $\langle \cdot \rangle = \langle \cdot \rangle + (\mathcal{E}, \mathcal{A})^{a1}$  with  $\mathcal{A}^{a1}$  defined by  $[\mathcal{T}.d]_{i,j}^{a1}$ 
Observe next action.
 $(\mathcal{E}, \mathcal{A})^{a2}$  with  $\mathcal{A}^{a2}$  defined by  $[\mathcal{T}.d]_{i,j}^{a2}$ 
Semantic Comparison.
 $\zeta_{\mathcal{E}} = \zeta(\mathcal{E}^{a1}, \mathcal{E}^{a2})$ 
if  $\zeta_{\mathcal{E}} < \tau_{\mathcal{E}}$  (small similarity!) then
  New Action! Create new memory (Accommodate).
   $\langle (\mathcal{E}, \mathcal{A})^{a1} \rangle = \langle (\mathcal{E}, \mathcal{A})^{a1} \rangle + (\mathcal{E}, \mathcal{A})^{a2}$ 
else
  Type-similar Action! Perform syntactic comparison.
  for each event  $\mathcal{T}_{i,j} \neq 0$  do
    for all descriptors  $d$  indexed by  $l$  do
       $\zeta_d = \zeta([\mathcal{T}.d^l]_{i,j}^{a1}, [\mathcal{T}.d^l]_{i,j}^{a2})$ 
      Compare syntactic similarity.
      if  $\zeta_d < \tau_d$  (small similarity!) then
        New element! Assimilate into existing memory.
         $\langle (\mathcal{E}, \mathcal{A})^{a1} \rangle$  with  $[\mathcal{T}.d^l]_{i,j}^{a1} = [\mathcal{T}.(d^{l,a1}, d^{l,a2})]_{i,j}^{a1}$ 
      end if
    end for
  end for
end if

```

---

Algorithm 1 requires that we can compare actions both at the semantic event chain level [6] and at the action descriptor level. In the current implementation we use object identities, relative poses, and trajectories for comparisons at the action descriptor level. While comparing identities and relative poses is rather straightforward, the comparison of trajectories is more difficult and is described in more detail below.

1) *Comparison of movements encoded by DMPs:* DMPs provide a temporary and spatially invariant representation of a movement. Even if the timing  $\tau$  and the absolute position of the movement ( $\mathbf{s}$  and  $\mathbf{g}$ ) in space change, the parameters  $\mathbf{w}_l$

stay the same. Thus trajectories with similar velocity profiles will be fitted by similar shape parameters  $\mathbf{w} = [\mathbf{w}_1^T, \dots, \mathbf{w}_6^T]^T$  [14]. Similarities between two trajectories encoded by DMPs can be measured by computing the correlation between their parameter vectors. The correlation is given by the cosine of the angle between these two vectors:

$$\frac{\mathbf{w}_1^T \mathbf{w}_2}{\|\mathbf{w}_1\| \|\mathbf{w}_2\|}, \quad (5)$$

where  $\mathbf{w}_1$  and  $\mathbf{w}_2$  are the parameter vectors of two different movements. Thus, in the training phase we store a set of prototype movements for each action primitive. Classification is then performed by comparing the newly observed trajectory extracted from SECs to the available prototypes. More advanced methods like support vector machines could be used, but this was not necessary in our experiments.

The proposed classification method can be applied to compare DMPs describing the movements as long as the shape of underlying motion trajectories does not change with respect to the current configuration of the task. If this is not the case, then we can use statistical generalization with respect to the parameters of the task [17] to generate new movement prototypes to which the newly observed trajectories can be compared.

### III. RESULTS

We have applied the structural bootstrapping algorithm described above to the three example actions (*Cutting*, *Chopping*, *Stirring*). The framework first extracts key events and generates SECs for all action sequences as explained in section II-A. Then action-encoder matrices ( $\mathcal{A}$ ) are determined. Fig. 5 shows an incomplete, while graphical, rendering of the action-encoder matrices for the *Cutting* and *Chopping* actions.

We use *Cutting* as our reference action (Action *cut*, see also ‘‘In Memory’’, Fig. 4) and commit it to memory  $\langle \cdot \rangle = \langle \cdot \rangle + (\mathcal{E}, \mathcal{A})^{cut}$ . Then we define *Chopping* and *Stirring* as action indices *chop* and *stir*, respectively.

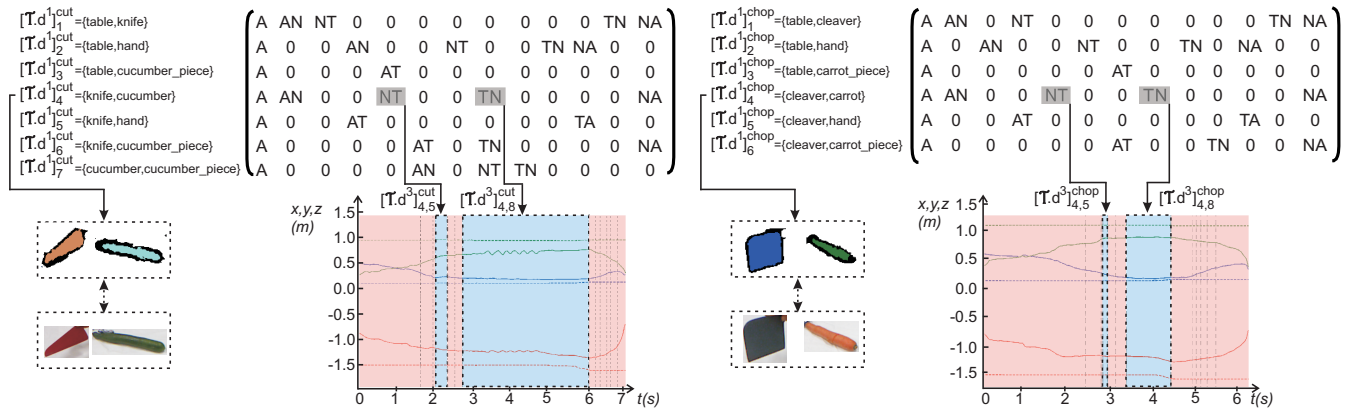


Fig. 5: Action-encoder matrices ( $\mathcal{A}$ ) with extracted descriptors for the *Cutting* and *Chopping* actions. Movement is described in table coordinate system,  $x$  and  $y$  - table plane coordinates (red and green),  $z$  - distance from the table (blue); solid lines stand for the tool demonstrator’s hand holding, dashed lines for the tip of the cucumber.

Structural bootstrapping continues with a semantic comparison of the event chains in the spatiotemporal domain. Fig. 6 (a) illustrates the similarity values for the different actions. Similarity measures are basically computed by comparing rows and columns of two event chains using simple sub-string search and counting algorithms. Relational changes are considered while comparing the rows, whereas for the columns the temporal order counts. We first search for the correspondences between rows of two event chains since rows can be shuffled. The searching process compares and counts equal entries of one row against the other using a standard sub-string search which does not rely on dimensions and allows to compare arbitrarily long manipulation actions. We then examine the order of columns to get the final similarity result. Details for similarity calculations are given in [6].

If one compares *Cutting* with itself, similarity is of course 100%, but we also observe high similarity values (88%) between *Cutting* and *Chopping*. On the other hand, the similarity between *Cutting* and *Stirring* is only (55%). In our earlier studies we had measured the discriminability of our

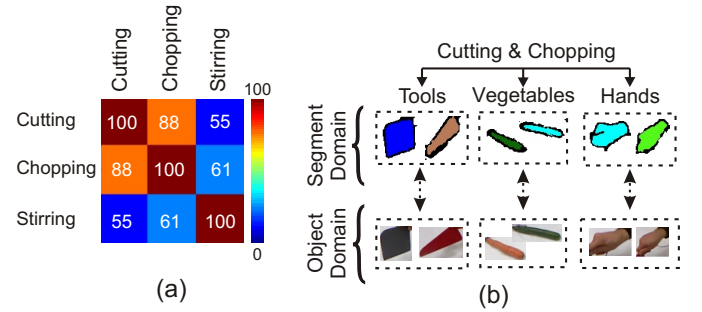


Fig. 6: Semantic comparison. (a) Similarity values between the *Cutting*, *Chopping*, and *Stirring* actions. (b) Segment categories showing which segments exhibit the same role in type-similar actions.

similarity measure using identical actions performed somewhat differently in a noisy environment [6]. From these studies we know that the discriminative threshold is usually about  $\tau_E = 65\%$ . When applying this threshold here we find that *Cutting* and *Chopping* are regarded as similar, whereas *Cutting* and *Stirring* are not. First we observe that this matches to our lay-man’s expectations. In general, we observe that this type of classification renders human-compatible semantics of “same/similar” versus “different” actions. [18] provides a huge confusion matrix showing the semantic similarities between different actions (e.g. push, hide, uncover, stir, cut, etc.) based on the manipulation action ontology. This huge confusion matrix shows the here presented semantic representation can distinguish actions to initiate the bootstrapping process.

The bootstrapping algorithm then proceeds differently for different actions. For *Stirring* we perform Accommodation and just commit the complete descriptor set to memory, i.e.  $\langle\langle \mathcal{E}, \mathcal{A} \rangle^{cut}\rangle = \langle\langle \mathcal{E}, \mathcal{A} \rangle^{cut}\rangle + \langle\langle \mathcal{E}, \mathcal{A} \rangle^{stir}\rangle$ .

For the type-similar actions *Cutting* and *Chopping* we perform a (syntactic) comparison at the level of the individual descriptors  $d$ . First we consider  $d^1$ , the objects. We find several objects (see Fig. 6 (b)). Note that noisy segment groups observed in some action versions are not categorized as objects since they can all be ignored after applying a SIFT-based object recognition algorithm [19] in the segment domain. Also different trajectories are observed. Fig. 5 shows sample object and trajectory descriptors computed for the *Cutting* and *Chopping* scenarios. For instance, as indicated by the shaded boxes, in both actions the same relational transitions (i.e. events) are observed from  $N$  to  $T$  and from  $T$  to  $N$  at index numbers  $i = 4, j = 5$  and  $i = 4, j = 8$ . Here we can perform Assimilation. Specifically  $[\mathcal{T}.d^1]_{4,j}^{cut} = \{knife, cucumber\}$  and  $[\mathcal{T}.d^1]_{4,j}^{chop} = \{cleaver, carrot\}$ . Assimilation renders:  $\langle\langle \mathcal{E}, \mathcal{A} \rangle^{cut}\rangle$  with  $[\mathcal{T}.d^1]_{4,j}^{cut} = [\mathcal{T}.(d^{1,cut}, d^{1,chop})]_{4,j}^{cut}$ . Here we note that the “concept of cutting”  $[\mathcal{T}.d^{cut}]^{cut}$  is extended by “aspects of chopping”  $[\mathcal{T}.(d^{cut}, d^{chop})]^{cut}$  (denoted by the general versus the specific indices in this notation).

Furthermore, Fig. 7 shows that relative poses of manipulated objects are rather weak cues to distinguish the cutting and chopping actions. The main reason is that the pose of a cutting tool is usually constrained to be perpendicular and in the middle of the object to be cut. In this sense, no assimilation is needed for pose. Note that each sample in Fig. 7 is recorded when the cutting tool starts to touch the object to be cut. However, the measured trajectory descriptors  $[\mathcal{T}.d^3]_{4,8}$ , as indicated with blue boxes in Fig. 5, are highly different in both actions as a consequence of the nature of cutting and chopping actions. Fig. 8 shows a correlation matrix between different instances of these two trajectory samples according to equation (5). On the other hand, descriptors  $[\mathcal{T}.d^3]_{4,5}$  in both actions are quite similar since in both versions the hand is approaching to vegetables in a similar way. Therefore, only  $[\mathcal{T}.d^3]_{4,8}^{chop}$  is added to the memory yielding:  $\langle\langle \mathcal{E}, \mathcal{A} \rangle^{cut}\rangle$  with  $[\mathcal{T}.d^3]_{4,8}^{cut} = [\mathcal{T}.(d^{3,cut}, d^{3,chop})]_{4,8}^{cut}$ .

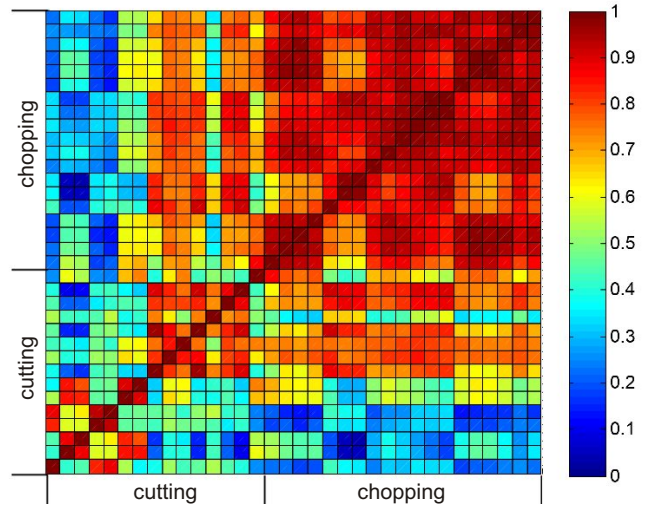


Fig. 7: Correlation between relative poses of the manipulated objects in 15 instances of cutting and 20 instances of chopping actions. Each instance is recorded when the cutting tool starts to touch the object to be cut. Red corresponds to the maximum correlation of 1.0 between the sample pair of poses and blue corresponds to the correlation of 0.0.

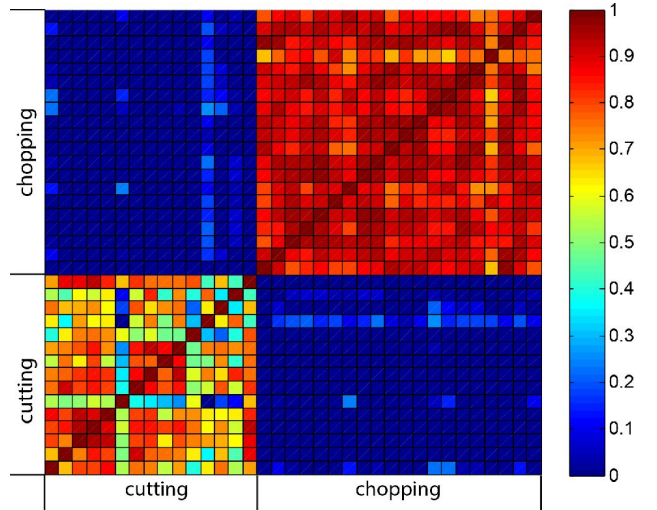


Fig. 8: Correlation between 15 instances of cutting and 20 instances of chopping trajectories according to equation (5). In red and blue are indicated the maximum (1.0) and minimum (0.0 or below) correlations between two sample trajectories.

#### IV. DISCUSSION AND CONCLUSION

In this paper we have presented two complementary approaches. (1) We have extended the Semantic Event Chain framework by action descriptors and (2) we have used the new framework to compare actions at different levels of semantic depth. This allowed us to subsume cutting and chopping into the same action category (still named “cut”) allowing to share most of the cutting- and chopping-action description within the same memory structure. On the other hand we were also

able to realize that cutting and stirring are more fundamentally different such that for both different memory representations have to be stored in their entirety. This distinction arises from a structural comparison either at the level of the SECs or “inside” the action descriptors  $d$ , which is called here Structural Bootstrapping. So far this study is based on only three actions. This is due to the fact that all these experiments take quite long and require storing various action-information combining different methods. Our current goal was to show the principles of accommodation and assimilation for which three actions suffice, but we are currently in the process of developing a more complete action-library based on the here presented encoding principles. Next we will now try to embed our study in the state of the art to show similarities and differences to other approaches.

#### A. State of the Art - Action Classification

Learning from Demonstration (LfD) has been successfully applied both at the control [1], [2] as well as the symbolic level [3], [4], [5]. Although various types of actions can be encoded at the control level, i. e. trajectory-level, this is not general enough to imitate complicated actions under different circumstances. On the other hand, at the symbolic level sequences of predefined abstract action units are used to learn complex actions, but this might lead to problems for execution as many parameters are left out in the resulting representation. Although our approach with SECs is a symbolic-level representation, SECs are enriched with additional decisive descriptors (e.g. trajectory, pose, etc.) and do not use any assumption or prior knowledge in the object or action domain. Ideas to utilize relations to reach semantics of actions can be found as early as in 1975. For instance, [20] introduced the first approach about the directed scene graphs in which each node identifies one object. Edges hold spatial information (e.g., LEFT-OF, IN-FRONT-OF, etc.) between objects. Based on object movement (trajectory) information events are defined to represent actions. The main drawback of this approach is that continuous perception of actions is ignored and is substituted instead by the idealized hand-made image sequences. This, however, had not been pursued in the field any longer as only now powerful enough image processing methods became available.

Thus, still there are only a few approaches attempting to reach the semantics of manipulation actions in conjunction with the manipulated objects [21], [22], [23]. The work presented in [21] represents an entire manipulation sequence by an activity graph which holds spatiotemporal object interactions. The difficulty is, however, that complex and large activity graphs need to be decomposed for further processing. In the work of [22], segmented hand poses and velocities are used to classify manipulations. A histogram of gradients approach with a support vector machine classifier is separately used to categorize manipulated objects. Factorial conditional random fields is then used to compute the correlation between objects and manipulations. [23] introduced visual semantic graph (inspired from our scene graphs) to recognize action

consequences based on changes in the topological structure of the manipulated object. Although all those works to a certain extent improve the classification of manipulations and/or objects, none of them extracts key events of individual manipulations.

[24] is one of the first approaches in robotics that uses the configuration transition between objects to generate a high-level description of an assembly task from observation. Configuration transitions occur when a face-contact relation between manipulated and stationary environmental objects changes. In our case, each relational transition is considered as a temporal anchor point, at which additional action descriptors are stored. All temporal anchor points are then used in the bootstrapping algorithm. In this sense, to our best knowledge, our work is the first attempts to evaluate semantics of manipulations in a Piagetian sense.

#### B. State of the Art - Link to Child Language Development

Apart from the aspect of action classification, there is another important link existing of our work to – in this case – an entirely different field, much unrelated to robotics. Structural Bootstrapping, as shown here, is strongly influenced from processes that dominate child language acquisition.

Children acquire the meaning of new words and constructions in their language using two related mechanisms. The primary process is *semantic bootstrapping* where the child associates “meaning-from-the-outside-world” with components of sentences. For example, if the word open is consistently uttered in situations where opening occurs (whatever else is going on), then the meaning of the word can be probabilistically inferred from the conceptual representation of the observed event ([25]). Once a certain amount of language has been acquired, a second process of syntactic bootstrapping can speed up this process by exploiting structural similarity between linguistic elements. This can take place even entirely within language (hence in a purely symbolic way without influence from the world). The most probably meaning of a new word can be estimated on the basis of the prior probability established by previously encountered words of the same semantic and syntactic type in similar syntactic and semantic contexts. For example, if a child knows the meaning of “open the box” and then hears the sentence “open the closet”, it can infer that a “closet” denotes a thing that can be opened (rather than a word meaning the same thing as “open”) without ever having seen one ([25], [26] see [27] for a comparison between semantic and syntactic bootstrapping). Essentially this amounts to inference of the syntactic and semantic type of an unknown word from its grammatical role and the surrounding context of probabilistically known words. These two generalization mechanisms are very powerful and allow young humans to acquire language without explicit instruction. It is arguable that bootstrapping is what fuels the explosion in language and conceptual development that occurs around the third year of child development [28], [26].

There is also a link between action and language. [29] provided a generative grammar describing the structure of



action. This grammar has both computational applicability and a biological basis.

### C. Conclusion

In the current paper we combined computer vision based action representation and classification with a bootstrapping process to accelerate (non-linguistic) acquisition of action knowledge in a robot. As discussed above, structural bootstrapping performs a comparison of the meaning (semantics) of actions at the level of SECs and – if required – then in a second step a comparison of its individual syntactic elements (descriptors  $d$ ). This way it becomes for the first time possible to perform the rather complex aspects of Accommodation and Assimilation [9] in a formal and algorithmically sound way with a robot-compatible action encoding. The resulting categorization allows for a better understanding of the underlying actions and their cognitive meanings. In [10] we demonstrate that the here-introduced action representation can be used to execute the respective action with a robot. Thus, learning the representation from observation together with robot execution does - we think - provide a substantial contribution to the field of cognitive robotics

### REFERENCES

- [1] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, pp. 233–242, 1999.
- [2] A. Billard, S. Calinon, and F. Guenter, "Discriminative and adaptive imitation in uni-manual and bi-manual tasks," *Robot. Auton. Syst.*, vol. 54, pp. 370–384, 2006.
- [3] M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zöllner, "Incremental Learning of Tasks From User Demonstrations, Past Experiences, and Vocal Comments," *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics*, vol. 37, no. 2, pp. 322–332, 2007.
- [4] S. Ekvall and D. Kragic, "Robot learning from demonstration: a task-level planning approach," *International Journal of Advanced Robotic Systems*, vol. 5, no. 3, pp. 223–234, 2008.
- [5] R. Cubek and W. Ertel, "Learning and Execution of High-Level Concepts with Conceptual Spaces and PDDL," in *3rd Workshop on Learning and Planning, ICAPS (21st International Conference on Automated Planning and Scheduling)*, 2011.
- [6] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter, "Learning the semantics of object-action relations by observation," *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1229–1249, 2011.
- [7] E. E. Aksoy, A. Abramov, F. Wörgötter, and B. Dellen, "Categorizing object-action relations from semantic scene graphs," in *IEEE International Conference on Robotics and Automation (ICRA)*, may 2010, pp. 398–405.
- [8] E. E. Aksoy, B. Dellen, M. Tamosiunaite, and F. Wörgötter, "Execution of a dual-object (pushing) action with semantic event chains," in *Proceedings of 11th IEEE-RAS International Conference on Humanoid Robots*, 2011, pp. 576–583.
- [9] J. Piaget, *The Origins of Intelligence in the Child*. London, New York: Routledge, 1953.
- [10] M. J. Aein, E. E. Aksoy, M. Tamosiunaite, J. Papon, A. Ude, and F. Wörgötter, "Toward a library of manipulation actions based on semantic object-action relations," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (submitted), 2013.
- [11] A. Abramov, K. Pauwels, J. Papon, F. Wörgötter, and B. Dellen, "Real-time segmentation of stereo videos on a portable system with a mobile GPU," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 9, pp. 1292–1305, 2012.
- [12] A. Abramov, E. E. Aksoy, J. Dörr, K. Pauwels, F. Wörgötter, and B. Dellen, "3d semantic representation of actions from efficient stereo-image-sequence segmentation on GPUs," in *5th International Symposium 3D Data Processing, Visualization and Transmission*, 2010, pp. 1–8.
- [13] T. Kulvicius, K. J. Ning, M. Tamosiunaite, and F. Wörgötter, "Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 145–157, 2011.
- [14] J. A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proc. 2002 IEEE Int. Conf. Robotics and Automation*, 2002, pp. 1398–1403.
- [15] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control—a unifying view," *Prog. Brain Res.*, vol. 165, pp. 425–445, 2007.
- [16] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance," in *Proc. 2009 IEEE Int. Conf. Robotics and Automation*, 2009, pp. 1534–1539.
- [17] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Trans. Robot.*, vol. 26, no. 5, pp. 800–815, 2010.
- [18] F. Wörgötter, E. E. Aksoy, N. Krüger, J. Piater, A. Ude, and M. Tamosiunaite, "A simple ontology of manipulation actions based on hand-object relations," *IEEE Transactions on Autonomous Mental Development*, (In press), 2012.
- [19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, nov 2004.
- [20] N. Badler, "Temporal scene analysis: Conceptual descriptions of object movements," Ph.D. dissertation, University of Toronto, Canada, 1975.
- [21] M. Sridhar, G. A. Cohn, and D. Hogg, "Learning functional object-categories from a relational spatio-temporal representation," in *Proc. 18th European Conference on Artificial Intelligence*, 2008, pp. 606–610.
- [22] H. Kjellström, J. Romero, and D. Kragic, "Visual object-action recognition: Inferring object affordances from human demonstration," *Comput. Vis. Image Underst.*, vol. 115, no. 1, pp. 81–90, jan 2011.
- [23] Y. Yang, C. Fermüller, and Y. Aloimonos, "Detection of manipulation action consequences (mac)," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, (In press), 2013.
- [24] K. Ikeuchi and T. Suehiro, "Toward an assembly plan from observation, part I: Task recognition with polyhedral objects," *IEEE Trans. Robotics and Automation*, vol. 10, no. 3, pp. 368–385, June 1994.
- [25] S. Pinker, *Language Learnability and Language Development*. Cambridge: Cambridge University Press, 1984.
- [26] J. Trueswell and L. Gleitman, "Learning to parse and its implications for language acquisition," in *Oxford Handbook of Psycholinguistics*, Oxford, 2007, pp. 635–656.
- [27] G. Chierchia, "Syntactic bootstrapping and the acquisition of noun meanings: the mass-count issue," in *Heads, Projections and Learnability Volume 1*, B. Lust and J. W. MARGARITA Suner, Eds. Hillsdale, New jersey, 1994, pp. 301–318.
- [28] F. Tracy, "The language of childhood," *Am. J. Psychol.*, vol. 6, no. 1, pp. 107–138, 1893.
- [29] K. Pastra and Y. Aloimonos, "The minimalist grammar of action," *Philosophical Transactions of the Royal Society B*, vol. 367, pp. 103–117, 2012.

# Learn to Wipe: A Case Study of Structural Bootstrapping from Sensorimotor Experience

Martin Do, Julian Schill, Johannes Ernesti, Tamim Asfour

**Abstract**—In this paper, we address the question of generative knowledge construction from sensorimotor experience, which is acquired by exploration. We show how actions and their effects on objects, together with perceptual representations of the objects, are used to build generative models which then can be used in internal simulation to predict the outcome of actions. Specifically, the paper presents an experiential cycle for learning association between object properties (softness and height) and action parameters for the wiping task and building generative models from sensorimotor experience resulting from wiping experiments. Object and action are linked to the observed effect to generate training data for learning a non-parametric continuous model using Support Vector Regression. In subsequent iterations, this model is grounded and used to make predictions on the expected effects for novel objects which can be used to constrain the parameter exploration. The cycle and skills have been implemented on the humanoid platform ARMAR-IIIb. Experiments with set of wiping objects differing in softness and height demonstrate efficient learning and adaptation behaviour of action of wiping.

## I. INTRODUCTION

The efficiency with which humans perform manipulation tasks in unstructured and dynamic environments is unattained by robotic systems. The key to this remarkable performance lies in the human cognitive capabilities which enables the autonomous acquisition of knowledge by processing complex sensor information and the application of this knowledge to rapidly explore unknown scenes, objects, and actions. Intelligent robots must be able to rapidly create new concepts and react to unanticipated situations in the light of previously acquired knowledge by making generative use of experience utilizing predictive processes. This process is largely driven by internal models based on prior experience (Inside-out). Such robots must also be able to help and learn from others by sharing these generative, experience based theories through teaching and interaction. During development, stimulus driven outside-in and internally driven inside-out processes need to interact with each other at the earliest possible moment to drive the development of cognitive capabilities. The development of such cognitive capabilities has to be embedded in a learning process in order to verify, extend, and revise this knowledge. Hence, in order to make a crucial step towards more autonomy, robots have to be equipped with a similar capabilities.

In [1], the concept of *Structural Bootstrapping* has been introduced to address how generative mechanisms which

rely on prior knowledge and sensorimotor experience can be implemented in robotic systems and employed to speed up learning. Structural Bootstrapping – an idea taken from child language acquisition research - is a method which provides an explanation of how the language acquisition process in infants is initiated. Hence, in robotic context, Structural Bootstrapping can be seen as a method of building generative models, leveraging existing experience to predict unexplored action effects and to focus the hypothesis space for learning novel concepts. This developmental approach enables rapid generalization and acquisition of new knowledge about objects, actions and their effects from little additional training data. Entities of the world are represented in form of Object-Action Complexes (OAC) – affordance-based object-action associations and can be understood as semantic sensorimotor categories, which are computable (learnable) and storable in a robot system (see [2]). OACs are related to state-actions transitions and incorporate object as well as action affordances. This allows the specification of actions based object percepts and vice versa enables the grounding of object representations based on the execution and observation of the actions and their effects. Based on the OAC representation, knowledge structures in the form of internal models are generated and intrinsically grounded. The benefit of this knowledge acquisition approach becomes particularly evident on the sensorimotor level where object and action embedded in a situational context are closely intertwined. The experience gained by actively exploring and interacting with the environment, objects and other agents and by observing the effect of actions is characterized by the specific embodiment. Therefore, representations and models emerging from this experience are better adapted to the robot's morphology and more suitable to capture the sensorimotor contingencies than those generated by traditional disembodied methods. The continuous grounding of internal models and representations through exploration provides a suitable basis for prediction and simulation.

In this paper, we provide an example for Structural Bootstrapping and demonstrate the validity of the approach on the sensorimotor motor level. Embedded in a learning cycle we show how generative models describing the relation between object properties and action parameters can be learned from experience and how these models can be used to make predication using internal simulation. More specifically, we show in the context of table wiping task how action parameters can be predicted and adapted based on object's softness and size.

This work was not supported by any organization  
M. Do, J. Schill, T. Asfour are with the Faculty of Informatics, Institute for Anthropomatics, High Performance Humanoid Technologies, Karlsruhe, Germany  
{martin.do, julian.schill, asfour}@kit.edu

## II. RELATED WORKS

Several approaches in the literature deals with the problem of exploration-based learning and generative model construction. In the following we give an overview on approaches related to the work presented in this paper. In [3], an affordance learning framework is introduced which models dependencies between actions and object features in the form of Bayesian Network. Using a set of manipulation actions (grasp tap touch) and based on object features effect of the performed action such as object motion and tactile information could be estimated. In [4], a interactive learning scheme is introduced which allows the identification of object grasp affordances. Based on a grasp representation in the form of a dynamic movement primitive learned from human demonstration. Point cloud representation of the object in order to parameterize the grasp primitive. Effect observation if grasp successful or not. especially field of grasping. In [5], an approach is presented for the learning object grasp affordance through exploration. These affordances are represented by grasp densities which associated with a specific visual object model. To refine these affordances to an object of interest, an initial grasp densities is determined based on the visual features (3D edges) which characterize the object appearance. Through exploration by having the robot applying grasps encoded within the initial densities and observing whether a grasp was successful, the object grasp affordances are grounded and grasp densities are refined. In [6], an approach is introduced which enables a robot to learn a grasping behaviour in based on initial reflex-like motor primitives. The execution of these primitives at different speeds and the observation of the tactile feedback when touching an object leads to the generation of further behaviour primitives. To link the resulting behaviour to different intrinsic and extrinsic object properties, the primitives are executed and the observed effects are categorized using the SVM. in [7], a system is described which allows the learning of manipulation strategies without any prior knowledge. Based on a tray-tilting action, the robot explores the parameter by performing an experiment which involves the execution of a sequence of differently parameterized tilting actions and observing the their effects on an arbitrary object placed on the tray. The experimental is used to create graph structure consisting of states and transition which enables the prediction of a future state based on the perceived object properties and the intended action. For the scenario of object-pushing, in [8], a method is proposed which enables a robot to learn goal-directed push-locations on multiple objects. Based on a parameterizable pushing action, In order to push an object to a designated goal location, stable pushing movements which move the object along a straight line are desired. Using a the Support Vector Regression method a model is learned from explorative pushing which allow the prediction of the effect (a pushing score which indicates the deviation from a straight line) of certain pushing action considering the current object shape and pose.

## III. THE LEARNING CYCLE

In order to enable a robotic system to learn and refine sensorimotor knowledge within a developmental process, a learning cycle has to be formalized which incorporates perceptual and motor skills. As suggested in [9], the presented learning cycle consists of four stages (Fig. 1). For our work, we define initial stage to be the exploration stage. Given generalized representations of objects and actions, the robot explores the scene in order to obtain instantiations of both, object and action. The resulting action and object representation  $A_0$  and  $P_0$  form the basis of an experiment which is conducted in the subsequent stage to create data from which concrete experience can be generated. The robot applies the action  $A_0$  and observes its effect  $E_0$ . In the third stage, based on the data  $D = (P_0, A_0, E_0)$  experience is created by grounding and adapting the representations. In the modelling stage, knowledge in the form of an internal model  $f$  which links observed effect to the perceived object and the performed action in the following form:

$$E_0 = f(P_0, A_0). \quad (1)$$

is extracted from this experience. To acquire larger data which provide a more accurate description of the object-action relationships embedded in the current experimental context, within a learning iteration the experiment can be repeated with different parameters. This way the parameter space can be efficiently explored. In subsequent iterations  $i$  with  $i > 0$  the grounding is transferred to novel perceived object representation  $P_i$ . Using  $f$  a prediction of the expected effect  $\hat{E}_i$  for  $P_i$  and an action  $A_i$  can be made. This prediction can be used to constrain and control the exploration of the action parameter space within the repeated experiment and, thus, leads to less, however, more relevant additional training data which has to be considered for the re-grounding the representations and revision of the internal models. Hence, this learning cycle allows the continuous acquisition, validation, and refinement of internal knowledge in long term association through exploration and predictive reasoning.

In order to enable a robotic system to learn and refine sensorimotor knowledge within a developmental process, a learning cycle has to be formalized which incorporates perceptual and motor skills. As suggested in [9], the presented learning cycle consists of four stages. For our work, we define the initial stage to be the exploration stage. Given generalized representations of objects and actions, the robot explores the scene in order to obtain instantiations of both, object and action. The resulting action and object representation  $A_0$  and  $P_0$  form the basis of an experiment which is conducted in the subsequent stage to create data from which concrete experience can be generated. The robot applies the action  $A_0$  and observes its effect  $E_0$  on object, environment, and on the robot itself. In the third stage, based on the data  $M = (P_0, A_0, E_0)$  experience is created by grounding and adapting the representations. In the modelling stage, knowledge in the form of an internal model  $f : (P, A) \rightarrow E$  which links the observed effect to the perceived object and the performed action is extracted from this experience.

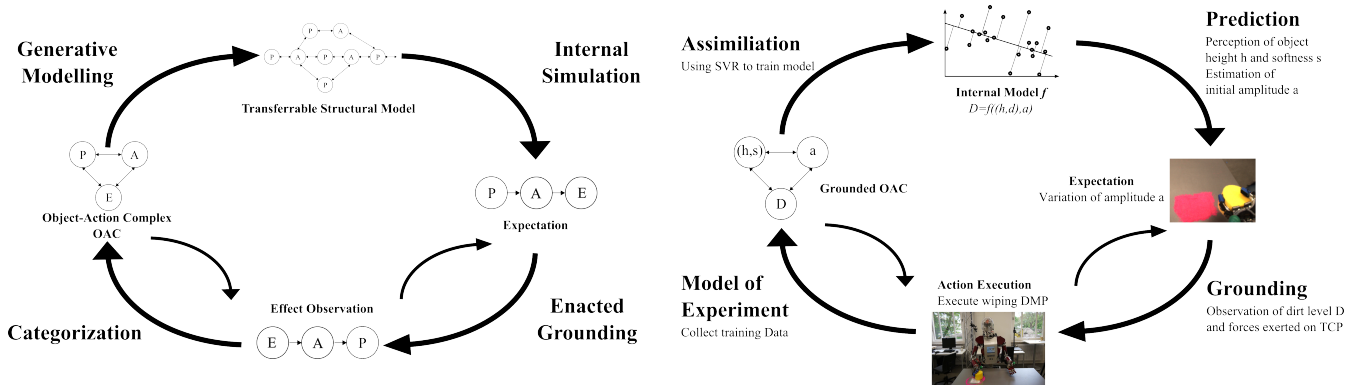


Fig. 1: Learning Cycles.

In subsequent iterations  $i$  with  $i > 0$  the grounding is transferred to novel perceived object representation  $P_i$ . Using  $f$  as a prediction function, the expected effect  $\hat{E}_i$  for  $P_i$  and the parameters for action  $A_i$  can be estimated. This prediction can be used to constrain and control the exploration of the action parameter space within the repeated experiment and, thus, leads to less, however, more relevant additional training data which has to be considered for the re-grounding the representations and revision of the internal models. Hence, this learning cycle allows the continuous acquisition, validation, and refinement of internal knowledge in a long term association through exploration and predictive reasoning.

#### A. Instantiation of the Learning Cycle for Wiping

Based on the learning cycle described in III, a behaviour is implemented which enables a robot to efficiently learn wiping movements with different objects. Using skills which have been implemented on our platform, the learning cycle has been instantiated as depicted in Fig. 1. To accelerate the learning process, observations of human wiping demonstrations trigger the bootstrapping process and provide data based on which a coarse representation of the wiping action can be inferred. The wiping action is represented in the generalized form of a periodic Dynamic Movement Primitive (see Section ??). In the initial iteration, the robot is focused on the adaptation of this representation to environmental circumstances, namely the surface to be wiped. This step corresponds to the grounding of the action representation.

In subsequent iterations, the robot attempts to establish the link between a object, action, and effect. For the object perception, a skill (as described in Section IV-B) is applied which enables the robot to deform an object and based on the extent of the deformations allows the determination of the object's height and softness. To generate different wiping movements the amplitude parameter of the primitive can be adapted. To assess the effect of a wiping movement the robot visually determines the dirt level (see Section ??) describing the ratio between the amount of remaining dirt enclosed by an area to be wiped and the entire wiping area size. Hence, in these iterations, the action parameter space is explored for the grounded movement primitive in order to generate a wiping movement adapted to perceived object.

For each stage, a separate experiment is specified. However, the goal for both experiments remains the same: wipe until the dirt level does not change. For a dirt levels  $d_i, d_{i-1} \in \mathbb{R}$  determined in iteration  $i$  and  $i-1$ , the goal can be formalized as follows:

$$d_i - d_{i-1} \leq d_\epsilon \quad (2)$$

where  $d_\epsilon$  denote a threshold at which the dirt level change can be disregarded.

#### B. Surface Adaptation

The grounding of the wiping DMP corresponds the adaptation of the DMP in order to attain goal-directed wiping movement. In the context of wiping, one prerequisite is constant contact of the object and the surface to be wiped. Therefore, wiping movements can only be adequately evaluated and adapted based on the forces exerted on the robot's end effector. Based on a wiping primitive which encodes a periodic movement pattern  $p_w$  in a  $(x, y)$ -plane parallel to the surface, we wish to adapt the movement to the shape of the surface. Following the force profile adaptation method introduced in [10], a force-feedback control mechanism is implemented which moves the end effector towards the surface while executing the wiping pattern. In this work, we restrict ourselves to the wiping of flat surfaces. Hence, for a periodic wiping trajectory  $p_w(t) = (x_w(t), y_w(t))$  with  $T_s < t < T_e$  and  $T_s, T_e$  denoting the start and end time of a period, a movement  $z_w(t)$  with each discrete time step  $\delta t$  is determined according to following equation:

$$\dot{z}_w(t) = k_f(f_{z_w}(t) - f_0)z_w(t) = z_0 + \dot{z}_w(t)\delta t. \quad (3)$$

Here,  $z_0$  stands for the initial height from which the wiping movement is initiated,  $f_0$  denotes the desired force with which the robot should press an object towards the surface,  $f_{z_w}(t)$  is the measured force on the end-effector, and  $k_f$  describes a force gain factor. A further simplification which allows which allows a safer execution of the experiment is to replace  $f_{z_w}$  with  $f_{z_w} = \sqrt{f_x^2 + f_y^2 + f_z^2}$ , since it forces to robot to move upwards when the robots collides with anything from any direction. As a result, the experiment leads to data triplet center of the wiping area  $p_0 = (x_0, y_0)$ :

$$(P, A, E) = (p_0, (p_w, z_w), d) \quad (4)$$



based on which the action representation is grounded and extended.

### C. Action Parameter Exploration

To attain an optimal wiping behaviour with a specific object, the wiping action has to be parameterized according the object properties. This can be accomplished by specifying the amplitude with which the a wiping action is executed. To find a suitable parameterization, the action parameter space is explored within the wiping experiment based on the forces acting on the robot. Starting from an initial estimate  $a_0$ , the amplitude is varied according following rules:

$$a(t) = \begin{cases} b^- a(t-1) & , f_{z_w}(t) - f_O > \gamma, \dot{z}_w < 0 \\ b^+ a(t-1) & , f_{z_w}(t) - f_O > \gamma, \dot{z}_w > 0 \\ b^+ a(t-1) & , f_{z_w}(t) - f_O < -\gamma, \dot{z}_w < 0 \\ b^- a(t-1) & , f_{z_w}(t) - f_O < -\gamma, \dot{z}_w > 0 \\ a(t-1) & \text{else} \end{cases} \quad (5)$$

where  $0 < b^- \leq 1$  and  $b^+ = 2 - b^-$  denotes a scalar factors which decreases respectively increase the amplitude according the current movement direction and exerted forces.  $a(t-1)$  represents the amplitude estimate made in the previous time step. For each iteration  $i$ , the overall amplitude factor  $a_i$  is calculated by  $a_i = \frac{1}{T_E - T_S} \sum_{t=T_S}^{T_E} a_i$ . The data which results from the experiment, can be described as follows: for the current object wiping:

$$(P, A, E) = ((s, h), a, d). \quad (6)$$

This data matrix provides the basis for the inference of an internal model.

### D. Learning of Internal Model

To generate an internal model representing the relationships between perception, action, and effect, computational methods have to be applied which are suitable to identify structures from non-linear data of arbitrary dimensionality without any prior knowledge. In this work, the Support Vector Regression (SVR), a supervised learning technique which is described in [11], is applied to approximate such a model, since it allows to capture complex relationships between the training data points. Furthermore, a sparse model can be obtained by applying the Support Vector method which facilitates the processing of large datasets and enhances the prediction and simulation using the internal model. Based on our experimental data collection  $\{(P_n, A_n, E_n)\}_{i=1, \dots, N}$ , a training dataset  $M$  with  $N$  input/output pairs is formed as follows::

$$M = \{(x_n, y_n)\}_{n=1, \dots, N}, \quad x_i = (P_i, A_i), \quad y_i = (E_i). \quad (7)$$

The internal model is described by  $f: x \rightarrow y$ . Finding a non-linear mapping appropriate function  $f$  solves the learning problem and leads to desired model enabling the mapping of an arbitrary input pair  $(P, A)$  on expected effect  $E$ . Usually, the search for  $f$  is performed by determining an approximation  $\hat{f}$  minimizing the risk functional:

$$R_{emp}[\hat{f}] := \frac{1}{N} \sum_{n=1}^N d(\hat{f}(x_n), y_n) \quad (8)$$

with  $d(f(x), y)$  being a distance function to define the relation between the model's output  $\hat{f}(x)$  and the correct output  $y$ .

Using the Support Vector method, the non-linear regression problem incorporated in Eq. 8 is transformed into linear problem by introducing a non-linear mapping  $\theta: \mathbb{R} \rightarrow \mathbb{R}^{N_h}$  which projects the original dataset  $M$  into a feature space of higher dimensionality. Hence, the SVR consists of finding a hyperplane  $(w, b)$  which satisfies:

$$g(x, w) = \sum_{j=1}^{N_h} w_j \theta_j(x) + b. \quad (9)$$

To determine a linear model which captures most training samples within an  $\varepsilon$ -margin, an  $\varepsilon$ -loss-insensitive function is defined as follows:

$$L_\varepsilon(g(x, w), y) = \begin{cases} 0 & \text{if } |g(x, w) - y| \leq \varepsilon \\ |g(x, w) - y| - \varepsilon & \text{else} \end{cases} \quad (10)$$

is introduced into the risk functional. Hence, the risk functional to be minimized can be written as follows.

$$R(C) = C \frac{1}{N} \sum_{i=1}^N L_\varepsilon(g(x, w), y) + \frac{1}{2} \|w\|^2 \quad (11)$$

Eq. 11 can be rewritten as following optimization problem. Our goal is to find a function  $f$  whose distance to any given data point does not exceed  $\varepsilon$  while being as flat as possible. This optimization problem can be described in Smola kernel functions suffice for the approximation of the mapping:

$$\text{minimize } \tau(w) = \frac{1}{2} \|w\|^2 + C \sum (\zeta + \zeta^*) \quad (12)$$

$$\text{subject to } y_i - (k(w, x_i) - b) \leq \varepsilon \quad (13)$$

$$\text{subject to } (k(w, x_i) + b - y_i) \leq \varepsilon \quad (14)$$

where  $\zeta$  are slack variables which are introduced to the problem in order to relax the constraints and to add a soft margin to the hyperplane and thus to tolerate a small error. Especially, in the case of nonlinear high dimensional data or data containing noise this.

## IV. IMPLEMENTATION

The implementation of the wiping learning behaviour is based on skills which already exist on the robot which allow learning and cognition. In the following, the skills and eventual modifications which have been made in order to combine them are briefly described.

### A. Wiping Skill

To enable a robot to learn and adapt wiping movements, a skill has been implemented which creates a generalized action representation of a wiping movement. In this work, wiping movements are encoded as periodic Dynamic Movement Primitives (DMP) using a slight extension of the DMP formulation as suggested in [12] which allows the representation of a periodic motion as well as its corresponding

discrete transient movement. In general, a DMP consists of two parts:

$$\begin{cases} \dot{s}(t) = \text{Canonical}(t, s), \\ \dot{v}(t) = \text{Transform}(t, v) + \text{Perturbation}(s). \end{cases} \quad (15)$$

$$(16)$$

The perturbation term in (16) is adapted to a demonstrated trajectory whereas the transformation system allows the generalization of the learned trajectory to new start and goal conditions. The canonical system defines the state of the DMP in time and drives the perturbation to control the transformation system. The encoding of both, periodic and transient motion, is accomplished by introducing a two-dimensional canonical system in the DMP formulation: a dimension  $r$  to describe distance from the periodic pattern and  $\phi$  denoting the phase of the periodic pattern. This yields the state of the DMP  $s(t) := (\phi(t), r(t))$  as the solution  $(\phi, r)$  of the following ordinary differential equation:

$$(17) \begin{cases} \dot{\phi} = \Omega, \\ \dot{r} = \eta(\mu^\alpha - r^\alpha)r^\beta. \end{cases} \quad (17a)$$

$$(17b)$$

Here,  $\mu > 0$  denotes the radius of the limit cycle and  $\eta, \alpha, \beta > 0$  are constants. The value of  $\Omega > 0$  defines the angular velocity of  $\phi$  and has to be chosen according to the period  $p$  of the desired trajectory, i.e.  $\Omega = \frac{2\pi}{p}$ . The value of  $\phi$  is linearly increasing whereas  $r$  converges monotonously to  $\mu$ . Thus, by interpreting  $(\phi, r)$  as polar coordinates the solution of (17) converges towards a circle with radius  $\mu$  around the origin on the phase plane. The transformation system is a critically-damped spring system with one global point attractor given by

$$\begin{cases} \dot{z} = \Omega \left( \alpha_z (\beta_z (g - v) - z) + a \cdot f(\phi, r) \right), \\ \dot{v} = \Omega z. \end{cases} \quad (18)$$

The constants  $\alpha_z, \beta_z > 0$  are chosen according the ratio  $\frac{\alpha_z}{\beta_z} = \frac{4}{1}$  in order to ensure critical damping. With  $f \equiv 0$  the system state  $v$  converges to the anchor point  $g \in \mathbb{R}$ . By adapting  $f$  corresponding to the demonstrated trajectory the system oscillates around  $g$  in a similar manner as featured the demonstration. Here,  $f$  is defined as

$$f(\phi, r) = \frac{\sum_{j=1}^M \psi_j(\phi, r) \tilde{w}_j + \sum_{i=1}^N \varphi_i(\phi, r) w_i}{\sum_{j=1}^M \psi_j(\phi, r) + \sum_{i=1}^N \varphi_i(\phi, r)}, \quad (19)$$

where  $W := (w_1, \dots, w_N, \tilde{w}_1, \dots, \tilde{w}_M)^T \in \mathbb{R}^{N+M}$  contains the weights which can be adjusted to fit the desired trajectory. The basis functions  $\psi_j$  vanish onto the limit cycle of (17) and hence encode the the transient part of the motion. In contrast to that, the basis functions  $\varphi_i$  are used to encode the periodic pattern since they vanish outside the limit cycle of (17). The special choice of  $\psi_j$  and  $\varphi_i$  guarantees a smooth transition from the transient to the periodic pattern, cf. [12].

The factor  $a > 0$  is changed on-line during the reproduction of the motion to modulate the amplitude<sup>1</sup>. For the learning process the value is  $a = 1$ .

<sup>1</sup>Note that this factor is not part of the original formulation in [12].

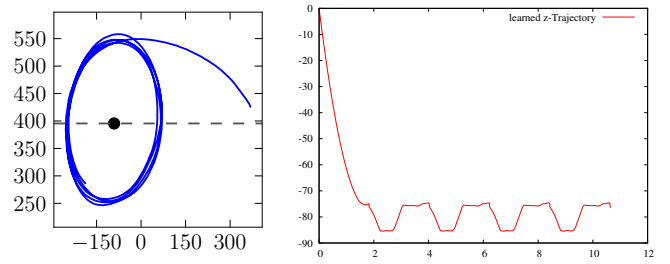


Fig. 2: A generated displacement trajectory  $z_w$ .

Since the transient motion can be learned by the chosen DMP formulation, no ad-hoc procedure is needed to initialize the periodic wiping motion. Rather than that, the pattern can be initiated in a well defined way from different initial configurations.

The learning of a wiping movement is decoupled in two phases: the learning of the wiping pattern from human observation and the adaptation of a wiping movement primitive to the surface to be wiped. In the first phase, motion data representing human wiping demonstrations gets segmented to identify the transient part and the periodic pattern. Then the weights in (19) are calculated to make the system reproduce the demonstration. Since a two dimensional trajectory on the plane surface to be wiped is learned in Cartesian coordinates, this has to be done for two dimensions separately. This yields the system  $(C, \tau_x, \tau_y)$ , where  $C$  is an instance of the canonical system (17) and  $\tau_x, \tau_y$  are instances of the transformation system (18) each of which encoding one coordinate, i.e.  $x_w$  or  $y_w$ , of the wiping trajectory. The transformation systems  $\tau_x$  and  $\tau_y$  are synchronized by the common canonical system  $C$ .

In the second phase, the adaptation to the surface is performed. To achieve that the robot repeats the learned motion by integrating  $(C, \tau_x, \tau_y)$ . By executing the trajectory the robot finds for each point  $(x_w(t), y_w(t))$  of the periodic pattern a displacement  $z_w(t)$  that points into the wiped surface. This displacement  $z_w(t)$  is chosen according to the surface adaptation policy introduced in Sec. III-B. The trajectory  $z_w = (z_w(t))_{t=T_s, \dots, T_e}$  gets extended by a transient that encodes a smooth transition from  $z_0$  to the determined displacement trajectory of the periodic part. Then, an additional transformation system  $\tau_z$  is trained with the resulting trajectory  $z_w$ . Fig. ?? shows an example of a displacement trajectory  $z_w$  generated in this way.

Summarizing, we obtain the system  $(C, \tau_x, \tau_y, \tau_z)$  encoding the wiping pattern and a displacement trajectory to keep the normal force constant.

## B. Softness Skill

To check the deformability and softness of an object the robot uses his ability to control the grasping force of the pneumatic actuated hand with a model based force position control [13]. When the object is in the hand and grasped between the finger tips with a low grasping force, the distance between the finger tips of the index finger, middle

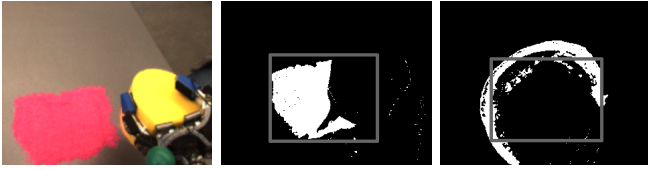


Fig. 3: Left: Robot view on the scene. Center: Segmented view of the scene in the beginning of the wiping execution. Right: Segmented view on a "clean" table.

finger and thumb is measured using the joint encoders and the forward kinematics. Then the grasping force is increased which results in a deformation of the object. After the fingers have stopped moving, the distance between the finger tips is measured again and the difference of the distances is used as a measure for the softness of the object.

### C. Effect Checking Clean Table

As mentioned before, the effect of a wiping action is described by the dirt level  $d$  within area  $O$  to be wiped. For the sake of simplicity, it is assumed that dirt features a specific color. Therefore, to determine the size and position of  $O$ , using the stereo camera setup the robot explores the table and performs a color segmentation in order to localize the largest blob. A bounding box  $B_i$  around that blob provides the image coordinates of  $O$ . Transformed into the world coordinate system, one obtains  $B_w$  which provide the global coordinates of  $O$ . In order to determine the current dirt level at any time  $t$  during the execution of the wiping experiment,  $B_w$  is transformed back onto image coordinates  $B_i^t$ . Hence, based on  $B_i^t$   $d$  can be calculated according to the following equation:

$$d = \sum_{i=y_{min}}^{y_{max}} \sum_{j=x_{min}}^{x_{max}} \frac{k(i,j)}{(x_{max} - x_{min})(y_{max} - y_{min})}. \quad (20)$$

Since the hand might occlude a considerable area of the surface, a reliable assessment of the dirt level cannot be performed at guaranteed any time during the execution of a wiping movement. Hence, to control the experiment, the current dirt level at  $t_c$  is set to  $d(t_c) := d_{max,i}$  which is defined as follows:

$$d_{max,i} = \max \{d_i(t)\}_{T_{s_i} < t < T_c} \quad (21)$$

with  $i$  denoting the index of the current period. The experiment is finished when following conditions are fulfilled as described by

## V. EXPERIMENTS

As depicted in Fig. 4, the implemented learning behaviour has been evaluated on our humanoid platform ARMAR-IIIb (see in []). The learning of the wiping primitive in the initial iteration is described in. In subsequent iterations, to facilitate the environmental perception, the color of the dirt (pink sand) has been specified. Based on this information, the robot initializes each learning iteration by localizing the dirty area  $O$ . The corresponding bounding box  $B_i$  is used to specify the target configuration of the DMP. In the following

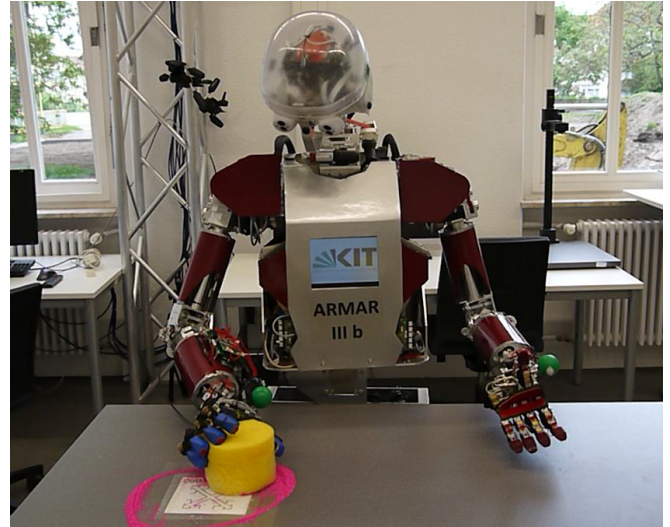


Fig. 4: The humanoid platform ARMAR-IIIb wiping the table with a sponge.

step, the robot determines the object softness and height by grasping the object at the bottom and top side of the object. The object exploration process is assisted by human operator since for wiping the object has to be reoriented in the robot's hand, so that the object is grasped from the side enabling the bottom to touch the table. Given the internal models, predictions are made for the amplitude and the expected effect. Subsequently, the robot performs a wiping movement with a and compares the observed effect with the expected effect. If the observations do not coincide with the expectation, a parameter exploration procedure as described in Sec amplitude is initiated in order to create further data for the grounding of the internal models. For now, the grounding of an internal model is done by updating the data set and retraining the entire model. We are aware that the learning cycle has to incorporate an incremental learning algorithm in order to be effective for the longer term and with an increasing amount of data.

Therefore, in this section, results of preliminary experiments are presented showing the effectivity of the experience learning cycle for the implementation of a cognitive learning behaviour for robots, in particular, in the context of wiping. The wiping experiments have been conducted on a set of twelve objects which includes instances designed for wiping (sponges, towel, toilet paper) and other household items (box, bottle, ball, can) that are less suitable. We restrict ourselves on objects whose height and weight are within a predefined range in order to prevent damage to the robot. Based on experimental data originating from wiping experiments with this object set, internal models  $f_1(P) = A$  and  $f_2(P,A) = E$  are generated using the support vector regression method. In this work, we used the libsvm library introduced in [14] for the training. The relevant parameters for the training of  $f_1$  have been determined to be  $C = 50$  and  $\gamma = 0.5$  whereas  $f_2$  has been trained with  $C = 10$  and  $\gamma = 0.33$ . The data and predictions of the amplitudes

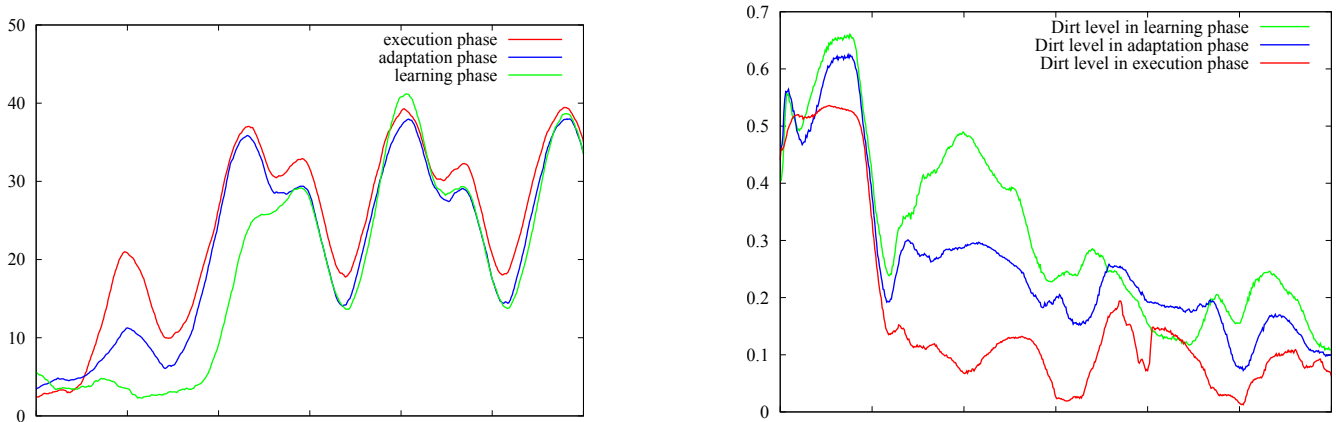


Fig. 5: Left: Trajectories of forces exerted on the endeffector in various phases of the wiping learning cycle. Right: The dirt level evolution in various phases of the wiping learning cycle.

and the expected dirt levels are listed in Table I. It is interesting to see that for soft objects the amplitude could be reliably re-estimated. The main reason for the variation of the amplitudes for harder objects lies in the increased sensitivity towards forces exerted on the object respectively the end-effector. A slight difference of the object pose in hand can produce very different results. Regarding the prediction of the expected dirt level, good estimations could be made for cubic objects. For spherical and cylindric objects, less useful predictions have been inferred.

Given a percept of a specific object, the corresponding amplitude estimate can be used to considerably reduce the adaptation effort of a wiping movement. The plots depicted in Fig. 5 indicate that with increasing knowledge leading to more accurate estimations of the action parameter the execution of an action converges faster towards the desired behaviour. With regard to the forces exerted on the end-effector, a force trajectory is desired which oscillates around the predefined force threshold of  $F_0 = 25$  whereas regarding the dirt level we wish to minimize the dirt level as fast as possible. The learning phase denotes the initial phase where the movement primitive is adapted to the environment. In the adaptation phase, based on a default value of  $a = 1$  the amplitude is varied in order to attain the desired effect. In the execution phase, the task is performed using the estimated amplitude parameter and without any adaptation.

## VI. CONCLUSION

An approach for the implementation of a cognitive learning behaviour enabling robots to create individual knowledge structures based on experience gained through physical exploration, interaction, and observation has been proposed. The behaviour manifests in the form a learning cycle which incorporates perceptual and motor skills in order to continuously acquire data based on which internal models are generated and grounded. For the scenario of table-wiping, we have showed that with these internal models further wiping primitives can be efficiently learned and adapted to different task and object-specific constraints.

Object	$h$	$s$	$\hat{a}$	$a$	$\hat{d}$	$d$
sponge (s)	79	0.0343	1.0	1.0	0.162	0.117
sponge (m)	91	0.0384	0.957	0.948	0.129	0.132
sponge (l)	102	0.0358	1.13	1.139	0.139	0.09
styrofoam cube (s)	87	0.00474	0.701	0.696	0.270	0.258
styrofoam cube (l)	91	0.00774	1.0	1.0	0.13	0.177
rolled towel	89	0.0213	1.215	1.057	0.229	0.142
styrofoam ball	100	0.00639	1.497	1.496	0.384	0.422
cardboard box	91	0.0171	1.072	1.072	0.240	0.178
plastic bottle	87	0.0263	1.453	1.453	0.310	0.568
metal can	86	0.00843	1.04	1.366	0.352	0.529
toilet paper	101	0.0232	0.887	0.887	0.162	0.128
foam	91	0.041	0.999	1.0	0.13	0.177

TABLE I: Object properties and the corresponding action and effect parameter.  $h$  denotes the object height in mm and  $s$  the softness of an object.  $\hat{a}$  and  $a$  represent the estimated and the actual amplitude of an adapted wiping movement.  $\hat{d}$  and  $d$  stand for the expected and actual dirt level which indicates the effect of wiping.

However, we have also experienced cases in which the estimation of action parameters and the prediction of the expected effect failed. This is mainly due to the simple object representation which merely relies on the object softness and height. As indicated in our results, for deformable objects, these features might suffice in order to determine the affordance in the context of wiping. To be able to determine the affordances for a variety of objects further object properties have to be considered such as the geometry, weight, and surface character. Therefore, in the future we will focus on the integration of an enriched object representation which allows the estimation of further action parameters such as different hand orientations or wiping patterns. Furthermore, we will conduct extensive experiments with numerous objects with the goal of enabling a robot to extend the knowledge structures.

## ACKNOWLEDGEMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement no. 270273 (Xperience).

## REFERENCES

- [1] “Xperience Project,” Website, available online at <http://www.xperience.org>.

- [2] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, A. Agostini, and R. Dillmann, "Object-action complexes: Grounded abstractions of sensorimotor processes," *Robotics and Autonomous Systems*, vol. 59, pp. 740–757, 2011.
- [3] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory motor coordination to imitation," *Transactions on Robotics*, vol. 24, no. 1, pp. 15–26, February 2007.
- [4] O. Kroemer, E. Ugur, E. Oztop, and J. Peters, "A kernel-based approach to direct action perception," in *IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, USA, May 2012, pp. 2605–2610.
- [5] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater, "Affordance prediction via learned object attributes," *Journal of Behavioral Robotics*, vol. 2, no. 1, pp. 1–17, March 2011.
- [6] 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 (IROS)*, Vilamoura, Portugal, October 2012, pp. 3260–3267.
- [7] A. D. Christiansen, M. T. Mason, and T. M. Mitchell, "Learning Reliable Manipulation Strategies without Initial Physical Models," *Robotics and Autonomous Systems*, vol. 8, no. 1–2, pp. 7–18, November 1991.
- [8] T. Hermans, F. Li, J. M. Rehg, and A. Bobick, "Learning stable pushing locations," in *IEEE International Conference on Developmental Learning and Epigenetic Robotics (ICDL-EPIROB)*, Osaka, Japan, August 2013.
- [9] D. Kolb, *Experiential learning: experience as the source of learning and development*. Englewood Cliffs, NJ: Prentice Hall, 1984.
- [10] A. Gams, M. Do, A. Ude, T. Asfour, and 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, USA, December 2010.
- [11] A. Smola, "Support Vector Regression," *International Journal of Robotics Research*, vol. 30, pp. 1229–1249, September 2011.
- [12] J. Ernesti, L. Righetti, M. Do, T. Asfour, and S. Schaal, "Encoding of periodic and their transient motions by a single dynamic movement primitive," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Osaka, Japan, December 2012, pp. 57–64.
- [13] A. Bierbaum, J. Schill, T. Asfour, and R. Dillmann, "Force Position Control for a Pneumatic Anthropomorphic Hand," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Paris, France, 2009.
- [14] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

# An active learning based sampling design for structural bootstrapping

Sandor Szedmak  
IIS, University of Innsbruck  
sandor.szedmak@uibk.ac.at

Justus Piater  
IIS, University of Innsbruck  
justus.piater@uibk.ac.at

January 15, 2014

## Abstract

Collecting a sufficiently representative sample for robot learning in a complex environment is a hard and open problem. Since gathering proper, reliable data about objects, effects of an action, or action series is very expensive, therefore that kind of data collection needs to be reduced to the minimum, but in the same time the collected data should be representative to guarantee the adequate decisions. In this paper that challenging problem is addressed via an active learning based sampling design. First an initial training set is built up on the available knowledge given by the geometry of the domain. Then via predicting the unknown cases by the maximum margin based multi-valued regression learning(MMMVR) framework the corresponding confidence measure is used to select the most informative cases to extend iteratively the initial training set. At the end the effectiveness of the proposed learning framework is demonstrated by comparing it to a random and to a fully informed, expert advice based, theoretically optimal approaches.

## 1 Introduction

One of the most challenging task in statistics and in machine learning is to collect a sufficiently representative sample of several variables with strong interdependence. This problem is even more exposed in those cases where a measurement providing a sample item has high cost, for example gathering information on affordances realized in a real robot environment. It is not only expensive but could be very time consuming as well.



To illuminate the real extent of the problem we provide a seemingly simple example. Let us assume that we are given a set of objects,  $\{o_1, \dots, o_{n_O}\}$ ,  $n_O = 100$  a collection of tools, utensils, foods of a kitchen, different type of spoons, bowls, pans, vegetables etc. We are going to test the potential interactions between these objects, e.g. cut a carrot with a knife, by trying a set of different actions,  $\{a_1, \dots, a_{n_A}\}$ ,  $n_A = 10$ , e.g. cutting, rolling, pushing.

Suppose that the robot has prior knowledge on the shapes of the objects, i.e. it can recognize them, but their functions are unknown, i.e. the knife is a proper tool to cut. Therefore we might need to check all possible interactions between all pairs of the objects and actions. This can lead to 100000 experiments not assuming any repetition. To carry out that series of experiments is obviously infeasible on a real robot. One might try to apply any robot simulator to reduce that huge number, but a faithful realistic simulation of the potential physical interactions between those objects requires sophisticated modules, e.g. modeling the solidness of an object, which could be as expensive and time consuming as the real robot experiments.

We can approach the interaction learning problem from a different point of view by exploiting the results of those experiments which we executed earlier. If a knife was suitable to cut an object it might cut other objects as well. First we give a rather informal description of the task we need to solve and in the sequel we move towards a formal approach. The inputs of the experiments can be given by a tuple (action =  $a_i$ , tool =  $t_j$ , target object =  $o_k$ ). There exists a function  $F$  expressing the result of the experiment. For the sake of simplicity  $F$  is chosen as a partially defined Boolean valued function saying the action  $a_i$  by applying the tool  $t_j$  on  $o_k$  was successful, *True*, or not, *False*, and if the outcome of the experiment is unknown then  $F$  gives ?. If  $a_i = \text{"cut"}$ ,  $t_j = \text{"knife"}$  and  $o_k = \text{"carrot"}$  and the experiment was successful then we might claim that on the set of tuples ( $a_i = \text{"cut"}$ ,  $t_j = \text{"knife"}$ ,  $o_k = ?$ ), where ? could be any available object, the outcome function  $F$  might yield the *True* value with sufficiently high probability.

To describe the connection between the known experiment ( $a_i = \text{"cut"}$ ,  $t_j = \text{"knife"}$ ,  $o_k = \text{"carrot"}$ ) and an unknown one, ( $a_i = \text{"cut"}$ ,  $t_j = \text{"knife"}$ ,  $o_k = ?$ ), a multi-layered graph  $G$  can be defined. Let all possible tuples form the set of vertexes  $\mathcal{V}$ . The edges of  $G$  are sorted into three layers ( $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ ) corresponding to the elements of the tuples, namely the first layer is built on the first elements, actions, and the other layers are similarly created. In one layer two tuples is connected if the corresponding element of both tuples are the same. Furthermore to each vertex the outcome function  $F$  assigns the values *True*, *False*, ?. Now the outcome of an unknown experiment given by a tuple might be predicted by those tuples which are connected to that and their outcome are exactly known, namely equal to *True* or *False*.

This layered graph structure can motivate a sampling strategy which could be much less expensive than checking all possible tuples in a real robot

environment. The strategy can be summarized in the following points:

- Choose those experiments, tuples, from which all other missing ones can be reached via the corresponding edges.
- If there is no prior knowledge on interactions then the variance of the prediction can be reduced if the known sample items are chosen as uniformly as possible, i.e. the sample distribution should follow the maximum entropy principle. See below the block sampling strategy of combinatorial design theory.

## 2 Sampling model

First we formalize the case where trying all tuples on the robot is possible, which also means all tuples could be predicted.

### 2.1 The complete model

We are given three sets of entities, namely actions  $\mathcal{A}$ , tools  $\mathcal{T}$  and target objects  $\mathcal{O}$ . All these entities are represented by labels uniquely identifying them, thus the sets contain those labels. The set of tools and target objects might overlap each other, even they might coincide.

To describe an experiment carried out in a real robot environment a set of tuples of all possible combination is used, formally that set is defined by  $\mathcal{E} = \{(a, t, o)\}$ ,  $a \in \mathcal{A}, t \in \mathcal{T}, o \in \mathcal{O}$ . Suppose that there is a function  $F$  defined on  $\mathcal{E}$  which can provide the outcome of an experiment given by a tuple of  $\mathcal{E}$ . This function might yield logical values  $\{True, False\}$  or a real number of  $[0, 1]$  qualifying the outcome of the experiment, e.g. the probability of success. In describing the sampling strategy we use the logical output as the values of  $F$ . The value of this function on one  $e \in \mathcal{E}$  is given us if the corresponding experiment has been realized. In this sense we can talk about known or unknown values of  $F$ . It is needed to emphasize that the function  $F$  is well defined, but only our knowledge is valid only on the realized experiments.

Our task can be the selection of a subset  $\mathcal{S}$ , a sample, of  $\mathcal{E}$  for which the values of  $F$  are known and we would like to predict the values of  $F$  on the complement  $\mathcal{E} \setminus \mathcal{S}$  of  $\mathcal{S}$ . We might call  $\mathcal{S}$  as training and  $\mathcal{E} \setminus \mathcal{S}$  as test. To solve this task we are going to find a function  $\hat{F} : \mathcal{E} \rightarrow \{False, True\}$ , an estimation of  $F$ . The estimation error, the risk, can be defined by

$$\hat{R} = \sum_{e \in \mathcal{E} \setminus \mathcal{S}} (F(e) \neq \hat{F}(e)). \quad (1)$$

The objective is to minimize this error. To reduce the cost of finding of training is another part of the objective, namely we need a training set,  $\mathcal{S}$ ,



which is as small as possible. To achieve a satisfactory trade off between the prediction error and the training cost the following symbolic objective can be defined

$$\min_{\mathcal{S}, \hat{F}} \frac{\sum_{e \in \mathcal{E} \setminus \mathcal{S}} (F(e) \neq \hat{F}(e))}{\text{card}(\mathcal{E} \setminus \mathcal{S})} + C \frac{\text{card}(\mathcal{S})}{\text{card}(\mathcal{E})}, \quad (2)$$

where  $C$  is the trade off constant balancing the influence of the terms on the value of the objective. The second term can be interpreted as regularization. The keyword card denotes the cardinality of the sets. The purpose of the scaling factors in the denominators are to reduce the effect caused of the different cardinalities. Here we focus on the second term, the selection of the set  $\mathcal{S}$ . The function  $\hat{F}$  is looked for by the learning method described in the Appendix of [11], and earlier versions of the method for a special application is detailed in [5] and [4].

To find a sufficiently small training set we can exploit the structure of  $\mathcal{E}$ . The method proposed here arises from the sampling theories provided in Algebraic Statistics, see for example in these books *Lectures on Algebraic Statistics* ([8]) and *Algebraic Statistics for Computational Biology* ([9]). Within the broad range of Algebraic Statistics the theory of combinatorial design should be mentioned, see in *Handbook of Combinatorial Designs* ([6]). Beside the theoretical foundation of these kind of sampling approaches, sophisticated software implementation is available for example in the freely downloadable open-source collection of mathematical programs of the Sage Mathematics, [3], in the sub-packages of Combinatorics, Designs and Incidence Structures, see details on [www.designtheory.org](http://www.designtheory.org). The use of algebra based methods have recently arise in those areas of machine learning and statistics where the strong interdependence can not be ignored, for example in deep learning and in other areas, see for example in [13].

The basic idea is to equip the space  $\mathcal{E}$  with a certain finite geometry structure  $\mathfrak{G}$ . In this geometry we can define a neighborhood and connections to a given  $e \in \mathcal{E}$ . If this neighborhood contains some elements of the training set  $\mathcal{S}$  then those elements can provide the knowledge to predict the value of  $F$  at  $e \in \mathcal{E}$ .

**Example 1.** Let  $e = (\text{cut}, \text{knife}, \text{carrot})$ . Assume that this experiment has not been realized yet, but the experiments relating to  $e = (\text{cut}, \text{knife}, \text{cucumber})$ ,  $e = (\text{cut}, \text{cleaver}, \text{carrot})$  and  $e = (\text{chop}, \text{knife}, \text{carrot})$  have. The overlaps between the tuples can glue together the items to allow the available knowledge to transfer to the case whose outcome is unknown.

We can define the following elements of the finite geometry,  $\mathfrak{G}$ :

**Point:** The point of this geometry is given by the elements, the tuples  $(a, t, o)$ , of  $\mathcal{E}$ .

**Line:** The lines going through of a point  $e = (a_i, t_j, o_k)$  are given by these three sets

$$\ell_{e=(a_i, t_j, o_k)} = \begin{cases} \{(a_i, t_j, o)\} & o \in \mathcal{O}, \\ \{(a_i, t, o_k)\} & t \in \mathcal{T}, \\ \{(a, t_j, o_k)\} & a \in \mathcal{A}. \end{cases} \quad (3)$$

**Plane:** The planes going through of a point  $e = (a_i, t_j, o_k)$  are given by these three sets

$$\pi_{e=(a_i, t_j, o_k)} = \begin{cases} \{(a_i, t, o)\} & t \in \mathcal{T}, o \in \mathcal{O}, \\ \{(a, t_j, o)\} & a \in \mathcal{A}, o \in \mathcal{O}, \\ \{(a, t, o_k)\} & a \in \mathcal{A}, t \in \mathcal{T}. \end{cases} \quad (4)$$

We say a line(plane) is common for two points if both points contained by that line(plane).

We can see these statements are valid in the geometry  $\mathfrak{G}$ :

- To any two distinct points of  $\mathfrak{G}$  there is at most one line containing them.
- Any two distinct lines meet(intersect) each other at most one point. If the intersection is empty then we say the lines are parallel.
- Any three distinct points are contained at most one plane.
- Any two distinct planes meet(intersect) each other at most one line. If the intersection is empty then we say the planes are parallel.
- Any three distinct planes meet(intersect) each other at most one point.

Note that we do not need to define any order on the sets of action, tools and target object since any permutation of the labels of those entities preserves the structure of the geometry described above.

Now we can define how the cardinality of the training set  $\mathcal{S}$  can be minimized while the predictive capability can be preserved. Here we mention two basic cases:

**Strong connection:** Find  $\mathcal{S}$  such that for every  $\tilde{e} \in \mathcal{E} \setminus \mathcal{S}$  all the lines going through on  $\tilde{e}$  contain at least one element of the training set  $\mathcal{S}$ .

**Weak connection:** Find  $\mathcal{S}$  such that for every  $\tilde{e} \in \mathcal{E} \setminus \mathcal{S}$  all the planes going through on  $\tilde{e}$  contain at least one element of the training set  $\mathcal{S}$ .

In the strong case a predictable element  $\tilde{e}$  share two components with the training items lying on the common lines, in the weak case only one component is common with  $\tilde{e}$ . Obviously we can create a mixed model falling between the weak and the strong model which contains lines and planes as well.

To estimate the cardinality of  $\mathcal{S}$  let us assume that

$$\text{card}(\mathcal{A}) = \text{card}(\mathcal{T}) = \text{card}(\mathcal{O}) = n. \quad (5)$$

The lines going through one point consist of  $3n - 3$  other points, and the planes going through one point consist of  $3n^2 - 3n$  other points. Therefore if we would like to cover all test items then we have a lower bound  $n^3/(3n-3) \sim n^2/3$  on the cardinality of  $\mathcal{S}$  in the strong connection case, and similarly we have  $n^3/(3n^2 - 3n) \sim n/3$  in the weak case. These bounds are relatively weak since here we assumed no overlaps between the covering lines and planes.

If all tuples are given then the exact minimum can be computed, but the real case can fall significantly away from that. An approximation of a possible real cases is outlined in the next subsection.

## 2.2 Incomplete model

Combining arbitrary actions, tools and target objects into triplets can provide cases which might be dangerous, or the outcome of the corresponding experiment turns to be useless, for example cutting a bottle, a glass by a knife should not be tried at all. It means, in advance the possible experiments have to be qualified with respect to their feasibility. Therefore only a subset  $\mathcal{D}$  of  $\mathcal{E}$  can be the target of the learning procedure. Unfortunately the distribution of  $\mathcal{D}$  can be entirely unknown, the elements of that subset can spread either uniformly or by following a very complex pattern.

To cover the incomplete case we can apply an approximation of the set covering or vertex covering problems of combinatorial optimization, see descriptions, details in [1] or [10]. since the set, and similarly vertex, covering problems are NP-hard, an approximation schema has to be applied. These kind of approximations are generally fast, but suboptimal.

A simple greedy algorithm can be implemented by considering the cover realized by those sets which are union of the three lines going through of a element of  $\mathcal{D}$ . In the vertex cover framework we need additional structure to define a graph where all elements of  $\mathcal{D}$  give the vertexes and there is an edge between two vertexes if the corresponding tuples can be connected by a line or a plane in the geometry defined on  $\mathcal{E}$ . Obviously these two approaches fundamentally yield the same model.

We have as many covering sets as many points, a one-to-one correspondence between points and sets. Let the set belonging to  $d \in \mathcal{D}$  be denoted by  $v(d)$ . The greedy algorithm to derive a set covering can be given as

**Step 0**  $\mathcal{S} = \mathcal{D}$ ,  $\mathcal{C} = \emptyset$ .

**Step 1** Choose that set  $v(d)$  which covers the largest number of elements of  $\mathcal{S}$ ,

**Step 2**  $\mathcal{S} = \mathcal{S} \setminus v, \mathcal{C} = \mathcal{C} \cup \{d\}$ .

**Step 3** Repeat from **Step 1** until  $\mathcal{S} = \emptyset$ !

[1] contains detailed analysis of this algorithm. We can consider weighted covering problems making differences between the vertexes, for example by the confidence of the prediction - see below. In the weighted case the algorithm proposed by Bar-Yehuda and Evan can be a good candidate, see in [1] as well.

### 3 Coverage of object interaction table required by the MMMVR

In the maximum margin based multi-valued regression(MMMVR), - presented in appendix of [11], and earlier versions for a special application in [5] and [4], - requires kernelization of the incomplete sparse tables. The next question arises: what kind of level of sparseness can be allowed to provide a sufficiently good prediction. To answer this question we present the following approach.

Let us fix an action, then we have a table whose row labels are the tools and the column labels are the target objects. This table is only given partially, since the vast majority of the elements are missing, unknown. Since we are going to apply a kernel learning method on this table, where the inner products are computed between two rows(tow columns) we need to guarantee that the rows(columns) containing elements in the same position, in the same columns(rows), otherwise the inner product can not be computed. To be more concrete, assume that the known interaction between tools and target object are given by a confidence value, a real number expressing the success of the fixed action. The model can then be formulated as in the following way. The items of the matrix corresponding the table is given by  $\mathbf{M} = \mathcal{T} \times \mathcal{O}$ , the known items are given by  $\mathcal{D} \subseteq \mathcal{T} \times \mathcal{O}$ . In what follows we assume linear kernel function between the rows, tools, hence an element of the kernel matrix  $K_{ij}$  connecting row  $i$  and  $j$  has the value

$$K_{ij} = \sum_{(ik) \in \mathcal{D}, (jk) \in \mathcal{D}} M_{ik} M_{jk}, \quad (6)$$

since it can be computed only on those pairs which are known. However if  $\mathcal{D}$  is a very sparse subset of  $\mathcal{T} \times \mathcal{O}$  then only the diagonal elements of the kernel matrix can be computed and no interactions are captured.

we assume a probabilistic model of data acquisition to estimate the minimum number of known object pairs needed in the training set to derive a nontrivial kernel matrix . The non-triviality of the kernel matrix means in

this context that the corresponding kernel matrix is not diagonal and contains sufficiently amount of off-diagonal elements capturing the interaction between rows describing the items.

For sake of simplicity assume that the number of tools and target objects are the same and equal to  $n$ . Furthermore each elements of the matrix  $\mathbf{M}$  is known with probability  $p$ , and this probability is given uniformly and independently to each element. Then two rows can contain known elements in the same fix position with probability  $p^2$  which is a consequence of the independence of the elements. Based on our assumption the number of elements occurring in the same column in two rows follows binomial distribution with parameters  $p^2$ , and  $n$ . Therefore we have at least one known elements appearing in the same columns with probability

$$q = 1 - (1 - p^2)^n, \quad (7)$$

since the binomial distribution might yield 0 common elements with probability  $(1 - p^2)^n$ . Suppose that if there are at least one common elements of two rows then the inner product is not zero. Now if we expect  $k$  non-zero elements in a row of the kernel matrix, (and by the symmetry of the kernel matrix in the corresponding column), then the value of  $q$  can be estimated by assuming that the expected number of the non-zero elements is equal to  $k$ , and it is also equal to  $qn$ . It is true because each element of the kernel matrix can be non-zero with probability  $q$ , and in the row of kernel matrix we have  $n$  inner products. Therefore we have

$$\begin{aligned} k &= nq = n (1 - (1 - p^2)^n) \\ \Rightarrow p &= \left(1 - \left(1 - \frac{k}{n}\right)^{\frac{1}{n}}\right)^{0.5} \end{aligned} \quad (8)$$

Based on probability  $p$  we can estimate the number of known pairs as function of expected non-zero elements,  $k$ , in a row of the kernel matrix. The relationship between  $p$  and  $k$  is displayed in Figure 1 assuming that the fixed table size is equal to 100. In the next Figure 2  $p$  is shown up again the changing table size in case of different expected non-zero kernel matrix items,  $k$  in any row.

Figure 1 shows that the minimum number of training items could be equal to  $\sim 142$  when the entire table contains  $10000 = 100^2$  elements, thus a significantly small training set can be used to drive the learning method.

## 4 Active learning

Based on the previous sections we can build a general learning strategy which requires a significantly small training set to start the accumulation of the information about the interaction of actions and objects. After starting on the initial training set we need to choose how to acquire the next items,

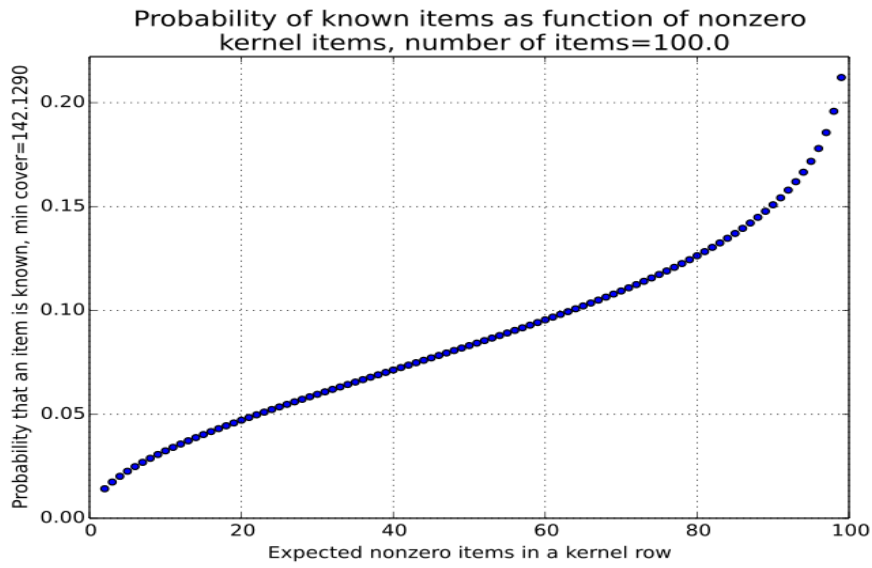


Figure 1: The probability of a pair of tool and target object, vertical axes, needs to be known to achieve the a certain number of non-zero elements of the kernel matrix in one row(column) of the kernel matrix, horizontal axes, assuming that the table size is equal to 100

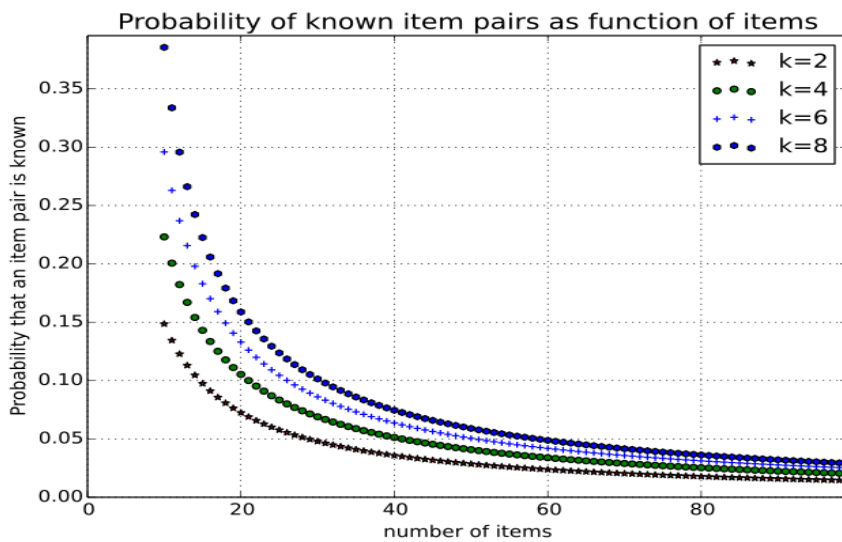


Figure 2: The probability of a pair of tool and target object, vertical axes, needs to be known to achieve the a certain number  $k$  of non-zero elements of the kernel matrix in one row(column) of the kernel matrix as function of changing the table size displayed in the horizontal axes.

to select the most promising experiment defined on the untried tuples. Here three alternatives are presented.

**Random:** The tuples are randomly chosen out of the pool of untested cases. It can serve as base line to compare methods which exploits the knowledge collected in the earlier phases of the learning.

**Expert advice:** Assume that there is an expert, supervisor, who can evaluate the prediction on the untested items, and provides the worst cases, the predictions which was made by high confidence but they were incorrect. It provides the best, but otherwise a very expensive strategy, Any feasible strategy needs to lie between the random and the expert advice based strategies. A very detailed description and analysis of those kind of learning approaches which are built on expert advice can be found in [2].

**Selection by confidence** The prediction is provided on the untried elements, and those items are selected where the confidence of the outcome is the lowest. This strategy is a feasible one and its implementation can have affordable low cost even in very large data sets.

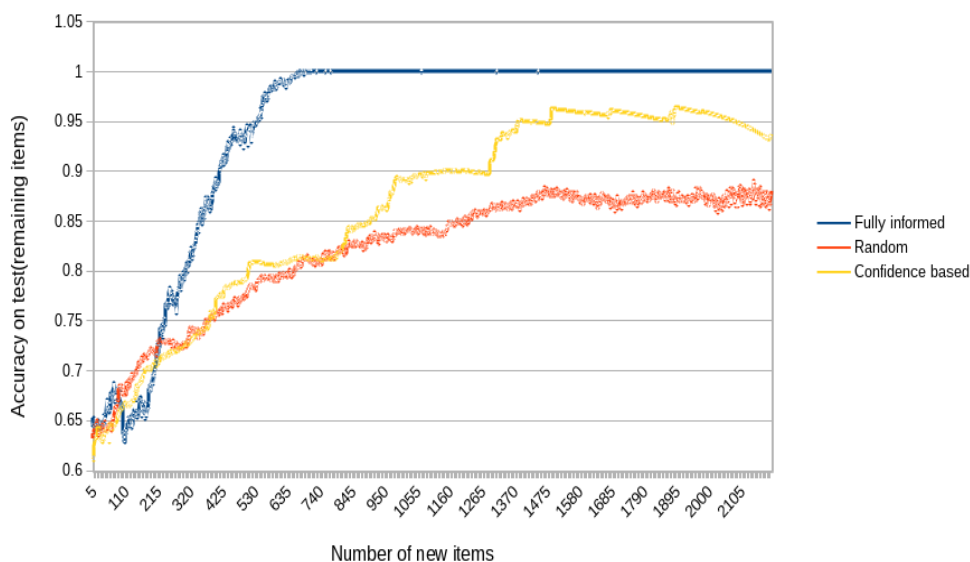
Here we assume that the learner can provide a confidence measure to each prediction, a real number chosen out of the interval  $[0, 1]$ , where 0 shows the lowest and 1 the highest confidence.

To demonstrate the performance of these methods we use the data set described in [14]. The learning approach follows that which is described in the appendix of [11]. All these methods are implemented by splitting the data set into two parts, a preselected training set and the remaining part is used as pool to pick up the new items to incorporate them into the training. The preselected training is chosen by set covering to minimize its size to the minimum by following the methods described in Sections 2 and 3

Figure 4 shows the evaluation of the accuracy

Since we applied our method on a finite pool of the possible tuples therefore picking up the unknown worse case elements from the test set, that set contains less and less hardly predictable elements where with high confidence the prediction result can be improved. This is what we can observe especially in case of the expert advice based learning, where at a certain point the test consists only of “good” remaining points. The finite pool setting can be achieved by deciding in advance on the set of possible sample items we are going to test and predict. This is a generally applied strategy in medical statistic where the statistical design theory is applied, see for example in [6]. One can illuminate this situation in an application of the Support Vector Machine(SVM). If we systematically select the support vectors and errors to incorporate them in the training set then remaining items can be found mostly above the hyperplanes provided by the SVM, hence they are well predictable points.

Accuracy evolution in acquiring new items



## 5 Future work

The learning approach presented in this paper will be applied on the affordance data base containing action-object interactions collected in real robot environment, see details in [12]. This data base is being created and will contain object features, Kinect, 2D images and videos of the scene as well, thus the high level features of the interactions and the corresponding sensorimotor data can be combined. Another data base where the method will be applied is the Berkeley 3-D Object Dataset, [7], where the table is constructed on the pairs of functional description of the objects and features extracted from their 2D images and the corresponding Kinect depth images taken of the same point of view.

**Acknowledgement 2.** *The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.*

## References

- [1] Korte Bernhard and Vygen Jens. *Combinatorial Optimization, Theory and Algorithms*. Springer-Verlag, Berlin, 2000.



- [2] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning and Games*. Cambridge University Press, 2006.
- [3] W. Stein et al. *Sage Mathematics Software, Versions: 4.7.1 and 5.3*. The Sage Development Team, 2010-12. <http://www.sagemath.org>.
- [4] M.A. Ghazanfar, A. Prugel-Bennett, and S. Szedmak. Kernel mapping recommender system algorithms. *Information Sciences*, 2012. accepted.
- [5] M.A. Ghazanfar, S. Szedmak, and A. Prugel-Bennett. Incremental kernel mapping algorithms for scalable recommender systems. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Special Session on Recommender Systems in e-Commerce (RSEC)*. 2011.
- [6] Colbourn C. J. and Dinitz J. H. *Handbook of Combinatorial Designs*. Boca Raton, Chapman & Hall, CRC, 2nd edition.
- [7] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T. Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3-d object dataset: Putting the kinect to work. In *ICCV Workshop on Consumer Depth Cameras for Computer Vision*.
- [8] Seth Sullivant Mathias Drton, Bernd Sturmfels. *Lectures on Algebraic Statistics*, volume Oberwolfach Seminars, Vol 40. Birkhauser, 2009.
- [9] L. Pachter and B. Sturmfels. *Algebraic Statistics for Computational Biology*. Cambridge University Press, 2005.
- [10] A. Schrijver. *Combinatorial optimization, polyhedra and efficiency*. Springer, Algorithms and Combinatorics, 24, 2003.
- [11] Sandor Szedmak. Learning object-action relations via knowledge propagation. Technical report, University of Innsbruck, 2012. Technical report.
- [12] Emre Ugor. Bootstrapping multi-object affordance learning using learned single-affordance features. Technical report, University of Innsbruck, 2014.
- [13] Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, 2009.
- [14] Hanchen Xiong, Sandor Szedmak, and Justus Piater. Homogeneity analysis for object-action relation reasoning in kitchen scenarios. In *2nd Workshop on Machine Learning for Interactive Systems, (Workshop at IJCAI)*, page 3744. 2013.

# Bootstrapping multi-object affordance learning using learned single-affordance features

Emre Ugur, Sandor Szedmak, and Justus Piater

Intelligent and Interactive Systems, Institute of Computer Science,  
University of Innsbruck

**Abstract**—The aim of this paper is to propose a learning framework where a developmental robotic system benefits from structural bootstrapping where structural similarities are learned and encoded within affordance relations. In this context, we show that how complex affordance learning can be bootstrapped through using pre-learned basic-affordances encoded as additional features. In the first stage, the robot learns affordances in the form of developing classifiers that predict effect categories given object features for different discrete actions applicable to single objects. These predictions are added to robot’s feature set as higher-level *affordance features*. In the second stage, the robot learns more complex multi-object affordances using object and affordance features. We first applied our idea in an artificial interaction database which include discrete actions, several manually coded object categories, and actions effects. Finally, we validated our bootstrapping approach in a real robot with poke and stack actions. We showed that complex affordance learning significantly speeds up with predictors that are bootstrapped with *affordance features* compared to predictors that use low-level features such as shape descriptors.

## I. INTRODUCTION

Even in the newborn infant, a basic neuro-muscular infrastructure for simple manipulative ‘move hand’ action with reach and grasp components, is present [1]. Within several months, the infant transforms his/her initial seemingly uncontrolled ‘move hand’ action into a set of action primitives such as grasping, hitting and dropping for single objects[2]. From that age, their sensorimotor skills develop to enable them to achieve progressively more complex tasks. For example human infants start inserting rods into circular holes in a box or stack up blocks into towers of two blocks from 13 months[3]. While in 13 months, the infants can insert circular rods into circular holes in a plate, by 18 months they can perceive the correspondance between different shaped blocks and they start inserting different shaped blocks into corresponding holes [4]. Studies with infant chimpanzees also revealed that there is a dramatic increase in exploration of object-object combinations at around 1.5 years of age while such combinatory actions were at a very low frequency before that period for several months [5]. Such development patterns suggest non-gradual development changes in human infants as well [6]. This data suggests that the infants first develop basic skills (such as fine grasping and object rotation) that are precursors of combinatory manipulation actions. They also probably use the learned action grounded

object properties in further development of complex action affordances.

Language-learning children of age 3-4 years old are observed to acquire the meaning of words very quickly compared to younger ones. Semantic and syntactic bootstrapping processes are assumed to play a significant role in this learning where children use co-occurring components of the sentences and syntactic/structural similarity between linguistic components in order to speed up learning new words and concepts [7]. Although there has been no reported experiments related to bootstrapping in other domains, it is plausible to assume that such structural relationships and similarities play a significant role in speeding up infant’s sensorimotor learning and generalization.

The aim of this paper is to propose a basic learning framework where a developmental robotic system benefits from structural bootstrapping where structural similarities are learned and encoded within affordance relations. In detail, our robot learns the affordances of single objects and uses these affordances as additional features in the next stages of development where multi-object affordances are discovered. The use of learned structural similarities in the form of affordances are expected to bootstrap the learning in the next stages.

Our approach can be explained by the following intuitive example: Let us assume that the robot learned rollability affordances of the objects in the first stage and can predict the rollability based on object shape properties. In the next stage of learning multi-object affordances such as stackability, with only two sample interactions where balls tumble over and boxes piled up, the learning system can directly find a correspondance between stackability and rollability. Then, given cylindrical objects, the robot can make better predictions for stackability depending on the roll orientation (and affordance) of the cylinders.

In the context of robot affordance learning research, multi-object affordance learning has not been studied extensively with exceptions of [8] where ‘tools objects’ are interacting with other objects, and [9] where two-object relational interaction models were directly learned. However none of these studies attempted to bootstrap their multi-object affordance learning system with previously obtained skills.

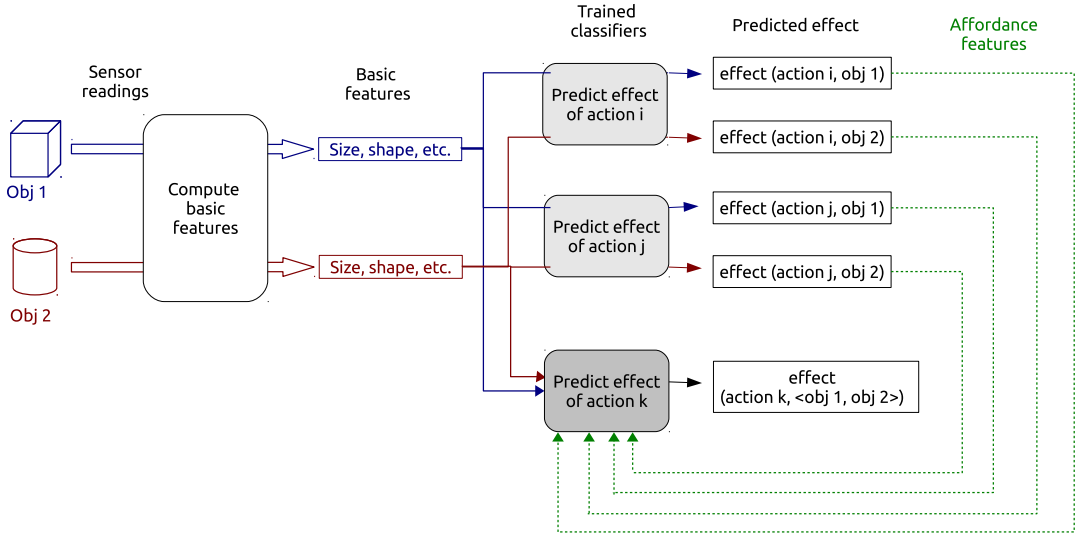


Fig. 1. Learning and prediction of action effects using *basic-features* are shown with solid lines. The learned *affordance-features*, i.e. predicted effects, can be further used as input to classifiers which predict multi-object action effects, i.e. in learning multi-object affordances.

## II. METHOD

Learning of affordances corresponds to learning the relations between objects, actions and effects [10], [11]. In this study, affordances are acquired through learning to predict what type of effects, i.e. discrete effect categories, can be generated given discrete robot actions and continuous object properties. To achieve this, we simply train a classifier for each action, which takes object features as input and predicts the effect category.

Here, we distinguish two different sets of object features. The first set includes hand-coded basic general-purpose features, computed mostly from visual perception with no explicit link to robot’s actions. These may include standard features used in literature related to size, shape and part properties of the objects. The second set of features are acquired through interaction and corresponds to the higher-level learned ones that are computed from basic features. They encode the dynamics between robot actions and object response (effect). The first set of features is called *basic-features* whereas the second that is learned through interaction is called *affordance-features* as it includes the relations between objects, actions and effects.

The straightforward approach to learn effect prediction is to find a function that takes *basic-features* and actions as input:

$$f_{\text{basic}}(\text{action}, \text{basic-features}) \rightarrow \text{effect}$$

whereas we propose to speed up learning of complex effect prediction using *affordance-features* that are computed using the learned basic effect prediction:

$$f_{\text{complex}}(\text{action}, \text{basic-features}, f_{\text{basic}}()) \rightarrow \text{effect}$$

which in a *flat* form corresponds to:

$$f_{\text{complex}}(\text{action}, \text{basic-features}, \text{affordance-features}) \rightarrow \text{effect}$$

Our approach is summarized in Fig. 1. The features shown with blue and red solid lines correspond to *basic-features* and action predictions based on these features give rise to *affordance-features*. The dashed lines correspond to *affordance-features*, that are learned in previous stages. The learning and prediction of complex affordances benefit from previously learned affordance features as shown in ‘Predict effect of action k’ predictor. No that action k is considered to be a complex action as two objects are involved in execution.

Our approach that is summarized in Fig. 1 is limiting in several aspects. The most severe limitation is on the unidirectional structure of the prediction and learning. However, this learning architecture can still be improved significantly. In its current version, affordance features are univariate values that correspond to the effect predicted to be generated by the discrete action. This can be replaced with a structural model that summarizes the predicted effect distribution given basic object features for an action with continuous parameters. For example, continuous grasp densities [12] can be learned and used as *affordance-features* to learn complex actions that involve multiple objects.

Complex affordance learning can be realized in different ways. In this paper, the action possibilities that are provided by two (or more) objects are considered to be complex. For instance, the effects created by a *stack* action (where the object is grasped and released over another one) is determined by the properties of both objects. We will use *affordance-features* (such as rollability, pushability, etc) and *basic-features* to learn and predict stackability affordances, and show that this learning significantly speeds up with predictors that are bootstrapped with *affordance-features*.

### III. BOOTSTRAPPING IN SIMULATION

In this section, we report our bootstrapping results obtained from an artificial database of objects and interactions. For this purpose, we compared *basic-features* and *affordance-features* based affordance classifiers that are trained to predict effect of *stack* action. We showed that learning benefits from bootstrapping through use of *affordance-features*.

#### A. Artificial affordance experiment database

We used an artificial object set with cylinders, boxes, spheres and triangular prisms in different orientations and with/without holes. When poked from different directions, different effects can be generated with these objects. For example, when poked from side, lying cylinders will roll away, boxes will be pushed, objects with holes in poke direction will not be affected as finger would go through the hole without any interaction, and tall objects will topple down. The set of manually encoded actions and their effects are as follows:

- Actions: {side-poke, top-poke, front-poke, stack}
- Poke-effects: {pushed, rolled, toppled, resisted, nothing}
- Stack-effects: {piled-up, inserted-in, covered, tumbled-over}

The effect of stacking objects on top of each other depends on their relative size. For example, while ‘inserted-in’ effect is generated when a small box is stacked on a hollow cylinder, ‘piled-up’ effect is observed when the box is larger than the opening on top of the cylinder. Based on these assumptions, we created a hypothetical set of rules that return the effect based on object categories and their relative sizes. Fig. 2 gives these rules where stack, inside, outside and fail correspond to piled-up, inserted-in, covered and tumbled-over.  $d$ ,  $w$ , and  $h$  correspond to dimensions in different axes; depth, width and height, respectively.

#### B. Basic and affordance features

The classifier trained with *basic-features* uses the following features for training (and prediction later):

$$TS_{\text{basic}} = \{(shape^{o_1}, shape^{o_2}, dim^{o_1}, dim^{o_2})\}$$

where *shape* feature includes high-level curvature information and direction of the hole if it exists; and *dim* encodes the object size in different axes. The classifier trained with *affordance-features* uses the following features:

$$TS_{\text{aff}} = \{(\varepsilon_{s\text{-poke}}^{o_1}, \varepsilon_{f\text{-poke}}^{o_1}, \varepsilon_{t\text{-poke}}^{o_1}, \varepsilon_{s\text{-poke}}^{o_2}, \varepsilon_{f\text{-poke}}^{o_2}, \varepsilon_{t\text{-poke}}^{o_2}, dim^{o_1}, dim^{o_2})\}$$

where  $\varepsilon^o$  refers to the effects of the corresponding poke action on the object  $o$ . Although  $\varepsilon^o$  is manually coded for each object category, we assume that it can be computed from shape features and this computation was learned in previous stages of development.

#### C. Bootstrapping Results

The performances of the classifiers trained with *basic-features* and *affordance-features* are provided in Fig. 3. We evaluated the classifiers by systematically changing the number of categories used in training set. For each number of categories, we trained 10 classifiers by selecting 5 objects of random size from each training category. To test these classifiers, we created test sets with random sized object from the remaining categories. Each bar corresponds to mean performance of these 10 classifiers. As shown, the prediction performance of both *basic-features* and *affordance-features* based classifiers improve by including more categories into the training set. We also included the performance of a category based predictor (which takes category index as input) to show the baseline. Because the categories used in training set are never included into test set, category-based predictors do fail independent of the training set size.

These results show that because *affordance-features* include properties related to object dynamics (pushability, rollability etc), classifiers that use these features have better performance especially for small training sets. With the increasing training set size, the effect of using high-level features is reduced as the system can find the invariance related to stackability affordance with large dataset. Finding this invariance with small datasets is easier with *affordance-features* as they already include some structural properties of the agent-object-environment interactions.

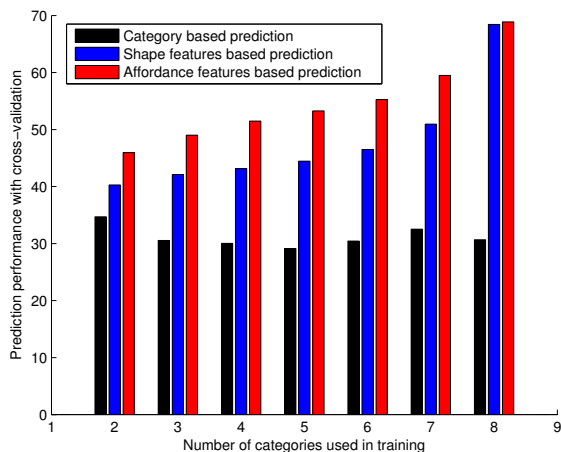


Fig. 3. The effect prediction performance of stack action obtained in artificial interaction database. The training of classifiers are done with the indicated number of categories with either shape features or affordance features. As shown, use of affordance features enable bootstrapping of the learning system.

### IV. BOOTSTRAPPING IN REAL WORLD

#### A. Robot system

The robot system employs a 7 DOF Kuka Light Weight Robot (LWR) arm, which is placed on a vertical bar similar to human arm in Fig. 4. A 7 DOF 3 fingered Schunk gripper is mounted on the arm to enable manipulation. For








								
	$d1 > d2 \rightarrow \text{fail}$ $d1 < d2 \rightarrow \text{stack}$	$d1 > d2 \rightarrow \text{outside}$ $ow. \rightarrow \text{stack}$	$h1 < d2 \rightarrow \text{stack}$ $ow. \rightarrow \text{fail}$	$h1 < d2 \rightarrow \text{stack}$ $ow. \rightarrow \text{fail}$	$w1 < d2 \text{ \& } d1 < d2 \rightarrow$ $\text{stack}$ $\rightarrow \text{fail}$	$w1 < d2 \text{ \& } d1 < d2 \rightarrow$ $\text{stack}$ $w1 > d2 \text{ \& } d1 > d2 \rightarrow$ $\text{outside}$ $ow. \rightarrow \text{fail}$	$d2 > C \rightarrow \text{stack}$ $ow. \rightarrow \text{fail}$	$w1 > d2 \text{ \& } d1 > d2 \rightarrow$ $\text{stack}$ $ow. \rightarrow \text{fail}$
	$d1 > d2 \rightarrow \text{fail}$ $d1 < d2 \rightarrow \text{inside}$	$d1 > d2 \rightarrow \text{outside}$ $d1 < d2 \rightarrow \text{inside}$	$h1 < d2 \rightarrow \text{inside}$ $ow. \rightarrow \text{fail}$	$h1 < d2 \rightarrow \text{inside}$ $ow. \rightarrow \text{fail}$	$w1 < d2 \text{ \& } d1 < d2 \rightarrow$ $\text{inside}$ $ow. \rightarrow \text{fail}$	$w1 < d2 \text{ \& } d1 < d2 \rightarrow$ $\text{inside}$ $w1 > d2 \text{ \& } d1 > d2 \rightarrow$ $\text{outside}$ $ow. \rightarrow \text{fail}$	inside	$w1 > d2 \text{ \& } d1 > d2 \rightarrow$ $\text{inside}$ $ow. \rightarrow \text{fail}$
	fail	$d1 > h2 \text{ \& } d1 > d2 \text{ \& }$ $h1 > d2/2 \rightarrow \text{outside}$ $ow. \rightarrow \text{fail}$	fail	fail	fail	$w1 > h2 \text{ \& } d1 > d2 \text{ \& }$ $h1 > d2/2 \rightarrow \text{outside}$ $ow. \rightarrow \text{fail}$	fail	fail
	fail	$d1 > h2 \text{ \& } d1 > d2 \text{ \& }$ $h1 > d2/2 \rightarrow \text{outside}$ $ow. \rightarrow \text{fail}$	fail	fail	fail	$w1 > d2 \text{ \& } d1 > h2 \text{ \& }$ $h1 > d2/2 \rightarrow \text{outside}$ $ow. \rightarrow \text{fail}$	fail	fail
	$d1 < w2 \text{ \& } d1 < d2$ $\rightarrow \text{stack}$ $ow. \rightarrow \text{fail}$	$d1 < w2 \text{ \& } d1 < d2 \rightarrow$ $\text{stack}$ $d1 > w2 \text{ \& } d1 > d2 \rightarrow$ $\text{outside}$ $ow. \rightarrow \text{fail}$	$h1 < w2 \text{ \& } d2 > C \rightarrow$ $\text{stack}$ $ow. \rightarrow \text{fail}$	$h1 < d2 \text{ \& } w2 > C \rightarrow$ $\text{stack}$ $ow. \rightarrow \text{fail}$	$w1 < w2 \text{ \& } d1 < d2 \rightarrow$ $\text{stack}$ $\rightarrow \text{fail}$	$w1 < w2 \text{ \& } d1 > d2 \rightarrow$ $\text{outside}$ $w1 < w2 \text{ \& } d1 < d2 \rightarrow$ $\text{stack}$ $ow. \rightarrow \text{fail}$	$d2 > C \text{ \& } w2 > C \rightarrow$ $\text{stack}$ $ow. \rightarrow \text{fail}$	$w1 < w2 \text{ \& } d1 < d2 \rightarrow$ $\text{stack}$ $ow. \rightarrow \text{fail}$
	$d1 < w2 \text{ \& } d1 < d2$ $\rightarrow \text{inside}$ $ow. \rightarrow \text{fail}$	$d1 < w2 \text{ \& } d1 < d2 \rightarrow$ $\text{inside}$ $d1 > w2 \text{ \& } d1 > d2 \rightarrow$ $\text{outside}$ $ow. \rightarrow \text{fail}$	$h1 < w2 \rightarrow \text{inside}$ $ow. \rightarrow \text{fail}$	$h1 < d2 \rightarrow \text{inside}$ $ow. \rightarrow \text{fail}$	$w1 < w2 \text{ \& } d1 < d2 \rightarrow$ $\text{inside}$ $ow. \rightarrow \text{fail}$	$w1 < w2 \text{ \& } d1 > d2 \rightarrow$ $\text{outside}$ $w1 < w2 \text{ \& } d1 < d2 \rightarrow$ $\text{inside}$ $ow. \rightarrow \text{fail}$	inside	$w1 < w2 \text{ \& } d1 < d2 \rightarrow$ $\text{inside}$ $ow. \rightarrow \text{fail}$
	fail	$d1 > d2 \text{ \& } h1 > d2/2$ $\rightarrow \text{outside}$ $ow. \rightarrow \text{fail}$	fail	fail	fail	$d1 > d2 \text{ \& } w1 > d2 \text{ \& }$ $h1 > d2/2 \rightarrow \text{outside}$ $ow. \rightarrow \text{fail}$	fail	fail
	fail	$d1 > d2 \rightarrow \text{outside}$ $ow. \rightarrow \text{fail}$	fail	fail	fail	$d1 > d2 \rightarrow \text{outside}$ $ow. \rightarrow \text{fail}$	fail	fail

Fig. 2. The set of rules that are used to create the artificial interaction database for *stack* action. One of the object at the top row is assumed to be dropped on one of the objects in the first column. The effect is determined based on the object categories and the relative sizes of the objects.  $d$ ,  $h$  and  $w$  refers to depth, height and width of the objects.

environment perception, Kinect sensor placed over the torso is used. The objects shown in Fig. 4 are used in learning single-object affordances as well as pairwise-affordances.



Fig. 4. The experiment setup. Kuka LWR robot arm and Schunk gripper are used for manipulation and Kinect is used to extract object features including object's position. The environment includes one and two objects during single-object and multi-object affordance learning experiments, respectively.

1) *Object features*: The robot's workspace consists of several objects and a table where the region of interest is defined as the volume over the table. The objects are segmented

based on depth information. In these experiments an object is represented by a feature vector composed of only shape related features which are encoded as the distribution of local surface normal vectors from object surface<sup>1</sup>. Specifically histograms of normal vectors along each axis, 18 bins each, are computed to form  $3 \times 18 = 54$  sized feature vector.

2) *Robot Actions*: The robot is equipped with a number of manually coded actions that enable single and multi object manipulation. The robot can 'poke' a single object from its side, front and top with *s-poke*, *f-poke*, and *t-poke* actions, respectively. It can also stack one object on the other using *stack* behavior, where it grasps the first object, move it on top of the other one and release it. The object position in world coordinate (shown in Fig. 4) is computed using the depth image of Kinect sensor. An inverse kinematic solver [13] is used to compute the joint angles for initial and final points defined in Cartesian space, and Reflexxes library [14] is utilized to generate smooth trajectories to achieve point-to-point movement. The action execution is as follows:

- Regarding to *poke* actions, the robot gripper is placed on one side of the object with  $5cm$  distance with an appropriate orientation. Two of the fingers are flexed to enable only the third finger to physically interact with

<sup>1</sup>Point Cloud Library normal estimation software is used to compute normal vectors.



the object (similar to index finger poking in humans). Next, the robot hand moves in the corresponding direction for  $10\text{cm}$  towards the object and it is retracted after the poking completed.

- Regarding to *stack* action, one object is grasped from above first by placing the gripper in a vertical orientation  $10\text{cm}$  over of the object, then moving the wide-open gripper towards the object and finally enclosing it. Next, the gripper that carries the grasped object is repositioned over the second object in a vertical orientation again, and the object in the gripper is released over the first one by extending all the fingers.

3) *Effect Categories*: As mentioned before, we limit affordance learning to predicting discrete effect categories in this study. As the robot learns prediction from experience, the robot should have the ability to perceive the discrete effect categories in the end of each interaction. These categories can be detected using unsupervised clustering methods that divide continuous effect feature space as in our previous work [15]. For simplicity in the current implementation, the effect categories are predefined for each action, and directly provided by the experimenter who observes robot’s interactions, i.e. the effects generated by robot’s actions.

Depending on the object(s) and executed action, different effects can be generated:

- When *poke* action is executed, the object might be pushed, toppled over or rolled away depending on the shape. There might be no effect in object state or robot’s sensors if the robot finger goes through the hole in the object without actually touching it. Finally, for *t-poke* action, the object might create resistance and obstruct gripper’s movement that can be detected using the force sensor.
- When *stack* action is executed, the objects can be piled up on top of each other if the object below provides a proper support (for example if it has a flat top surface). Depending on the existence of concave surfaces and holes, the released object might be inserted in or hide (by encapsulating) the object below. The released object may also tumble over due to the lack of stable support. Note that we assume that the first object is always graspable and we focus on the relational effects.

Based on the above possibilities that we observed empirically, the sets of effect categories ( $\mathcal{E}$ ) are as follows for different actions:

$$\mathcal{E}_{poke-*} = \{\text{pushed, rolled, toppled, resisted, no-change}\}$$

$$\mathcal{E}_{stack} = \{\text{piled, inserted-in, covered, tumbled-over}\}$$

## B. Experiment Results

1) *Learning single-object affordances*: The robot executed its poke action on the objects (Fig. 4) placed in different orientations, and it collected 24 interaction instances for each poke action. The object shape features along with generated effect categories are stored for learning

affordances. Support Vector Machine classifiers are used to learn the mapping between object features and effect categories. Here we do not provide an analysis of single-object affordance learning performance as we studied it in detail before [15]. Instead, we show that with the current set of features, the robot can learn and generalize affordances for *poke* actions. For this purpose, we divide the interaction set into training and test sets with the deliberate purpose of distributing objects with same affordances into different sets. The training objects with their orientations are shown in Fig. 5 and the predictions of the classifiers trained with these objects are given in Fig. 6. As shown, the robot was able to detect the affordances of the object (in terms of effect prediction) correctly, except a small number of cases shown with stars (\*). Note that none of the test objects were included in the training set with the tested orientations.

The trained three classifiers (for *s-poke*, *t-poke*, and *f-poke*) are transferred to the next stage and their predictions are used as high-level features to learn complex affordances.



Fig. 5. The robot learned single-object affordances with the training set given above.

2) *Learning pairwise affordances*: In this section, the robot learns the multi-object affordances by exploring the two-object environments with its *stack* action. This learning is again achieved by training an SVM classifier that predicts the effect of the action given object features. Here we compare the prediction performance of the classifiers that are trained either with *basic-features* or *affordance-features*. Regarding to *basic-features*, normal vector histograms are used as we did in learning single-affordances in the previous subsection. Regarding to *affordance-features*, the list of effect predictions (provided by the classifiers transferred from the previous stage) for the poke actions are used.

$$\text{affordance-features} = (\varepsilon_{s-poke}^o, \varepsilon_{f-poke}^o, \varepsilon_{t-poke}^o)$$

where

$$\varepsilon_{*poke}^o = \text{classifier}_{*poke}(\text{shape}^o)$$

The robot executed *stack* action with 18 pairs of random objects. A number of snapshots taken during these interactions are given in Fig. 7, where all the possible effects











Object from robot's view	Explanation	Front-poke prediction	Side-poke prediction	Top-poke prediction
	Lying cylinder	pushed	rolled	resisted
	Lying mug	rolled	pushed	resisted
	Cylinder wall	pushed	pushed	no-change
	Tennis ball	rolled	rolled	resisted
	Lying mug	pushed	rolled	resisted
	Wooden block	pushed	pushed	resisted
	Lying mug	pushed *	rolled	resisted
	Tea box	pushed	pushed	resisted
	Thin wooden block	pushed	toppled ***	resisted
	Thin wooden block	toppled **	pushed	resisted

Fig. 6. Robot's basic-affordance prediction on objects which are not included in the training set with the same orientations. Prediction fails in the examples with star (\*), which are difficult cases to predict.

were observed with different object pairs. In each interaction, *basic-features* and *affordance-features* of both objects are computed and stored along with the observed effect category.

The classifier trained with *basic-features* uses the following features for training (and prediction later):

$$TS_{\text{basic}} = \{(shape^{o_1}, shape^{o_2})\}$$

and the classifier trained with *affordance-features* uses the following features:

$$TS_{\text{aff}} = \{(\epsilon_{s-poke}^{o_1}, \epsilon_{f-poke}^{o_1}, \epsilon_{t-poke}^{o_1}, \epsilon_{s-poke}^{o_2}, \epsilon_{f-poke}^{o_2}, \epsilon_{t-poke}^{o_2})\}$$

where  $\{\}$  corresponds to the set operator so  $|TS|$  is the size of training set.

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. Fig. 8 gives these cross-validation results. When large training sets are used, both *basic-features* and *affordance-features* based classifiers have similar prediction performances. However, with small number of training samples, the *affordance-features* based classifiers have better performance.

The initial high performance of *affordance-features* based classifiers demonstrates the advantage of using bootstrapping in learning affordances.

## V. FUTURE WORK

In this study, we showed that learned basic affordances can be used as additional features in order to bootstrap the next stage of affordance learning. This bootstrapping enabled the robot to speed up its learning particularly with small training data. While this work serves as one of the proof-of-concept

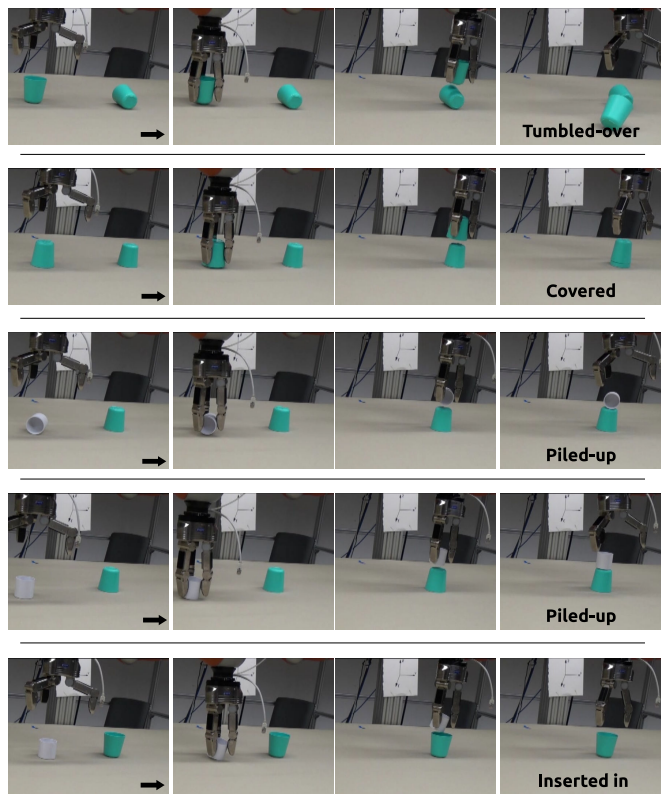


Fig. 7. Sample interactions observed during *stack* action execution on different object pairs.

application of the structural bootstrapping idea [16], we need to adapt advanced representations and learning methods that can truly exhibit the real potential of this idea.

Particularly, knowledge propagation framework of [17] suits well to learning affordance relations in our developmental scenario. This learning framework can deal with sparse, incomplete and complex relations where affordance predictions can be propagated through exploiting the similarities among object properties, action parameters and generated effects. The high complexity of affordance representation can be addressed through use of Maximum Margin Multi Valued Regression which is a large scale approach to complex problems of several layers. With increased variety and size of object database and addition of other parametric combinatory actions, we expect to achieve complex systems which truly uses 'structural' bootstrapping in its lifelong learning and development.

## 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 agreement no. 270273, Xperience.

## REFERENCES

- [1] B. Vollmer and H. Forssberg, "Development of grasping and object manipulation," in *Sensorimotor Control of Grasping: Physiology and Pathophysiology*. Cambridge University Press, 2009.

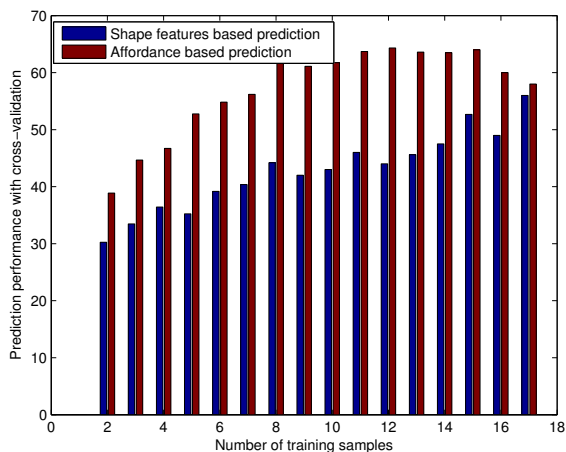


Fig. 8. The effect prediction performance of stack action that involves two objects. The training of classifiers are done with the indicated number of samples (interactions) with either shape features or affordance features. As shown, use of affordance features enable bootstrapping of the learning system. Note that affordances features are computed from shape features with classifiers trained before.

ater, A. Ude, N. Kuger, and M. Steedman, "Structural bootstrapping – a novel concept for the fast acquisition of action-knowledge," 2014, in preparation.

- [17] S. Szedmak and J. Piater, "An active learning based sampling design for structural bootstrapping," University of Innsbruck, Tech. Rep., 2014.

[2] M. Asada, K. Hosoda, Y. Kuniyoshi, H. Ishiguro, T. Inui, Y. Yoshikawa, M. Ogino, and C. Yoshida, "Cognitive developmental robotics: a survey," *IEEE Tran. Auton. Mental Dev.*, vol. 1-1, 2009.

[3] H. Takeshita, "Development of combinatory manipulation in chimpanzee infants (pan troglodytes)," *Animal Cognition*, vol. 4, pp. 335–345, 2001.

[4] M. Ikuzawa, *Development diagnostic tests for children*, 2000.

[5] M. Hayashi and T. Matsuzawa, "Cognitive development in object manipulation by infant chimpanzees," *Animal Cognition*, vol. 6, pp. 225–233, 2003.

[6] W. King and B. Seegmiller, "Performance of 14- to 22-month old black, firstborn male infants on two tests of cognitive development," *Developmental Psychology*, vol. 8, pp. 317–326, 1973.

[7] L. Gleitman, "The structural sources of verb meanings," *Language acquisition*, vol. 1, pp. 3–55.

[8] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *Proceedings of the 7th IEEE International Conference on Development and Learning*. IEEE, Aug. 2008, pp. 91–96.

[9] P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 4373–4378.

[10] E. Şahin, M. Çakmak, M. R. Doğar, E. Uğur, and G. Üçoluk, "To afford or not to afford: A new formalization of affordances towards affordance-based robot control," *Adaptive Behavior*, 2007, (in press).

[11] N. Kruger, J. Piater, F. Worgotter, C. Geib, R. Petrick, M. Steedman, A. Ude, T. Asfour, D. Kraft, D. Omrcen, B. Hommel, A. Agostino, D. Kragic, J. Eklundh, V. Kruger, and R. Dillmann, "Object-action complexes: Grounded abstractions of sensorimotor processes," *Robotics and Autonomous Systems*, vol. 59, pp. 740–757, 2011.

[12] R. Detry, E. Başeski, M. Popović, Y. Touati, N. Krüger, O. Kroemer, J. Peters, and J. Piater, "Learning continuous grasp affordances by sensorimotor exploration," in *From Motor Learning to Interaction Learning in Robots*, ser. Studies in Computational Intelligence. Springer Berlin / Heidelberg, 2010, vol. 264, pp. 451–465.

[13] B. Moore and E. Oztop, "Redundancy parametrization for flexible motion control," in *ASME IDETC*, 2010.

[14] T. Kroger, "Opening the door to new sensor-based robot applications – the reflexes motion libraries," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 1–4.

[15] 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.

[16] F. Worgotter, C. Geibb, M. Tamosiunaite, E. E. Aksoy, T. Asfour, J. Pi-



This is the title page and the abstract of a recent IEEE TAMM submission. The full version of this is identical to Chapter 2 of this Deliverable.

# Structural bootstrapping - a novel concept for the fast acquisition of action-knowledge

Florentin Wörgötter<sup>a,i</sup>, Chris Geib<sup>b,c,i</sup>, Miniya Tamosiunaite<sup>a,d,i</sup>, Eren Erdal Aksoy<sup>a</sup>, Justus Piater<sup>e</sup>, Hanchen Xiong<sup>e</sup>, Ales Ude<sup>f</sup>, Bojan Nemeč<sup>f</sup>, Dirk Kraft<sup>g</sup>, Norbert Krüger<sup>g</sup>, Mirko Wächter<sup>h</sup>, Tamim Asfour<sup>h</sup>

<sup>a</sup>*Georg-August-Universität Göttingen, Bernstein Center for Computational Neuroscience, Department for Computational Neuroscience, III Physikalisches Institut - Biophysik, Göttingen, Germany*

<sup>b</sup>*School of Informatics, Edinburgh, United Kingdom*

<sup>c</sup>*College of Computing and Informatics, Drexel University, Philadelphia, USA*

<sup>d</sup>*Department of Informatics, Vytautas Magnus University, Kaunas, Lithuania*

<sup>e</sup>*Institute of Computer Science, University of Innsbruck, Innsbruck, Austria*

<sup>f</sup>*Humanoid and Cognitive Robotics Lab, Dept. of Automatics, Biocybernetics, and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia*

<sup>g</sup>*Cognitive and Applied Robotics Group, University of Southern Denmark, Odense, Denmark*

<sup>h</sup>*Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany*

<sup>i</sup>*These authors have contributed equally to this work.*

---

## Abstract

Soon after birth children begin to use their first knowledge bits to quickly assimilate novelty in a generative way getting more and more efficient at this with age. New information is bootstrapped by prior experience that allows the child to perform *guided guesses* about novel observations. Different from this, up to now learning complex tasks by a robot remains a tedious and time-consuming undertaking. Even for the simplest actions, many components (e.g. planning sequences, object knowledge, motor control information, etc.) need to be captured, processed, and stored. Furthermore, all this information needs to be processed in a way that allows the agent to perform the same action also in different situations. Thus, efficient, human-like generative knowledge acquisition has not yet been achieved for artificial agents. The goal of the current study is to devise for the first time a general framework for such a generative process across the different complexity levels that exist for action knowledge acquisition. To this end, we introduce the concept of structural

bootstrapping – borrowed and modified from child language acquisition – to define a probabilistic process that uses existing knowledge together with new observations to supplement our robot’s data-base by missing planning, object, as well as action information. In a kitchen scenario, we use the example of making batter by pouring and mixing two components and show that the agent can acquire new knowledge about planning operators, objects as well as required motor pattern for stirring by structural bootstrapping. Some benchmarks are shown, too, that demonstrate the substantial speeding-up of the learning.

*Keywords:* Generative Model, Knowledge Acquisition, Fast Learning

---

# Homogeneity Analysis for Object-Action Relation Reasoning in Kitchen Scenarios \*

Hanchen Xiong Sandor Szedmak Justus Piater  
Institute of Computer Science, University of Innsbruck  
Technikerstr.21a A-6020, Innsbruck, Austria  
{hanchen.xiong,sandor.szedmak,justus.piater}@uibk.ac.at

## ABSTRACT

Modeling and learning object-action relations has been an active topic of robotic study since it can enable an agent to discover manipulation knowledge from empirical data, based on which, for instance, the effects of different actions on an unseen object can be inferred in a data-driven way. This paper introduces a novel object-action relational model, in which objects are represented in a multi-layer, action-oriented space, and actions are represented in an object-oriented space. Model learning is based on homogeneity analysis, with extra dependency learning and decomposition of unique object scores into different action layers. The model is evaluated on a dataset of objects and actions in a kitchen scenario, and the experimental results illustrate that the proposed model yields semantically reasonable interpretation of object-action relations. The learned object-action relation model is also tested in various practical tasks (e.g. action effect prediction, object selection), and it displays high accuracy and robustness to noise and missing data.

## 1. INTRODUCTION

Manipulations of objects are core and indispensable functions in robotic systems to fulfill various practical tasks. However, because of the diversity of real-world objects in shape, material and other properties, manipulation design at the instance level is very effort-consuming and thus prohibitive. Learning principles or correlation patterns of different actions based on trial experiences is an appealing direction of robotics research. In other words, an agent can acquire knowledge of object-action relations in a data-driven manner by making use of a limited number of experiments.

\*The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience.

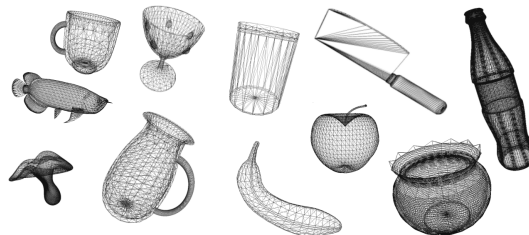


Figure 1: A sample set of kitchen objects

In addition, the study of object-action relations has also attracted attention within the cognition and psychology communities [5, 10], since it is expected to be related to how human beings accumulate knowledge by physically interacting with different objects. Humans begin to interact with their environment in their infancy, and in many interactions, two elements are involved: objects and actions. Actions are executed on objects with the humans' motor capabilities, and the effects of these actions are observed with their perception abilities. Based on such repeated interactions, human beings can quickly acquire object-action knowledge, and easily fulfill different actions on various objects by transferring such knowledge to novel objects. Although the exact mechanism of how the human brain organizes and learns object-action relations is still unknown, it has been pointed out that computational modeling of object-action relations can be a plausible perspective for the study of both robotics and human cognition.

Nevertheless, modeling and learning object-action relations has been a difficult task. The difficulties mainly stem from two sources. First, the structure of descriptions of both objects and actions can be very complex. The descriptions are derived from several sources, and the corresponding feature spaces are high-dimensional (i.e., objects and actions are characterized by large numbers of parameters). The second difficulty is due to the small number of experiments which can confirm the effects of different actions on objects. Even worse, in some cases, the experiments might provide contradicting outcomes. In consequence, the empirical data are rather sparse and noisy.

In this paper we put forward a novel model of object-action relations, in which objects are represented in a multi-layer action-oriented space, and actions are represented in an object-oriented space. The object-action relations are encoded in

these two spaces, on which various reasoning tasks can be performed. The training data for the model are constructed from two sources, objects and the collection of effects (positive and negative) of different actions executed on objects. Two pieces of information are summarized in a structure called object-action profiles. Objects are represented by categorical indicators of basic properties and binary labels of various low- and high-level geometric features. Actions are represented by binary labels of object-dependent effects. The model is learned with *homogeneity analysis*. The strength of homogeneity analysis is that it can map multi-variate categorical/binary data to a homogeneous Euclidean space. Via these projections, objects and actions can be effectively represented with object scores and category quantifications. Basically, the object scores are computed as the average of quantifications of categories which they belong to, and category quantifications are computed as the geometric centroid of objects they belong to. These two projections are iteratively updated until convergence. Based on homogeneity analysis, action-category quantifications are represented in an object-oriented manner. The resulting object scores, however, do not fit our modeling scenario. The object scores are computed by treating all variables of object-action profiles equivalently, and therefore the scores are unique for all actions. By contrast, our model is designed to represent objects differently with respect to different actions. Therefore, we provide dedicated means to determine the dependencies between category quantifications of object and action variables, and decompose the object scores/representations into different action layers.

We present our model and associated learning/reasoning procedures in the context of object-action relations within a kitchen scenario (Figure 1). A database of typical kitchen objects and actions is constructed as well to evaluate our model. The experimental results demonstrate that the model yields semantically good interpretation of object-action relations by displaying reasonable dependencies and correlations between object and actions variables. In addition, the experiment with sparseness and noise added into training data highlights the robustness of our model to noisy and missing data.

## 1.1 Related Work

For object manipulation knowledge modelling, the concept of *affordance* [5] has been widely used [9, 8, 10, 3, 4] to link objects and actions in terms of object-action-effect triples. An affordance defines how an object “affords” a manipulation by an agent based on its motor abilities, and how this manipulability can be perceived by the agent [5]. For instance, the grasping affordance of a stone is much higher for a human being than a dog since human hands have better motor control of fingers than dogs’ paws. More concretely, object affordances represent how can an agent interact with real-world environment by encoding the relations among actions, objects and effects. Although there have been numerous studies on how affordances can be modelled such that they can be effectively learned and utilized to assist practical robotic manipulation, the object/action affordance problem, at its base, is about how an agent can understand objects based on interactions with them by using its motor and perceptual capabilities. However, most previous studies are limited to one isolated object affordance (e.g. grasping). In

some cases, multiple objects are involved and interact with each other within one manipulation. For example, a single action such as cutting involves two objects, the cutting tool (e.g. knives) and the object being cut (e.g. an apple). In [7], the affordance definition was extended to object relations. However, since only geometric relation (distance, angle of orientations) between multiple objects are used in [7], it still cannot model concepts such as cutting affordances for objects. Our work, by contrast, seeks to model general object-action relations. Our relational model connects objects and all possible actions that can be performed on them.

Our model is mainly inspired by [3], of which the basic assumption is that objects that share similar parts (e.g. rim, handles) should also hold similar grasping affordances. We extend [3] in two ways: first, we consider general object-action relations instead of only grasping affordances; second, the dependency of actions on different parts can be learned, in which way, for different actions, different co-occurring parts among objects will be considered for their action-effect reasoning.

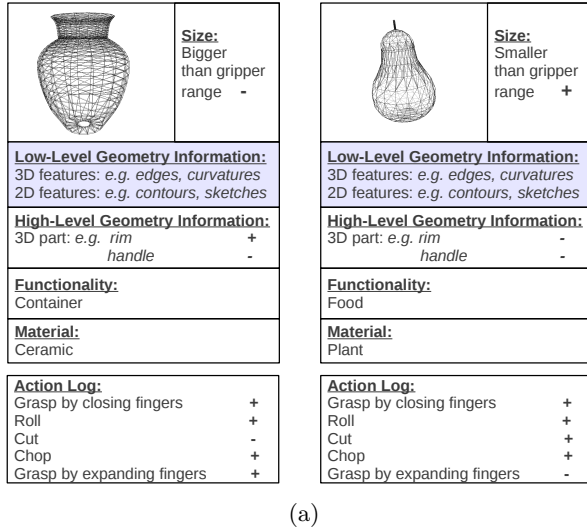
Other related work involves modeling of *sensorimotor coordination* [9], where a Bayesian network is employed to model multiple affordances associated with objects based on visual properties (e.g. color, size, concavity) and basic motor actions (grasping, touching, tapping). The dependencies between actions, perception and effects are encoded in the directed edges within the Bayesian network. One shortcoming of this model is the dependency learning (i.e. the Bayesian network structure). Since in a Bayesian framework it is impractical to estimate the likelihoods of all possible dependency structures, Markov chain Monte Carlo (MCMC) sampling was used to approximate them. However, one practical problem with MCMC is that it can be quite inefficient (usually multiple chains are necessary); secondly, the approximation can be misleading when the training data is small in size, noisy and incomplete. By contrast, the dependency learning of our model is based on the category quantifications from homogeneity analysis, which is robust to noisy and missing data.

## 2. MODELING

In this section, two basic elements are explained for object-action relation modeling. First, we introduce a new data structure constructed from empirical object and action data (section 2.1). Secondly, section 2.2 presents an overview of the model structure (Figure 3), in which objects are represented in a multi-layered action-oriented space, and actions are likewise represented in an object-oriented space.

### 2.1 Data Structure

Since our objective is to learn the relations between objects and actions, the training data is constructed from two sources. One is the *object dataset*, in which basic properties (e.g. size, functionality, material) are labelled, and various low- and high-level geometric properties can be extracted by visual perception. The other source is the *action dataset* that collects the effect of different actions applied on objects. In our study, the information from these two sources is merged into *object-action profiles*. Figure 2(a) presents two examples of object-action profiles. In the upper part, object shape is displayed. Some basic properties are labelled



O	Mesh	<Gripper	L_Geo		H_Geo		Func	Mate	Action log				
			3D	2D	rim	handle			Grasp_C	Roll	Cut	Chop	Grasp_E
1	file1	1			1	-1	1	1	1	-1	*	1	-1
2	file2	-1			-1	*	2	2	-1	*	1	1	*
3	file3	-1			1	1	2	5	*	1	*	1	1
4	file4	1			1	1	5	3	1	1	-1	*	1
5	file5	1			-1	-1	1	4	*	1		1	-1
6	file6	-1			1	1	4	6	1	-1	*	-1	1

Functionality	Container	Food	Cooker	Cutting tool	Eating tool
	1	2	3	4	5

Material	Plastic	Glass	Ceramic	Plant	Animal	Metal
	1	2	3	4	5	6

Figure 2: (a) Two examples of object-action profiles. (b) Collection of object-action profiles, \* denoting missing data. Incompleteness (or sparseness) will always be a problem in training data.

and geometric features are extracted. Because we are only concerned with the kitchen scenario, functionalities are limited to {container, food, cooker, cutting tool, eating tool}, and materials are limited to {plastic, glass, wood, plant, animal, metal}. For size, a binary indicator is used to check if it is smaller than the gripper’s maximum range. In addition, low-level and high-level geometric features of objects can be detected or labelled (although we currently only use high-level geometric features such as rim, handle<sup>1</sup>, because they are more informative of our actions than low-level features). In the lower part, the resulting effects of different actions on the object are recorded with binary values (+1 means successful and -1 otherwise). We consider some more-or-less common kitchen actions {grasping by closing fingers, rolling, cutting, grasping by expanding fingers, chopping}. It is worth noting that the strategies of feature labelling and action selection used in this paper are just one among many ways of describing the proposed model (section 2.2) and learning/reasoning procedure (section 3); they can be replaced by equivalent or more elaborate mechanisms. It should also be noted that in practice a very limited number of action experiments or simulations can be conducted on only a few objects, so incompleteness (or sparseness) of experimental data is a fact we have to deal with (Figure 2(b)).

## 2.2 Model Structure

In this paper, the object-action relations are modeled as shown in Figure 3. Actions and objects are represented in different spaces, that is, *action space* and *object space* respectively. The object space is composed of different layers that correspond to different actions. In each layer of the object space, the objects are linked pairwise (Figure 3), and the connection between a pair of objects is weighted proportionally to their similarity with respect to the corresponding

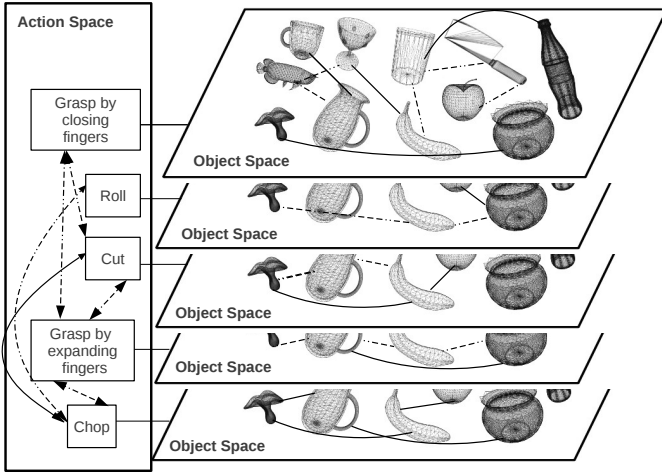
<sup>1</sup>Such labels can be obtained by straightforward shape analysis systems.

action. The similarities between objects can be measured based on co-occurring properties or geometric features that can influence the outcome of the action. For instance, if object *A* (mug) and *B* (goblet) are both containers (therefore exhibit hollow structure), their similarities will be high in the “Grasp by expanding fingers” layer. However, their similarity would be low in the “roll” layer since *A* has a handle but *B* does not, and having a handle or not is a decisive factor for rolling.

In action space there is only one layer. Different actions are connected with each other, likewise with the connections weighted proportionally to their similarities. The similarities between actions can be interpreted as the similarities between their corresponding layers in object space.

## 3. MODEL LEARNING AND REASONING

With training data organized in the form of Figure 2(b), we straightforwardly apply homogeneity analysis [2, 6] to project all columns of Figure 2(b) to category quantifications and rows to object scores (section 3.1). However, the object scores computed by homogeneity analysis are the same for all actions, which does not fit our multi-layer object space (section 2.2). The underlying principle of our multi-layer object representations is that the dependencies between every action and object properties and geometric features are different; therefore, objects should be represented differently with respect to different actions. Meanwhile, the dependency and correlation relations between different basic properties, geometric features and actions are usually complicated. Two examples of such dependencies can be seen in Figure 4. It can be easily imagined that if a container is smaller than the gripper range in size, then it probably can be grasped by expanding fingers, so there should be dependencies on “Container” and “<Gripper” for action “Grasp by expanding fingers”(Figure 4(a)). At the same time, containers smaller than the gripper are often made of ceramic or



**Figure 3: Object-action relational model.** The object space is composed of action-specific layers, in which objects are interconnected (solid lines denote strong and dashed lines weak connections). There is only one layer in action space, and actions are connected in a similar way.

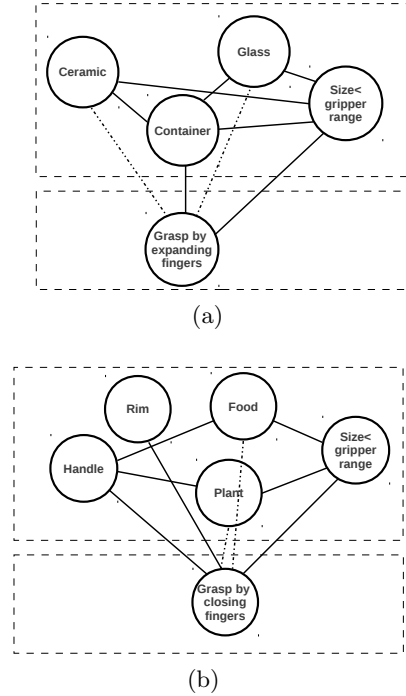
glass (e.g. bowls, mug, wineglass) in contrast to larger objects (e.g. plastic buckets or metal trash cans), so “Grasp by expanding fingers” might also be correlated with “Ceramic” and “Glass” (dashed lines). Similarly, usually an object is graspable if it is smaller than the gripper size or if it has a handle or rim, so it is reasonable to add dependencies between them (Figure 4(b)). Food items and plants are usually smaller than the gripper in a kitchen scenario, and they are unlikely to have handles. So extra dependencies on “Food” and “Plant” may be added as well. Instead of tediously reasoning about the dependencies for all actions, in section 3.2 a dependency checking mechanism is provided to remove unlikely or weak dependencies. The computed dependencies are also utilized to remap objects to different action layers with dependency weights.

### 3.1 Initial Learning with Homogeneity Analysis

Homogeneity analysis [2, 6] is a popular statistical tool for categorical multivariate analysis. Here we briefly review the procedure of homogeneity analysis with its application to object-action profile data. There are  $M$  object-action profiles in the dataset, each profile represented by a  $J$ -dimensional vector  $O_i = [v_1, v_2, \dots, v_J]^T$  ( $i = 1, \dots, M$ ), with each variable  $v_j$  denoting an attribute in the profile. Variable  $v_j$  takes on  $n_j$  categorical values (e.g., the action effect has binary values  $\pm 1$ ). By gathering the values of  $v_j$  over all  $M$  profiles in an  $M \times n_j$  binary indicator matrix  $G_j$ , the whole set of indicator matrices can be gathered in a block matrix:

$$G = [G_1 | G_2 | \dots | G_J] \quad (1)$$

The key feature of homogeneity analysis is that it simultaneously produces two projections to the same Euclidean space  $\mathbb{R}^p$ , one from  $J$ -dimensional profiles  $O_i$ , the other from the  $M$ -dimensional categorical attribute indicator vectors



**Figure 4: Two examples of dependencies between actions and objects’ basic properties and geometry features: (a) grasp by expanding fingers; (b) grasp by closing fingers.**

(columns of  $G$ ). These projections are referred to as *object score* and *category quantification*, respectively [2, 6]. Suppose the collection of object scores is represented by an  $M \times p$  matrix  $X$ , and category quantifications for variable  $v_j$  are represented by a  $n_j \times p$  matrix  $Y_j$ . Then, the cost function of a projection can be formulated as:

$$f(X, Y_1, \dots, Y_J) = \frac{1}{J} \sum_{j=1}^J \text{tr}(X - G_j Y_j)^\top (X - G_j Y_j) \quad (2)$$

As emphasized above, in realistic cases the training dataset is usually sparse and incomplete, i.e., values of some  $v_j$  are missing. So for each  $G_j$ , we construct an  $M \times M$  diagonal matrix  $S_j$  with diagonal values equal the sum of the rows of  $G_j$ , i.e.,  $S_j(i, i) = 0$  if the  $v_j$  value of  $O_i$  is missing. Then the corresponding cost function is

$$f(X, Y_1, \dots, Y_J) = \frac{1}{J} \sum_{j=1}^J \text{tr}(X - G_j Y_j)^\top S_j (X - G_j Y_j) \quad (3)$$

Usually two extra constraints are added to avoid trivial solution ( $X = 0, Y_j = 0$ ):

$$\frac{1}{M} \mathbf{1}_{M \times 1}^\top S_* X = \mathbf{0} \quad (4)$$

$$\frac{1}{M} X^\top S_* X = I \quad (5)$$

Here,  $S_* = \sum_{j=1}^J S_j$ . The first constraint (4) essentially normalizes the projected object scores to be centered around the origin. The second restriction (5) standardizes all  $p$  dimensions of object score by rescaling the square length of each



dimension to  $M$ . In addition, another effect of (5) is that the  $p$  columns of  $X$  are imposed to be orthogonal to each other.

To minimize the cost function (3) under these constraints (4, 5), usually the alternating least squares (ALS) algorithm [2, 6] is used. The basic idea of ALS is to iteratively optimize with respect to  $X$  or to  $[Y_1, \dots, Y_M]$  with the other held fixed. Assuming  $X^{(0)}$  is provided arbitrarily at iteration  $t = 0$ , each iteration of ALS can be summarized as:

1. update  $Y_j$ :

$$Y_j^{(t)} = (G_j^\top S_j G_j)^{-1} G_j^\top X^{(t)}; \quad (6)$$

2. update  $X$ :

$$X^{(t+1)} = S_*^{-1} \sum_{j=1}^J G_j Y_j^t; \quad (7)$$

3. normalize  $X$ :

$$X^{(t+1)} = \text{Gram-Schmidt}(X^{(t+1)}). \quad (8)$$

It can be seen (6) that category quantification of  $Y_j$  is computed as the centroid of the object scores that belong to it. Step 2 (7) updates object scores  $X$  by taking the average of the quantifications of the categories it belongs to. In step 3 (8) a *Gram-Schmidt* procedure is used to find the normalized and orthogonal basis of updated object scores from the previous step.

### 3.2 Dependency Learning

According to the description in the previous section, the objects and action effects can be projected into two spaces (object scores  $X$  and category quantifications  $Y_j$  of action variables  $v_j$ ) by applying homogeneity analysis on the set of object-action profiles. Although this observation is close to how we model object-action relations (section 2.2), there still exist some obstacles that prevent us from directly putting them to practical use. First, by using homogeneity analysis, basic properties, geometric features and action effects are simultaneously projected to their corresponding category quantifications without modelling their interrelations explicitly. As we illustrated in Figure 4, the dependency between them is an important factor in our object-action relational model, so we must disentangle how each action depends on different basic properties and geometric features. Secondly, in our model the objects are represented at different layers corresponding to different actions, while the representations of objects with homogeneity analysis are unique object scores. Hence, it is also required to strategically decompose the object scores into different action layers.

To resolve these two problems, some extra steps can be developed to exploit more information from the object scores and category quantifications. First, the  $J$  variables  $[v_1, v_2, \dots, v_J]$  of each object  $O_i$  are divided into two groups, the object (variable) group  $V_o$  which covers basic properties and geometry features, and the action (variable) group  $V_a$  which contains action effects on the object  $O_i$ . We initially assume that each variable in action group  $v_\beta^a \in V_a$  depends

on all variables of the object group  $V_o$ . Then, for variable  $v_\beta^a$ , we find its corresponding positive and negative category quantifications  $Y_{\beta,+}^a$  and  $Y_{\beta,-}^a$ , and compute the distances between them and all categories' quantifications in the object group as

$$d(Y_{\beta,+}^a, Y_{\omega,k}^o) = \|Y_{\beta,+}^a - Y_{\omega,k}^o\|_2 \quad (9)$$

where  $Y_{k,w}^o$  denotes the  $k$ -th category quantification of variable  $v_\omega^o$  in the object group. We compute the maximum ratio between them as

$$\lambda_{\omega,k}^\beta = \max \left\{ \frac{d(Y_{\beta,+}^a, Y_{\omega,k}^o)}{d(Y_{\beta,-}^a, Y_{\omega,k}^o)}, \frac{d(Y_{\beta,-}^a, Y_{\omega,k}^o)}{d(Y_{\beta,+}^a, Y_{\omega,k}^o)} \right\} \quad (10)$$

and eliminate the dependencies between action variable  $v_\beta^a$  and category quantifications in  $V_o$  if

$$\frac{\lambda_{\omega,k}^\beta}{\sum_{\omega,k} \lambda_{\omega,k}^\beta} < \sigma \quad (11)$$

where  $\sigma \in [0, 1]$  is a predefined threshold. The elimination criterion (11) is defined based on the concept that the object variables on which the action variable depends should have good discriminative abilities between its positive and negative categories.

Once the dependencies have been found, the second problem can be solved as well. Instead of computing object scores as the average of the all quantifications of the categories they belong to (7), the representations of objects in each action layer  $\beta$  are computed as the weighted average of quantifications of the (positive and negative) action categories and the category quantifications in  $V_o$  which the action is dependent on:

$$X_\beta = \hat{S}_{*,\omega,k}^{-1} \sum_{\omega,k \in \text{dependent}(\beta)} \pi_{\omega,k} \hat{G}_{\omega,k} \hat{Y}_{\omega,k} \quad (12)$$

where the  $\hat{Y}_{\omega,k}$  are the category quantifications (out of  $n_\omega$ ) of variable  $v_\omega^o$  on which action variable  $v_\beta^a$  depends.  $\hat{G}_{\omega,k}$ ,  $\hat{S}_*$  are the corresponding indicator matrix and diagonal matrix.  $\pi_{\omega,k}$  denotes the normalized dependency weights which reflect how  $\beta$  depends on quantifications in  $\hat{Y}_{\omega,k}$ :

$$\pi_{\omega,k} = \frac{\lambda_{\omega,k}^\beta}{\sum_{\omega,k \in \text{dependent}(\beta)} \lambda_{\omega,k}^\beta} \quad (13)$$

Correspondingly, the centroid of object representations which belongs to positive and negative category in  $\beta$  action layer is:

$$\beta_c^{+/-} = (G_{\beta,+/-}^\top S_{\beta,+/-} G_{\beta,+/-})^{-1} G_{\beta,+/-}^\top X_\beta \quad (14)$$

where  $G_{\beta,+/-}$  is the positive/negative-category column in  $G_\beta$  and  $S_{\beta,E}$  is corresponding diagonal counting matrix.

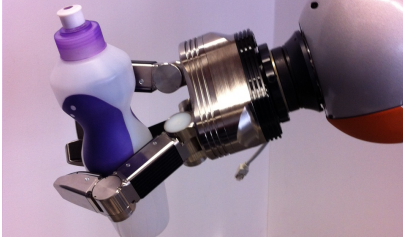
The dependencies between action variables can be also similarly learned to find the correlation or anti-correlation between object effects. Since our model is dedicated to relations between objects and actions, action-action relations will be exploited in our future work.

### 3.3 Reasoning

Given the object-action relational model learned with the procedure above, typical reasoning tasks are presented in

input	output	applications
object & action	effect	effect outcome prediction
action & effect	object	object selection
object & effect	action	action planing/recognition

**Table 1: Typical applications of the object-action relation model.**



**Figure 5: Robot hand used for action labelling**

Table 1. First, we discuss effect ( $E$ ) prediction given object ( $O$ ) and action ( $\beta$ ). Assume  $O$  is an unseen object. Its representation in action layer  $\beta$  can be computed (12), and then the binary effect classification can be easily done by majority voting of the  $k$ -nearest neighbouring objects of training set (or using any other suitable classifier).

Second, the model can perform object ( $O$ ) selection out of a set of candidates  $\mathbf{C}$  based on action ( $\beta$ ) and effect ( $E \in [-1, 1]$ ). Given the desired category  $E$  of action  $\beta$ , first object representations in candidate set  $X_{\beta}^{(O \in \mathbf{C})}$  can be computed (12). Then the ratio of the distance between each  $X_{\beta}^{(O)}$  and  $\beta_c^E$  to the distance between  $X_{\beta}^{(O)}$  and  $\beta_c^{-E}$  (14) can be computed:

$$\phi_O = \frac{d(X_{\beta}^{(O)}, \beta_c^E)}{d(X_{\beta}^{(O)}, \beta_c^{-E})} \quad (15)$$

The optimal object  $O^\dagger$  is the one with smallest  $\phi_O$ . Alternatively, with the ratios of all objects in  $\mathbf{C}$  computed, the object retrieval result can be ranked by their ratios in increasing order.

Finally, action selection or planning is also useful to find an optimal action among many that share similar semantic effects based on certain criteria. For example, both cutting and chopping are actions that break objects into smaller parts. However, they are executed with different tools (cleavers for chopping and knives for cutting) and with different strength. So if the task is to break an object  $O$  into parts with minimum strength from the higher-level planner, then one may want to perform a chopping action only if necessary. To this end, we compute the representation of  $O$  in cutting and chopping layers respectively and predict their corresponding effects, based on which the most energy-saving action will be selected.

## 4. EXPERIMENTS

### 4.1 Synthetic Database and Model Learning

To evaluate the proposed object-action relational model and learning method, we constructed a synthetic dataset of object-action profiles. We collected 140 kitchen objects (Figure 1)

from the web [1] and annotated them as shown in Figure 2. The labeling and actions are set in the same way as described in section 2.1. Basic properties and high-level geometry features<sup>2</sup> of objects were labelled by a student volunteer. The effects of different actions applied on objects are labeled as well based on common sense<sup>3</sup>. The robot gripper is presented to the labeller (Figure 5) for the consideration of different actions.

First, the model is learned with full and noisy-free data. By applying homogeneity analysis as described in section 3.1, we obtain 3-dimensional category quantifications of 10 variables in object-action profiles (Figure 6). With extra maximum ratio computation (10) (Figure 7), the dependency between each action and objects’ basic properties and geometric features are discovered (Table 2). Table 2 shows that “grasp by expanding fingers” and “grasp by closing fingers” exactly match our previous dependency analysis in Figure 4, i.e. the proposed model yields semantically reasonable object-action relations.

### 4.2 Reasoning Tasks

To quantitatively evaluate the proposed model, the following experiments test the model on two reasoning tasks, effect prediction and object selection<sup>4</sup>. In both experiments, the 140 object-action profiles are randomly divided into training set (100) and test set (40). In addition, as we already pointed out, in practice the empirical object-action data can be noisy and incomplete because of inaccuracy of perception systems and lack of real (or simulated) experiments. Therefore, to test the robustness of the model to noise and missing data, 10% noise are added and 20% entries are removed from the 100 training instances. The noise is generated by shifting the labels of variables with probability 0.1, and entries in Figure 2 are removed with probability 0.2.

#### Effect Prediction

According to the reasoning procedure described in section 3.3, 40 test objects are first projected to different representations at different action layers. Then the final effects of actions are decided by using a simple  $k$ -nearest-neighbour (KNN) classifier with the 100 representations of training objects. We use  $k = 10$  for both full-data and missing-and-noisy-data conditions. We ran 50 trials in which different size-100 training (both full and missing-and-noisy) and size-40 test data sets are randomly generated. The average precision of correct effect classification of five actions are presented in Figure 8(a), from which it can be seen that the prediction results with both full training data and missing-and-noisy data are rather accurate, with the former slightly outperforming the latter (as is to be expected).

#### Object Selection

The object selection experiment is set up to test how accurate an object can be “recommended” to meet the effect of an action. The reasoning is based on the procedure in

<sup>2</sup>We did not use low-level geometric features in our experiments.

<sup>3</sup>In future work, we plan to use simulated and ultimately physical robotic action.

<sup>4</sup>Since action selection applications usually require higher-level planners to handle constraints, robustness criteria etc., we did not consider them in our pilot experiments.



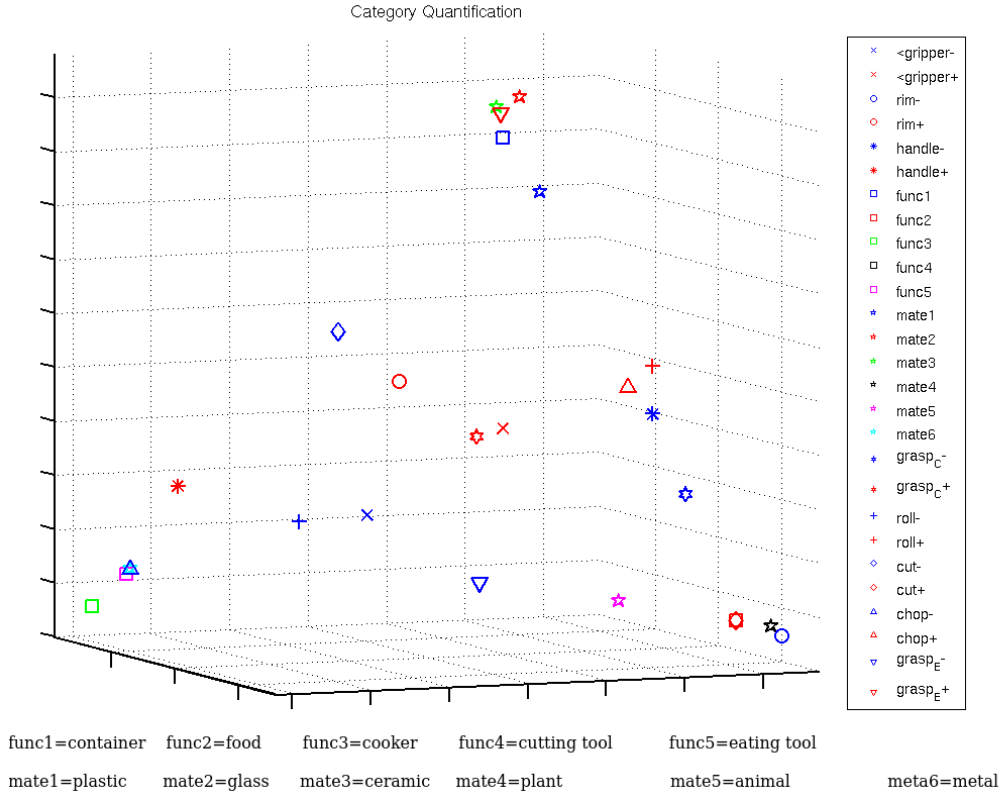


Figure 6: Category quantifications of variables in object-action profiles (best viewed in color).

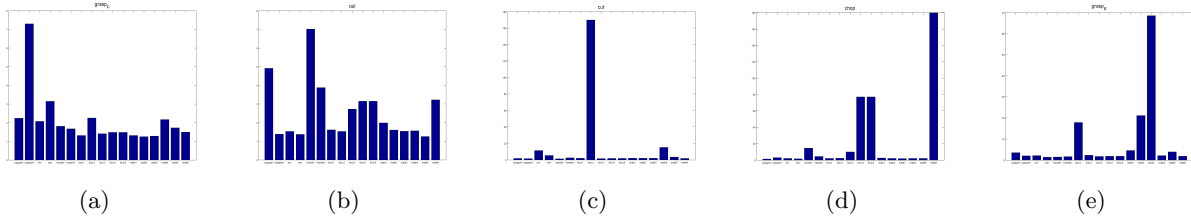


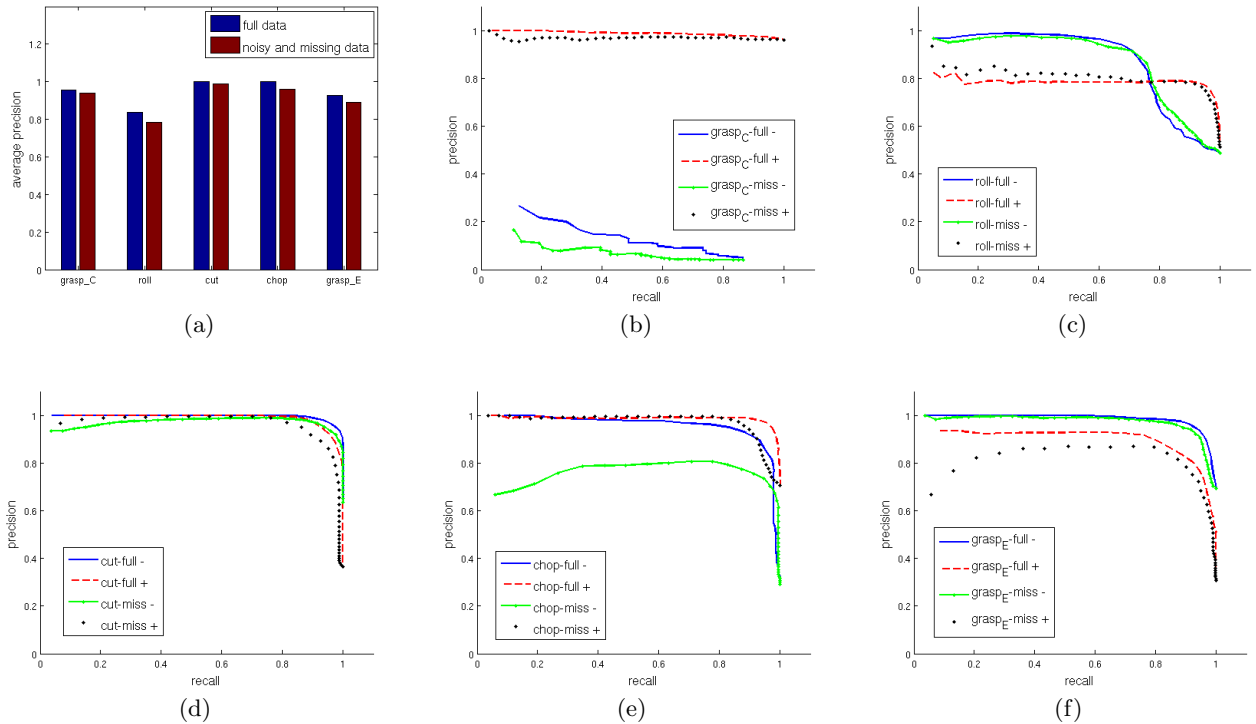
Figure 7: Check the dependency of five actions ((a) grasp by closing fingers (b) roll (c) cut (d) chop (e) grasp by expanding fingers) on category quantifications of object variables (from left to right bars denotes the maximum ratios (10) of <gripper-, <gripper+, rim-,rim+, handle-, handle+, container, food, cooker, cutting tool, eating tool, plastic, glass, ceramic, plant, animal, metal).

$V_a$	Depended category quantification of variable in $V_o$
$grasp_C$	<gripper-, <gripper+, handle-, rim-, rim+,function=food, material=plant
$roll$	<gripper-, handle-, handle+, function=cooker,function=cutting tool, function=eating tool, material=metal
$cut$	function=food, material=plant
$chop$	function=cutting tool, function=eating tool, material=metal
$grasp_E$	function=container, material=glass, material=ceramic

Table 2: Dependency of five actions on category quantifications of object variables after elimination (11).

section 3.3, and the recommendation is ranked based on ratios (15). Similarly to the effect-prediction experiment, 50 trials with different training and test data are run, and the average results of 5 actions (positive and negative) are presented in Figure 8(b)-8(f) with precision-recall curves. It can be seen that except for the poor results on grasping by

closing fingers, object retrieval of all other actions and effects are acceptable. The reason for poor performance in the negative case of grasping by closing fingers, according to our preliminary analysis, is that there are too few instances of  $grasp_C-$  in the training data; most objects in the kitchen are graspable. The results with missing-and-noisy



**Figure 8:** (a) The average precision of correct effect prediction of five actions; (b)-(f) the precision-recall curves of average object selection results in all positive and negative of five actions.

training data are slightly inferior to those with full training data. Two obvious performance gaps appear in the negative case of chopping, and in the positive case of grasping by expanding fingers. In conclusion, both effect prediction and object selection experiments quantitatively demonstrate the promising capabilities of our object-action relational model by displaying its high accuracies and robustness to noisy and incomplete data.

## 5. CONCLUSION

We presented a novel computational model of object-action relations. Actions are represented in terms of their effects on objects, and objects are represented as well in an action-oriented manner. The model can be effectively learned with homogeneity analysis and extra discovery of dependencies between action and object variables. One strength of the proposed model is that it does not require complex, highly-combinatorial descriptions of objects and actions. The object representations with respect to different actions are computed with only a small number of the most decisive object variables. Actions are presented by their positive and negative action-effect category quantifications. Another merit of the model, according to experimental results, is that it is robust to noisy and missing data, which is an unavoidable problem in practice.

## 6. REFERENCES

- [1] [www-roc.inria.fr/gamma/download/](http://www-roc.inria.fr/gamma/download/).
- [2] J. de Leeuw and P. Mair. Homogeneity Analysis in R: The Package homals. . Technical report, Department of Statistics, UCLA, 2007.
- [3] R. Detry, C. H. Ek, M. Madry, J. Piater, and D. Kragić. Generalizing Grasps Across Partly Similar Objects. In *International Conference on Robotics and Automation*, pages 3791–3797. IEEE, 2012.
- [4] R. Detry, D. Kraft, O. Kroemer, L. Bodenhagen, J. Peters, N. Krüger, and J. Piater. Learning Grasp Affordance Densities. *Paladyn Journal of Behavioral Robotics*, 2(1):1–17, 2011.
- [5] J. J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.
- [6] G. Michailidis and J. de Leeuw. The Gifi System of Descriptive Multivariate Analysis. *Statistical Science*, 13:307–336, 1998.
- [7] 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 International Conference on Robotics and Automation, ICRA 2012*, pages 4373–4378, May 2012.
- [8] L. Montesano and M. Lopes. Learning grasping affordances from local visual descriptors. In *IEEE 8TH International Conference on Development and Learning*, China, 2009.
- [9] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory-motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26, Feb 2008.
- [10] E. Oztop, N. Bradley, and M. Arbib. Infant grasp learning: a computational model. *Experimental Brain Research*, 158(4):480–503, 2004.