Bootstrapping Relational Affordances of Object Pairs using Transfer

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

***Abstract*—Robots acting in everyday environments need a good knowledge of how a manipulation action can affect pairs of objects in a relationship, such as 'inside' or 'behind' or 'on top'. These relationships afford certain means-end actions such as pulling a container to retrieve the contents, or pulling a tool to retrieve a desired object. We investigate how these relational affordances could be learnt by a robot from its own action experience. A major challenge in this approach is to reduce the number of training samples needed to achieve accuracy, and hence we investigate an approach which can leverage past knowledge to accelerate current learning (which we call bootstrapping). We learn Random Forest based affordance predictors from visual inputs and demonstrate two approaches to knowledge transfer for bootstrapping. In the first approach (direct bootstrapping), the state-space for a new affordance predictor is augmented with the output of previously learnt affordance. In the second approach (category based bootstrapping), we form categories that capture underlying commonalities of a pair of existing affordances and augment the state-space with this category classifier's output. In addition, we introduce a novel heuristic, which suggests how a large set of potential affordance categories can be pruned to leave only those categories which are most promising for bootstrapping future affordances. Our results show that both bootstrapping approaches outperform learning without bootstrapping. We also show that there is no significant difference in performance between direct and category based bootstrapping.**

I. Introduction

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

In this paper, we focus on bootstrapping the learning of one part of a robot's knowledge that is important for manipulation with pairs of objects, that is the knowledge about how the spatial relationship of a pair of objects affords certain outcomes.

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

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

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

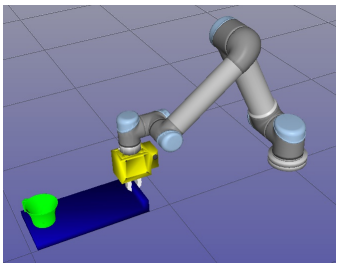

Fig. 1. Illustration of the Robot Simulation environment with the robot attempting to pull a tray-like object with a cup on top of it.

However, the general approach described here is also relevant for other bootstrapping tasks where there is prior knowledge, and rapid learning from few samples is desired.

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

The key technique which we explored for bootstrapping of this learning is inspired by works in cognitive science, where some unit of knowledge is abstracted from sensorimotor interaction and can then be reused where it benefits subsequent learning, for example the 'synthetic item' of Drescher and Chaput [9], [10]. These techniques have not been applied to high dimensional robot manipulation scenarios to the best of our knowledge (Ugur et al. [4] being a very recent exception). In our work, we learn individual affordances, and also affordance categories from early experience, where these implicitly capture spatial relationships like 'on top' and/or 'inside'. Note that the categories we learn are not imposed by the human designer, but rather are learnt by a classifier with the task to discriminate situations from visual input, where a certain

action will have one outcome or another. These individual affordances, or affordance categories, can then be used as binary inputs to subsequent learning, resulting in a speed up where they have discriminative power. The approach using individual affordances we call 'direct bootstrapping' (DB), while using affordance categories is called 'category based bootstrapping' (CB). We learn from simulated robot actions as this was the only feasible way to generate the ten–thousands of examples we needed to compare different approaches.

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

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

As a representation for learning, we use Random Forests. This allows us to compare category based bootstrapping (CB) with the simpler direct bootstrapping (DB). We show that direct bootstrapping achieves comparable performance to the category based approach, which indicates that category formation – although required for higher level transfer learning involving, e.g., language – might not be necessary for transfer on such a low sensorimotor level as dealt with in this paper.
To summarise the contributions of this work:

- We demonstrate a 'direct bootstrapping' approach that accelerates the learning of relational affordances in a basic state-space, and brings performance comparable with learning in a state-space with features tailored to the learning task.
- We realise a system in which autonomous category formation on relational affordances is performed during robot exploration.
- We show that by exploiting these categories, we can efficiently bootstrap the learning of new relational affordances leading to a significant speed up.
- We show that by applying a heuristic for selecting, we can circumvent the inherent problem of accumulating a large number of highly correlated categories.
- We compare the category based bootstrapping with the 'direct bootstrapping' approach, which leads to similar bootstrapping performance.

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

II. Literature Review

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

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

Within this development a particularly interesting stage concerns the acquisition of means-end behaviours which begins about 8 months (i.e. where one action is used in order to facilitate another [13]), because it is through learning means-end behaviours that infants begin to learn about relationships between objects [14]. For example in pulling a supporting object (e.g. cloth) to retrieve a more distant object (e.g. toy) that is on top, the precondition that determines success of the action must capture the relationship between the objects.

Much of the child's learning up to this point is related to relationships with its own body (subjective), such as something being reachable, or suckable. However the relationship 'on top' marks a beginning of learning more objective properties of the world. Probably this is initially learnt in quite a context-bound manner, but gradually it will be generalised.

According to Mandler's analysis [15] infants begin with 'perceptual knowledge', which is implicit in individual schemas, and develop towards more abstract 'conceptual knowledge'. Conceptual knowledge is essential for more advanced problem solving. The mechanism may also be the same as, or at least closely related to, 'representational redescription' [16]. As infants make this transition, they begin to see things at a higher level of abstraction, realizing precisely those relationships which are important in determining what object manipulations are possible (by the infant or other agents). For this reason we have explored the acquisition of chunks of knowledge that are locked in the context of a particular manipulation, as well as more abstract chunks of knowledge that begin to capture a category like 'on top'. We explore the use of both for bootstrapping subsequent acquisitions.

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

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

Rosman and Ramamoorthy [23] learn spatial relationships between objects using a support vector machine based approach, where support vectors are picked for their ability to differentiate the point cloud into two objects. This has the effect that the subset of points considered by the classifier are on the edges of the object. Relations are then learnt based upon the relative positions of clusters of the support vectors. Rosman et al.'s [23] approach makes sense for the relationships they considered like 'on' and 'adjacent', whereas we sought a more generic approach. Our perception approach preserves more information about the objects themselves, focusing not only on the interface between them. For example we capture elongation of the objects, or histograms of surface patches. This could be important for example when a small object is near to one edge of a large one, in a containment relationship.

A further work on support relations is by Panda et al. [24]. The work exploits a number of visually derived features regarding the relationship between the objects: proximity, boundary overlap, depth boundary, containment, relative stability. In addition, a rule based method is employed to infer what supports what, when multiple objects are stacked or leaning on each other. In contrast to this work, we have approached the problem more from a developmental robotics perspective; we are attempting to see what the system can learn without significant prior knowledge, and learning from the effects of its actions. The reasons for our preferring the developmental approach have been discussed elsewhere [25], [8].

All of these works which can recognise containment or support relationships (and implied affordances) have importance beyond the task of informing a robot of action possibilities. Affordance work is beginning to be used to help solve classic computer vision problems such as object categorisation [26] which take the approach of imagining an actor exploiting the affordance defining the object. Also it is highly relevant to learning about human activities from observations, for the purposes of imitation or understanding [27], [28]. These works use semantic scene graphs and can benefit from accurate descriptions of spatial relationships between objects within these graphs.

The main aspect of our work which deserves comparison with others is bootstrapping. As stated in Sec. 1, what we mean by 'bootstrapping' is a type of transfer learning: '*transfer learning* aims to extract the knowledge from one or more *source tasks* and applies the knowledge to *a target task*' [11]. In our work our feature space is the same for our source and target domains, but the source and target tasks are different, making this an example of 'inductive transfer' [11]. In terms of the detailed taxonomy on transfer learning by Lazaric [29], ours is 'learning speed improvement'; i.e. a reduction in the amount of experience required to learn the solution, which is distinct from transfer approaches which improve the asymptotic performance, or which 'jumpstart' to give better performance at the first attempt on the new (target) task; i.e. before training.

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

positive and negative bootstrapping effects on a variety of nine different actions.

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

In fact the work of Ugur et al. [4] above is attempting something of much broader scope than what we tackle in this paper; they attempt a staged development where there are successive developments which build on each other. This is an area of great interest to developmental robotics, and there are several more examples [30], [31], [32], [33]. One area of interest within this is whether the stages need to be prescripted [30], or could emerge naturally, e.g. as a consequence of some relatively simple intrinsic motivation mechanism [34]. We see our work as contributing to this area, because we believe spatial relationship categories learnt could influence subsequent behaviour and the experiences generated (e.g. the robot creates 'on top' and 'inside' situations), however this has not been pursued yet.

We do not feel that our work is particularly close to computer vision work in scene understanding (e.g. [35]) because those works typically recognise all objects, and then can use higher level knowledge to assist in understanding. Our work in contrast is at a lower level, and is more concerned with the physical relationships among surfaces without regard for object knowledge. We think of it more like how an infant might recognise simple physical relationships between household objects without any idea of what their names are or what their typical purposes are.

III. Methods

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

A. Learning Affordance Classifiers

The goal of the Affordance classifiers trained in this work is to accurately predict whether the action associated with the classifier can be executed successfully in a given scene. The actions are (with one exception) means-end actions where one action (means) is executed on one object in order to facilitate the successful execution of another action (end) on another object. It is the spatial relationship between the objects involved that determines whether the means action can succeed to facilitate the end action, and if the means will succeed we say the relational affordance is present. To learn the mapping between the current state of the environment and the affordance, the classifiers are trained using a state-space description including the positions of both objects relative to the robot and information describing the spatial relationship between the objects. In this paper, we investigate how additional knowledge can be used to speed up the learning of affordance classifiers.

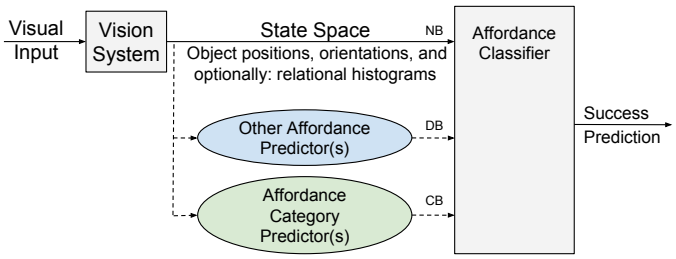

Fig. 2. Illustration of the inputs for learning affordance classifiers. The solid lines represent learning with no additional information (NB=No Bootstrapping). The blue ellipse illustrates the addition of other previously learned affordance predictions (DB=Direct Bootstrapping). The green ellipse illustrates the addition of other previously learned category predictions (CB=Category based Bootstrapping).

Throughout the paper, we will refer to the different classifiers with the following terminology:
Affordance Classifier: The classifier that is currently being learned with the goal of predicting the success of a new action.
Affordance Predictor: Already existing classifiers that predict the success of other existing actions. Their output is used to bootstrap the learning of a new affordance classifier.
Category Predictor: Classifiers that recognise specific categorical patterns in the environment. The category predictor's output is used to bootstrap the learning of a new affordance classifier.

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

- **NB**) Learning with no additional information - No Bootstrapping (represented by the solid line in Fig. 2).
- **DB**) Using previously learned affordance predictor(s), for other affordances, as additional input - Direct Bootstrapping.
- **CB**) Using previously learned category predictor(s) as an additional input - Category based Bootstrapping.

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

1) Learning Without Bootstrapping (NB): Learning without bootstrapping corresponds to the typical isolated learning of an individual affordance classifier. An affordance classifier

receives as input a set of visually derived descriptors of a scene before an action execution trial and a success label differentiating successful trials from unsuccessful ones. The classifier learns a mapping from these visual inputs to success labels. The combination of all the inputs that go into the classifier, apart from the success label, define the state-space (see Fig. 2).

2) Direct Bootstrapping With Already Learnt Affordance Predictors As Knowledge Source (DB): Our approach to bootstrapping is based on extending the state-space of a new affordance classifier with the output of one or multiple affordance predictors already learnt for other affordances. If the new affordance is related to an already existing affordance, the state-space extension can carry relevant information and hence bootstrap the learning. For example if we already have an affordance predictor for 'pull' then the learning of 'lift' could be effectively bootstrapped by having the 'pull' affordance predictor as input (because the situations where pulling moves two objects are very similar to the situations where lifting lifts two objects).

We investigate direct bootstrapping by extending the state-space with the output of:

DB1 A single already existing affordance predictor,
DBn All already existing affordance predictors,

().

3) Category Based Bootstrapping: Automatically Created Categories As Knowledge Source (CB): This approach extends the state-space of a new affordance classifier with the output of a category predictor. A category is an abstraction from the state-space that describes properties of the environment. E.g., the abstract notion of one object being 'on top' of another could be captured by a category predictor. Further examples are one object 'inside' another or an object being 'in reach' of the robot's arm.

The main difference between category predictors and affordance predictors is the degree of abstraction, which can have important consequences in a cognitive architecture. While affordance predictors only represent the preconditions for one specific action (e.g. 'pull'), category predictors generalise across similar affordances (e.g. 'pull' and 'lift') to capture an abstract relationship between objects such as 'on top'. In section III-C we investigate the correlation (pairwise) between our different affordances. We found one set of affordance predictors that is strongly correlated and which corresponds to the 'on top' relation between objects. In addition, there is one more weakly correlated set of relational affordances that roughly corresponds to a 'beside' relation.

The condensation of such correlated affordance predictors to an abstract category predictor has two advantages:

1) It reduces the input to a new affordance predictor to one binary value instead of a set of predictors. This is important since the combination of all possible affordance predictors would lead to a combinatorial load once large sets of action types are performed that would lead to larger memory and computational requirements.
2) In a larger cognitive architecture, such category predictors can constitute symbolic representations that could be used for high-level planning; the idea of developing such symbols autonomously rather than having them provided by a human designer is increasingly being recognised as important [36], [37]. As one of the main achievements of our approach, we suggest a way for creating categories automatically.

We investigate two different variations of category based bootstrapping:

CB1) Adding the output of a single existing category predictor,
CBn) Adding the output of all existing category predictors.

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

To create a category predictor, we combine the training data that was used to train two individual affordance predictors. This combined training set is then used to train a new classifier that captures properties of the state-space, that are required for both of the two different affordances. In this initial approach, we limit ourselves to creating categories from two affordances while the concept could be extended to combine more affordances, e.g., by finding clusters of correlated affordances (i.e., affordances that are often present in the same states).

B. Classifiers

To predict the affordances in a particular scene, we use ensemble classifiers based on the Random Forest [38] algorithm. In the following sections, we will first describe Random Forests in general and then give details about the parametrisation of the Random Forests used in this work.

1) Random Forest Classifiers: The idea of ensemble classification methods is to turn a set of 'weak' classifiers into one 'strong' classifier, where the final classification is based on some form of voting scheme. In Random Forests, these 'weak' classifiers are standard *Classification and Regression Trees* (CART) [39], which iteratively split the dataset into more and more subsets where each split maximises the class purity of the created subsets. The number of trees used can vary from ten up to hundreds or even thousands of trees per forest. The final prediction of the Random Forest is the class that is returned by most of the individual trees.

Random Forests (RFs) have a series of advantageous properties compared to other types of classifiers, e.g. support vector machines or artificial neural networks, some of which they inherit from classification trees. We focus on just two advantages here which make RFs particularly well suited for our use case. Firstly they can handle both numerical and categorical data. Numerical data comes from our perception (see Section IV-B on feature spaces used), while categorical data is used as our bootstrapping input. The learner needs to find which data is most useful, and again RFs are advantageous: they inherently do feature selection and hence identify the relevant features

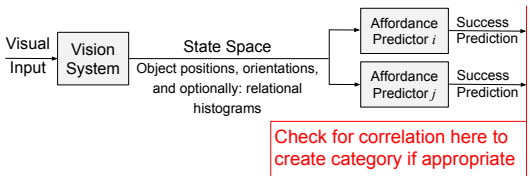

Fig. 3. Searching for correlations between affordance predictions.

from the (potentially) large amount of state-space variables (see Section IV-B).

2) Random Forest Parameterisation: The amount of trees (T) in our Random Forests is dependent on the amount of samples available for training (N), where $T = 95 + N/20$ up to a maximum size of 115 trees (the maximum forest size is reached with 400 samples). To parametrise the trees in our Random Forests, we followed the guidelines of Breiman et al. [38], [40]. For each tree in the forest, we use $2/3$ of the samples in the training set to train the tree. We use standard CART trees as base classifiers using the *Gini impurity criterion* [39] to split nodes and allow a minimum size of 1 sample for a partition created by a split. We do not prune the trees, but we do limit growth to a maximum depth of 200 splits. For every split, each tree uses all samples available to it ($2/3$ of the samples available to the forest), but at each node, out of M dimensions, only $m = \sqrt{M}$ dimensions are randomly picked, and the best split on these is used to split the node.

C. Heuristic For Category Formation

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

We hypothesise that strong correlations between affordance predictions indicate that a meaningful category predictor can be learnt. Hence, as a heuristic we use the correlation between pairs of affordance predictor outputs. To compute this correlation, two affordance predictors are presented with the same scenes. The agent then records their predictions and calculates the correlation between them (see Fig. 3). A category predictor is created only if the correlation exceeds a certain threshold.

D. Bootstrap Factor

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

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

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

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

IV. Experimental Setup

The majority of our experiments are done using a physically realistic simulated robotics setup (Section IV-A). To learn an affordance predictor for a particular action, say 'lift', a large set of training examples is gathered. Each example places a pair of objects in positions determined by a probability distribution to give about 50% probability that one object is on top of the other, then the visual scene is captured. Next the robotic manipulator attempts the action on one of the objects, and the outcome is recorded. A positive action outcome (i.e., the pair of objects moves together, in the case of 'lift') is used to automatically label the vision data as indicating that the affordance is present.

Subsequent to learning individual affordance predictors we run bootstrapping experiments which use the same labelled training data, but have additional bootstrapping inputs as described in Secs. III-A1 and III-A2.

The specifics of the experimental setup are described in the next subsections: Section IV-B describes the perception system. Section IV-C describes the objects used. Section IV-D describes the actions of the manipulator.

A. Simulation Environment

To collect data we used a simulator designed for robot simulations [41]; this also simulates a Kinect sensor [42] with an appropriate noise model [43] to produce data that resembles the depth map results from real Kinect camera setups.[1] We simulated a robot arm with six degrees of freedom (DOF), mounted on a table with a two finger gripper attached as tool (see Fig. 1).

Our experiments are based on ca. 1.5M action simulations, which in total took approximately 900 hours (ca. 35 days) of processing in total (on a computer cluster running 32 simulations in parallel with approximately 1 minute runtime per action simulation). Processing of all the point cloud data into both, PCA and RHF features took approximately 1 day. The learning of all classifiers for all affordances and categories, with and without bootstrapping, with the different bootstrapping inputs, using different state spaces, their 25 and

[1]The noise model used in this work adds noise on the z-coordinate of the points in the point-cloud based on depth and location in the projected image. This is a reasonable and close approximation of the real sensor [43]. (By now newer models have appeared that use relative surface angle as an additional factor [44]). Our simulated sensor does not account for the following phenomena: depth data is smoothed and step depth discontinuities are not represented faithfully by the physical device, the real sensor is not able to deal with strong foreign light nor strong specular reflections. The later two are typically avoided in real setups by careful selection of recording location and object choice. The former should only introduce small deviations in z-direction of a limited set of points and we anticipate that our process is robust to these.

50 repetitions, lead to an overall Random Forest learning runtime of approximately 1.5 weeks on the same computer cluster.

B. The Robotic Perception System

The Kinect camera is positioned in the workspace on the opposite side of the robot, looking down on the workspace area in front of the robot. The Kinect records images with VGA resolution (640x480 pixels) and provides one 3D point per pixel (307200 points per scene). For performance reasons, we subsampled this point cloud via voxel grid subsampling with a resolution of 0.0125 m which resulted in point cloud sizes between 40k and 50k points.

We use simple colour based segmentation in this work. All objects are coloured in one of a set of known colours, and every object in the scene has a different colour selected from this set. All points that have the same colour are assigned to a new point cloud, representing one object. This is a common simplification also used in real vision set ups, e.g., by Rosman and Ramamoorthy [23][2]. We acknowledge that highly sophisticated object segmentation algorithms exist, e.g., [45], [46] and we assume they could be employed to work in a more complex environment, e.g. with real objects and clutter.

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

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

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

[2]The mentioned approach uses stereo but this is equally applicable to the Kinect sensor since for Kinect the necessary registration between point-clouds and colour is straightforward.

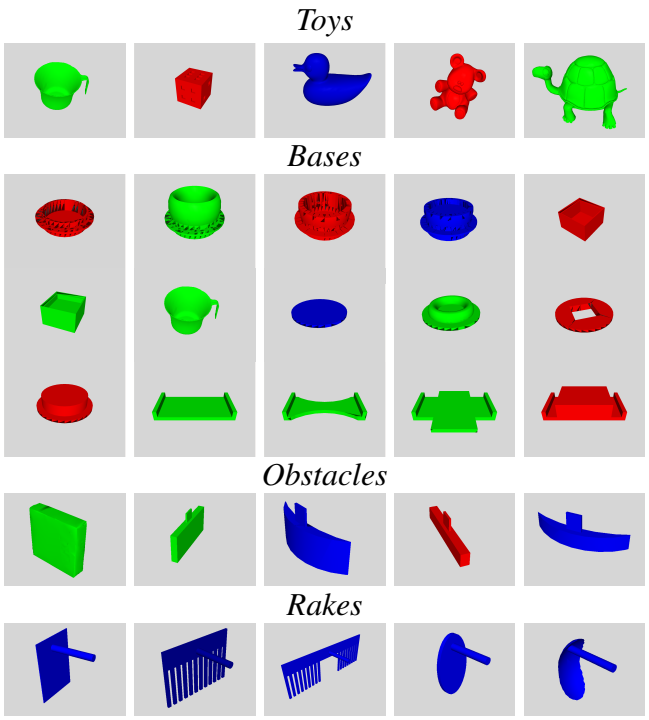

Fig. 4. Illustration of the Objects used for the experiments of this work

Independently from the Eigen decomposition based object representation, the segmented point clouds are also used to create *relational histogram features* (RHF) [47] to capture the spatial relationships between objects. These RHFs form a relational space into which the absolute geometric information (3D position and orientation) of the segmented object point clouds is transferred. For this, every point of the first object is compared with each point of the second object to calculate a set of distance and angle features. These relational features encode the spatial relationship of the two objects and are captured in form of a histogram with 300 bins. This RHF extraction process is described in more detail in [48], [49] and we use the RHFs described there as 1D histograms as these were found to show the best performance in [48].

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

C. Objects

In our experiments, we used an overall set of 29 objects, which can be split into four different groups (see Fig. 4)[3]:

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

TABLE I
LIST OF ACTIONS

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

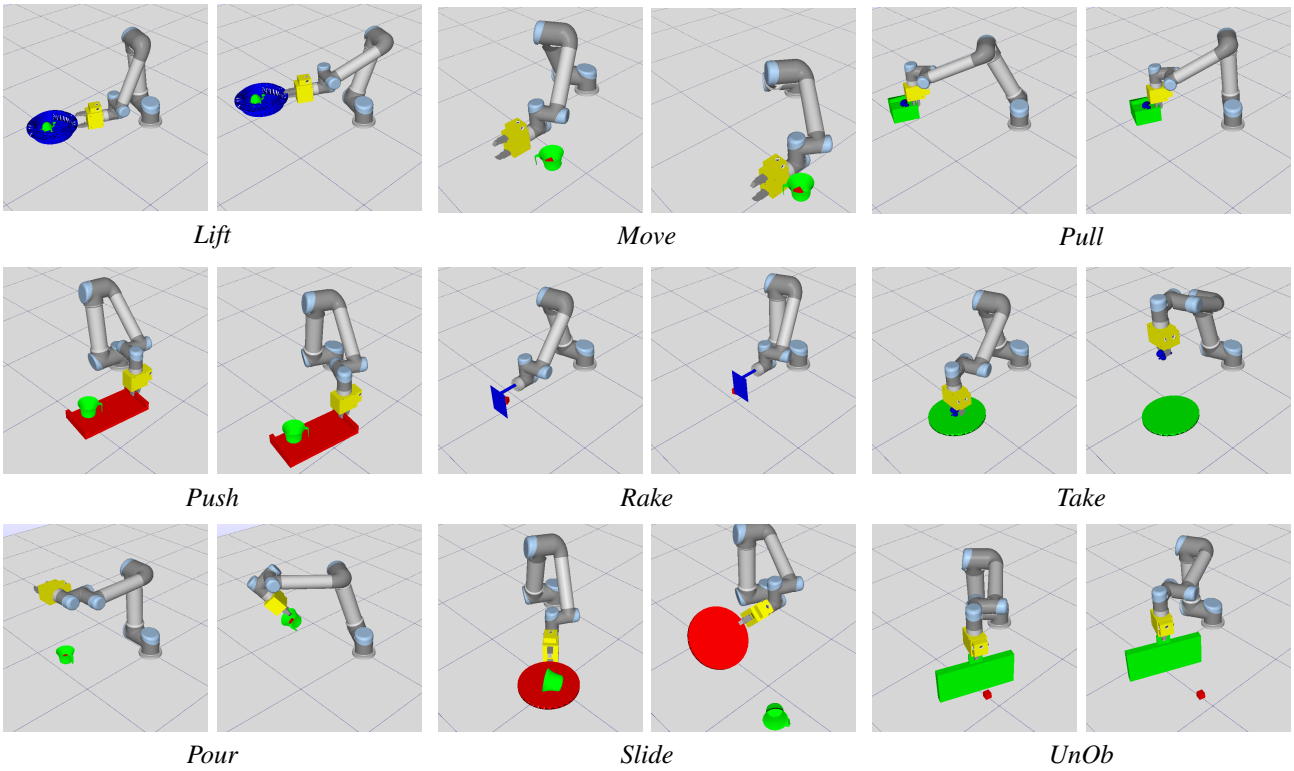

Fig. 5. Action execution effect snapshots of the nine actions. The left figures show a scene before action execution. The right figures show the state afterwards. (UnOb is short for Unobstruct.)

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

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

D. Actions

The robot is equipped with nine pre-defined actions it can perform. Depending on the two objects in the workspace (one of which is always a toy object) different actions are available for execution. Table I briefly describes the actions, their motor programs, and the objects involved and the affordance being tested. Fig. 5 illustrates the actions' motor programs. The 'pour' action does not actually attempt to pour out contents, instead it does a partial pouring action, to check if the container has the affordance of retaining the object despite a tilt. It is closely related to slide, with an opposite goal, however it is not completely the inverse as both pour and slide fail when objects are 'beside'.

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

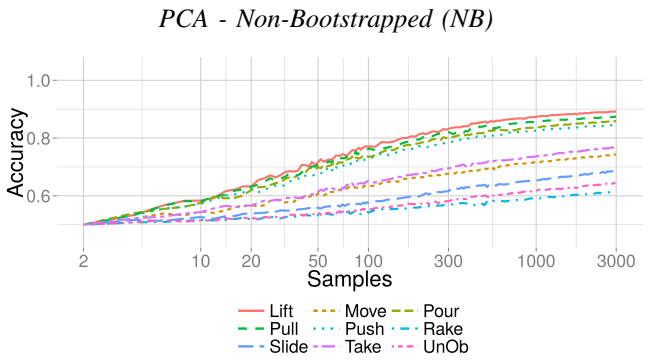

Fig. 6. Illustration of affordance classifier learning for nine affordances using the PCA state-space without bootstrapping. The X-axis is log-scaled to highlight that the learning continues even after several thousand training samples.

bootstrapping results.

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

V. Results

In the following subsections, we present the results of learning affordance classifiers based on the different approaches introduced in Section III. The results presented here show the average of 50 repetitions in the PCA case and 25 repetitions in the RHF case. For each repetition, from the overall available data n samples with $2 < n < 100$ have been randomly selected for training and 4000 different samples have been randomly selected for evaluation. The first four plots also show the standard error of the mean (SEM), in form of shaded regions around the repetitions' average.

The results of learning without bootstrapping (NB) are presented in Subsec. V-A. The results of direct bootstrapping following the approaches (DB1) and (DBn) are shown in Subsecs. V-B and V-D respectively. In Subsec. V-E the results of bootstrapping from a single category (CB1) are presented. The outcome of using the heuristic for category formation is shown in Subsec. V-F and the result of learning affordance classifiers from all created categories (CBn) is presented in Subsec. V-G. In section V-H we describe an approach to select appropriate inputs for bootstrapping.

A. Non-Bootstrapped Learning of Affordance Classifiers (NB)

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

The long term learning rate for the different affordance classifiers using the PCA state-space representation is illustrated in Fig. 6. We can see that learning continues after learning from several thousand training samples. It is also notable that some affordance classifiers are harder to learn than others, which warrants explanation: The success of the actions depends not only on the spatial relationship between the objects in the scene but also on the stability of the predefined motor programs. Some of the motor programs had a relatively high fail rate, e.g., the take motor program often failed to grasp the toy object, even if it was in reach and in the spatial relation that would afford the take action to succeed. This acts like a kind of noise that makes learning difficult.

The early stage of learning affordance classifiers for the same affordances using different state-space representations are highlighted in Fig. 7. We can see that the state representation using RHFs in general massively outperforms learning without histograms (PCA). This is not surprising as the RHFs were specifically designed with this use case in mind. We show in Section V-B that we can achieve similar performance in the PCA state-space via bootstrapping.

Nevertheless, the 'Take' affordance serves as a good example of the potential shortcomings of such specialised state-spaces as RHF. The 'Take' affordance has a weak negative correlation with an aspect the RHF is able to represent (i.e., 'inside') but has a much stronger correlation with aspects of PCA space (e.g., object orientation). Even though our RHF space includes all PCA variables, the increased amount of input variables in the RHF case swamps the RF and causes a decrease of learning speed (curse of dimensionality).

The non-bootstrapping affordance classifier results presented in this section serve three purposes: Firstly, they illustrate the 'standard' learning rate for learning affordance classifiers when learning without bootstrapping and will be used as a baseline for comparison in the following sections. Secondly, they constitute part of the 'knowledge' that will be used for bootstrapping in the following sections. Thirdly, the RHF based results serve as performance 'benchmark'. Using bootstrapping in the PCA state-space, we attempt to achieve similar learning speeds and accuracy as with the specially designed RHF state-space.

B. Direct Bootstrapping With a Single Already Learnt Affordance Predictor as Knowledge Source (DB1)

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

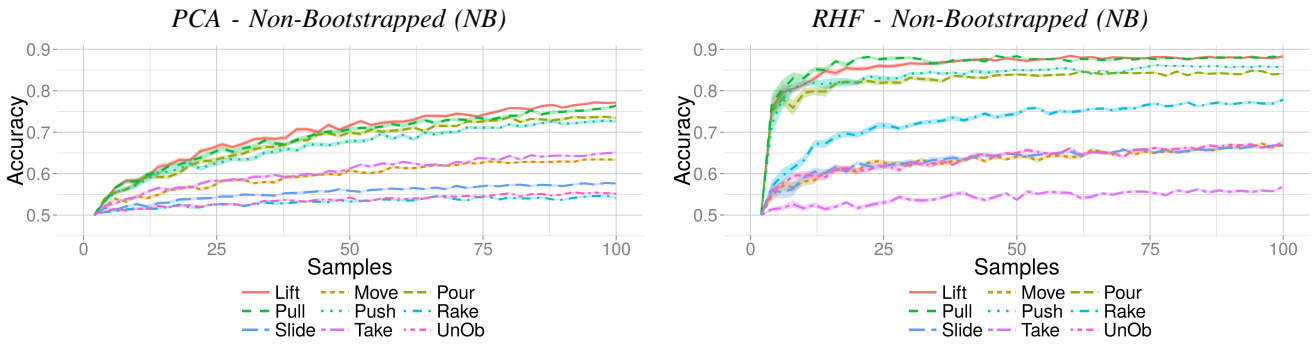

Fig. 7. Illustration of the early stage of affordance classifier learning for nine affordances in different state-spaces without bootstrapping. Note that the RHF space generally outperforms PCA space, but some affordances like 'take' are learnt better in PCA space.

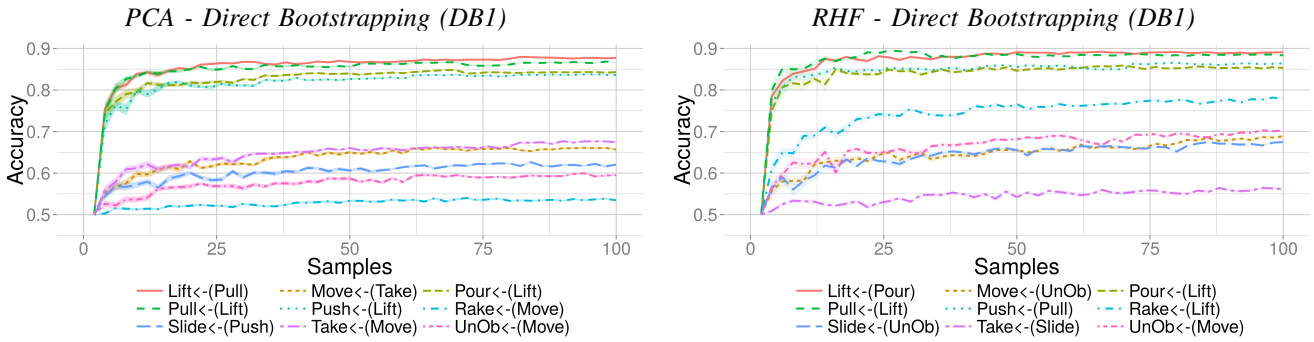

Fig. 8. Illustration of the affordance classifier learning speed for nine affordances in different state-spaces using another affordance predictor for direct bootstrapping. For each affordance we illustrate only the best result (i.e. augmenting its state-space with the affordance predictor which gives the best bootstrapping). The notation X<-(Y) means Y was used to bootstrap the learning of X. We see that direct bootstrapping in the simple PCA state-space can achieve similar performance to learning in the RHF state-space (Fig. 7).

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

Fig. 8 demonstrates the direct bootstrapping on all nine affordances. For each of the nine affordances we tried all 8 other affordances as input; in Fig. 8 only the input giving the best bootstrapping result is presented, highlighting the potential of bootstrapping. This 'best' result is the one with the highest average accuracy from two to 100 samples. Comparing the PCA cases from Fig. 8 and Fig. 7, the increase in learning speed achievable through bootstrapping becomes evident. Fig. 8 also demonstrates that bootstrapping can bring the performance in simple state spaces (PCA) to parity with the specifically designed state-spaces (RHF).

Fig. 9 emphasises the bootstrapping effect of the results demonstrated in Fig. 8, by presenting the corresponding bootstrap factors (as defined in section III-D). Two observations can be made here: **a)** The bootstrap factors are quite large with values between three and eight in the PCA case, especially when learning from few samples. In the RHF case, however, an approximate bootstrap factor of one indicates that there is no or only little bootstrapping effect. The bootstrap factors then decrease until they asymptotically approach a value of 1.0 as the performance of the unbootstrapped learner catches up with the bootstrapped learner. **b)** Fig. 9 also highlights the fact that bootstrapping in simple state-spaces like the PCA case is multiple times more effective than in specialised state-spaces where learning is already efficient without bootstrapping like in the RHF case.

It is evident that, in order to achieve these bootstrapping effects, the single best candidate from a group of available affordance predictors has to be selected for bootstrapping. This is also the case when using single categories (CB1) for bootstrapping. How this selection can be achieved in practice is described in Sec. V-H.

We demonstrated here that bootstrapping in PCA state-spaces can lead to similar performance as the RHF state-space (which is specially designed for the task) and that there is no significant bootstrapping achievable in the RHF state-space, we will therefore, in the following, only present the bootstrapping results of the PCA state-space.

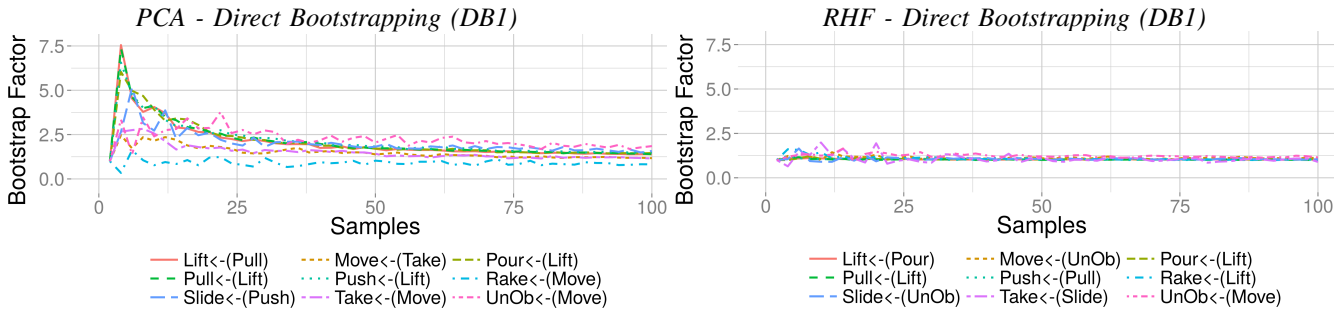

Fig. 9. Illustration of the bootstrap factors of nine affordance classifiers in different state-spaces using another affordance predictor for direct bootstrapping. For each affordance we illustrate only the best result (i.e. augmenting its state-space with the affordance predictor which gives the best bootstrapping).

TABLE II
AVERAGE BOOTSTRAP FACTORS RESULTS INDICATIVE OF OVERALL PERFORMANCE FOR PCA.

	avg	min	max
NB	1	1	1
DB1 average	1.356	0.8055	1.582
DB1 best	2.819	0.9760	3.657
DB1 best after ten	2.751	0.8756	3.657
DBn	2.357	1.064	3.090
CB1 average	1.548	0.7860	1.913
CB1 best	2.889	0.8503	3.701
CB1 best after ten	2.802	0.7702	3.752
CBn	2.895	0.9072	3.888

C. A Single Measure for Bootstrap Factor

Looking at Fig. 9 we can see that bootstrapping happens in early learning (if it happens at all), roughly from samples 4 to 25 (and this holds true for a large number of bootstrap plots that we looked at). For this reason we will now reduce the Bootstrap Factor time-series for various bootstrap methods to a single number. This also makes it easier to compare various bootstrap methods side-by-side. We call this the 'average bootstrap factor'. It averages the individual bootstrap factor over the interval of 4 to 25 training examples. Fig. 10 shows a bar chart which allows comparison of the bootstrap methods, including DB1 already discussed as well as the other methods to be discussed below.

Note that the 'DB1 best' values in Fig. 10 are the average bootstrap factors from Fig. 9. In addition we include in Fig. 10 a 'DB1 average' which is the average of using the eight available other affordance predictors as an input to bootstrap a particular affordance. This 'DB1 average' corresponds to the expected value of the average bootstrap factor if a random affordance predictor is used for bootstrapping. This is interesting e.g. in the case of sequential learning of affordances where there is only one action available to bootstrap the second.

To further compact the data and to be able to compare the different bootstrapping approaches we in addition average the average bootstrap factors of the nine affordances used in this work. This is shown in Fig. 10 (rightmost bars). Table II shows this number for the different bootstrapping strategies (column labelled avg), as well as the highest and the lowest average bootstrap factor out of the nine affordances.

D. Direct Bootstrapping With All Affordance Predictors As Knowledge Source At Once (DBn)

Figure 10 shows DBn, the results when using the affordance predictions of all eight other affordances as knowledge source, as described in Section III-A2. The bootstrapping effect in this case is lower than with the single best other affordance prediction, 'DB1 best'. This is because not every affordance prediction is helpful as an input for bootstrapping; some hinder learning by adding noise to the state-space and increasing the size of the state-space (the curse of dimensionality [50]). The DBn approach, however, is the most straightforward one, as it relieves the agent from searching for the best bootstrapping input from a set of candidate affordance predictions and also does not require the learning of category predictors.

E. Category Based Bootstrapping With a Single Automatically Created Category Predictor (CB1)

Figure 10 shows the results when using the output of a single automatically created category predictor as knowledge source (CB1), as described in Section III-A3. We show 'CB1 average' which is the expected value of picking a random category for bootstrapping, and 'CB1 best' using the best category for bootstrapping. As before, we added the inputs six times to the PCA state-space. Following the binomial combinatorics rule there are 36 distinct category predictors we create by choosing two out of our nine affordances.

We see that CB1 category based bootstrapping is comparable to DB1 direct bootstrapping from single affordance predictions.

F. Using a Heuristic For Creating Category Predictors

In order to avoid the combinatorial increase of the number of categories created by our approach, we limit the creation of categories to those that are based on strongly correlating pairs of affordance predictors (see Sec. III-C). This is based on the hypothesis that strong correlations between affordance predictions indicate a precondition that is meaningful for a variety of affordances. The correlations between the nine affordance predictors are shown in Fig. 11 (after learning from 2000 samples).

Two clusters can be noted in Fig 11. The first cluster contains primarily 'Lift', 'Pull' and 'Push', with 'Pour' and

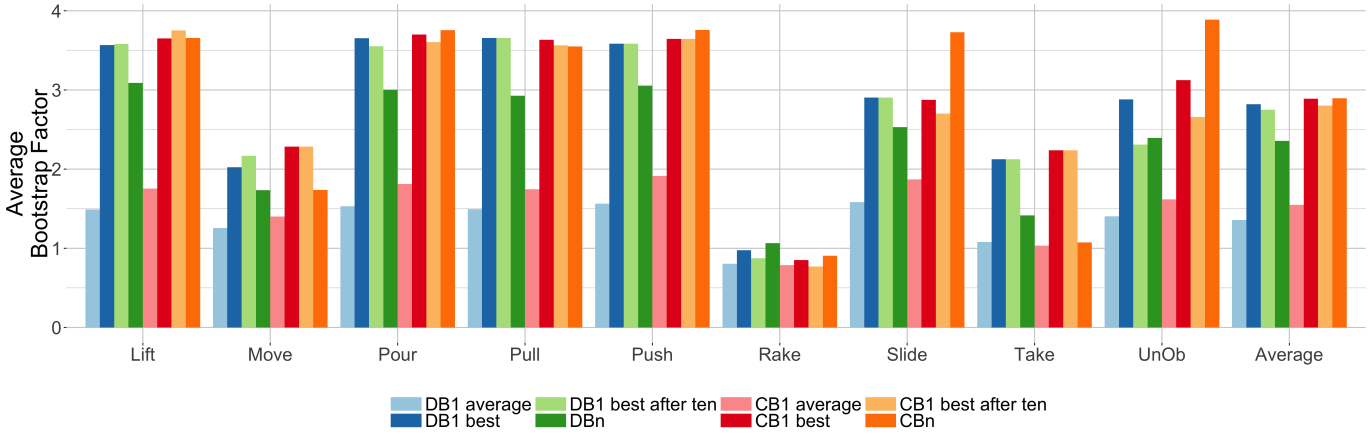

Fig. 10. Average bootstrap factors for the different bootstrapping types and actions using PCA.

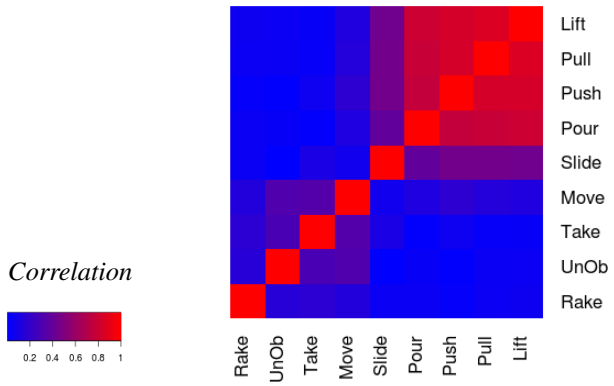

Fig. 11. Correlations between the affordance predictions for nine different affordances after training with 2000 samples.

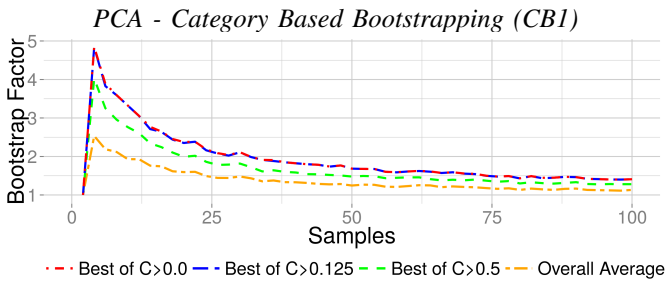

Fig. 12. Illustration of the bootstrap factor of different category predictors, averaged across all nine affordances. The plot shows the bootstrap factor of the best category created above the following correlation thresholds: 1) 0.0 (i.e., no heuristic used), 2) 0.125 (only approximately $1/2$ of the possible category predictors are created) and 3) 0.5 (only approximately $1/4$ of the possible category predictors are created). The plot also shows the overall average bootstrap factor using all categories, which corresponds to expected performance from bootstrapping with a random knowledge source.

'Slide' being more weakly related. The three affordances 'Lift', 'Pull' and 'Push' have in common that they are fairly likely to be present when the toy object is 'on top' or 'inside' of the base object, while they never work otherwise (i.e., when the objects are only beside each other on the table). The affordances 'Pour' and 'Slide' are related in the sense that they share these properties with 'Lift', 'Pull' and 'Push', and differ only slightly with 'Pour' being unsuccessful in the 'on top' case and 'Slide' being unsuccessful in the 'inside' case.

The Second cluster is weaker and separated from the first cluster as the affordances 'Move', 'Take', 'UnOb' and 'Rake' are more likely to be present when the two objects are 'beside' each other on the table than if they are 'on top' or 'inside' each other. At the same time, no two affordances in the second cluster have identical cases where they work and where not (unlike 'Lift', 'Pull' and 'Push' in the first cluster), hence there is a generally weaker correlations between them.

The heuristic we use requires only one free parameter: the correlation threshold T, specifying the minimum correlation required between two affordance predictors, for a category predictor to be created. A high threshold will lead to fewer category predictors, but the resulting category predictors are more likely to be useful for bootstrapping. A low threshold instead will lead to more category predictors, but not all of them might be useful for bootstrapping a new affordance classifier.

We experimented with three values for the correlation threshold T:

1) $T = 0.0$ (i.e., the heuristic is not used)
2) $T = 0.125$ (approx. $1/2$ of the category predictors created)
3) $T = 0.5$ (approx. $1/4$ of the category predictors created)

For a given T value we create category predictors for all the categories with correlation C exceeding T. We then analysed the bootstrap factor of each category $C > T$, and only the best is illustrated in Fig. 12.

We can see in Fig. 12 that the drop in performance through increased thresholds is small. The average performance of the full set of category predictors (threshold $T = 0.0$) and the $1/2$ set of category predictors (threshold $T = 0.125$) are almost identical. Even when using the threshold $T = 0.5$, where only $1/4$ of the category predictors are created, the bootstrap factor only drops slightly. Not surprisingly, the bootstrap

factor for each of the three sets is significantly better than the bootstrapping when using an arbitrary pick of available affordance predictors as knowledge source (Overall Average).

We can conclude that it is in general possible to neglect a significant proportion of category predictors without significant loss of performance. This is because even a small number of category predictors based on highly correlated affordances are sufficient to give good bootstrapping effects for most affordance classifiers.

G. Category Based Bootstrapping With All Available Category Predictor Outputs As Knowledge Source at Once (CBn)

The last section showed that an elimination of categories by means of a correlation threshold of 0.5 does not lead to any visible loss of performance. We discuss in this section the results when using all category predictors using this correlation threshold. This is CBn in Figure 10.

When bootstrapping from all available affordance predictions, DBn, we noticed that the performance was not as high as when bootstrapping from the single best affordance prediction DB1 (see Section V-D). We argued that this was due to the curse of dimensionality. However, when bootstrapping from all available category predictions, CBn, the learning and bootstrapping performance remains high and sometimes surpasses the results of learning with the single best affordance prediction DB1 or single best category prediction CB1 as input for bootstrapping (apart from the 'Take' affordance that does degrade in performance). this may be due to the more generic and hence transferable spatial relationship knowledge captured by category predictors, compared to the knowledge made available by affordance predictors.

H. Candidate Selection For Bootstrapping

Figure 10 shows the results for bootstrapping each affordance classifier with the input that is 'best', however we did not yet address the issue of how to the select the best input. An exhaustive method is to select the best input after all alternatives have been tested and compared. This is not viable in an online system where the input has to be selected automatically and rapidly. We expect that the suitability of a candidate input becomes clear after a small number of samples are processed. If this was not the case, bootstrapping of learning a new affordance classifier and reaching high accuracy after few training samples (i.e., bootstrapping as presented in this paper) would not be possible.

Therefore, here we start by learning multiple affordance classifiers in parallel – one for each knowledge source candidate. After ten samples have been processed we can decide which input to retain, and to stop the other parallel processes. Figure 10 shows this as 'best after ten'. The bootstrap factors of this automatic selection approach are similar to when using the best input. This shows that this simple selection approach is effective.

VI. On the Validity of Simulation based Results

In this section we investigate how our simulated results can be transferred to the real world. For this we present work that involves data collected from both simulation and real world setups. We believe that our result would match real world results due to a) the realistic noise model used for the simulated Kinect camera [43] and b) the relatively robust-to-noise way in which our vision features are created [48].

The following subsections are structured as follows: Subsection VI-A describes the approach, Subsection VI-B presents the results, and Subsection VI-C concludes.

A. Approach

The major difference between the real world work reported in this section and the simulations in the rest of this paper is that here we learn to recognise spatial relationships between objects, rather than affordances (predicting action success). In this section training examples (scenes) are set up much as before, but we manually label the data with the spatial relationship between the pair of objects rather than executing an action. In [49] we have shown that affordances strongly correlate with spatial relationships between objects. Hence, we have reason to expect that our simulation results on affordance recognition should translate well to the real world.

Fig. 13 illustrates the learning approach of this real world comparison work.

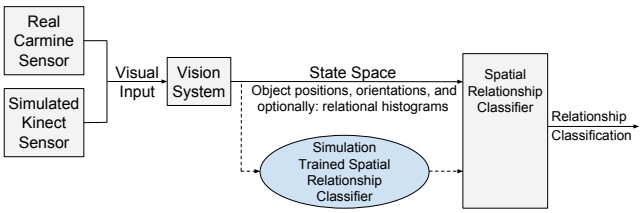

Fig. 13. Illustration of the inputs for learning spatial relationship recognising classifiers.

The following training and testing approaches were used:

1) Learning from simulation and testing in simulation.
2) Learning from the real world and testing in the real world.
3) Learning from simulation and testing in the real world.
4) Using a classifier learnt in simulation to bootstrap learning in the real world.

All four approaches use identical features for training Random Forest classifiers to recognise the spatial relationship between a pair of objects.

For collecting real world data, a Carmine 1.09 sensor (both Carmine and Kinect use the same PrimeSense technology internally) was used and positioned above a workspace similar to the Kinect in the simulation, looking diagonally down onto the workspace area. On the workspace objects were randomly placed to be a) both on the workspace surface, b) on top of each other, or c) inside one another; which reflects the labels that the simulator provided for a pair of objects. The spatial relationship labels where manually assigned. The objects used in the real world setup were of different colours. After removing the background with RANSAC plane removal, k-means clustering was used to find three clusters; the two

biggest clusters belong to one object each and the third cluster to the background.

The PCA features are created using the same approach as described above (see Section IV-B). The Random Forest classifiers are also the same except that there are always 100 trees in a forest.

For the real vision based classifiers, at each repetition, from a pool of 160 (80 positive and 80 negative) samples, 100 (50 positive and 50 negative) samples have been randomly selected for training, and the remaining 60 samples have been used for testing. Since we had 6500 simulation based training samples available it would not be a fair to compare with the real world trained on 100 samples. Therefore we made two sets of simulation based classifiers, one SIM_full that was trained from 6500 simulation based training samples, and one SIM_fair that was trained from 100 simulation based training samples. Both sets of simulation based classifiers where tested on the same amount of samples as they were trained from (6500 and 100), which were different from the training samples. All Training and Test sets have been 50% positive and 50% negative samples.

B. Results

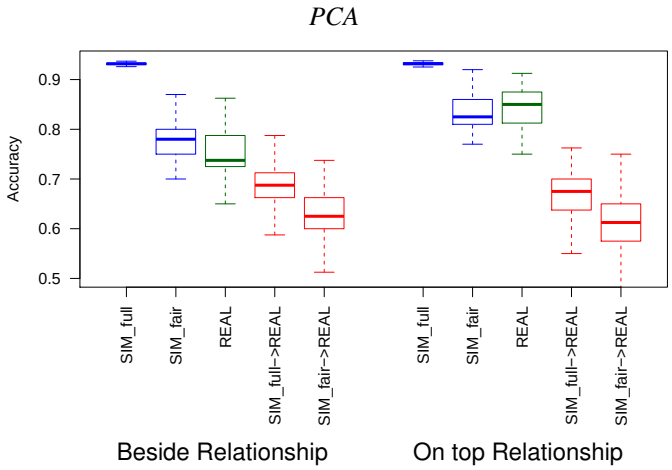

Fig. 14. Illustration of the classifier performance in different train/test configurations for the PCA state space. The blue boxes labelled SIM show the results when learning and testing in the simulation environment. The green boxes labelled REAL show the results when learning and testing in the real world environment. The blue boxes labelled SIM ...->REAL show the results when a classifier trained in simulation is tested on real world data.

The results presented in this section are based on the average of 50 repetitions. Fig. 14 shows the achieved results of the first three approaches (see Sec. VI-A for approaches) in the PCA state space, using both the spatial relationships 'beside' and 'on top'.

We can see that learning and testing purely in simulation works very well. The real world classifiers have lower performance and higher variance than the purely simulation based results, but still show good performance. We can make a similar observation for classifiers trained in simulation and tested on real world data: while there is a significant drop in performance and increase of variance compared to purely simulation based results, the simulation trained classifier does still show good performance when evaluated in real world settings and can compete with real world trained classifiers.

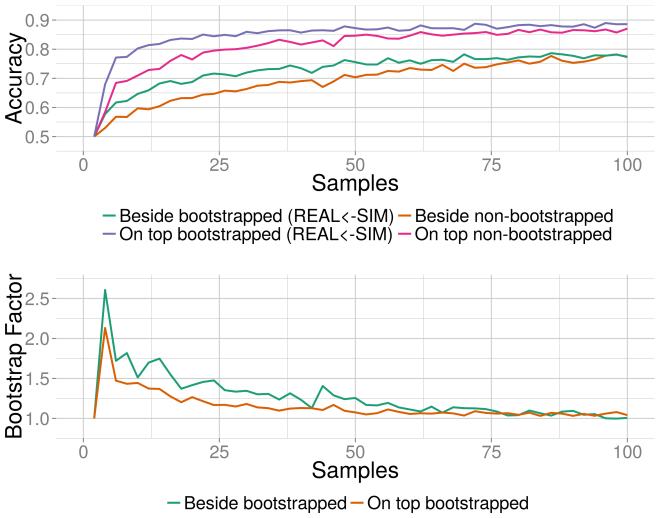

Fig. 15. Top Graph: Illustration of Bootstrapping in the real world when using the prediction of a simulation trained classifier as bootstrapping input (REAL<-SIM), and also non-bootstrapped learners for comparison. Bottom Graph: The bootstrap factors of the same two bootstrapped curves above.

The results regarding the fourth approach described above are shown in Fig. 15. Here the simulation trained classifier (SIM_full) is used to provide a prediction for a real world scene. This prediction is provided as additional input to the classifier trained on the real world data. It can be seen in Fig. 15 that also in this setting the benefit from bootstrapping at the early stage of learning is significant.

C. Conclusion

With these results we demonstrated four outcomes: **a)** The PCA based state space representation gives good results when using the more noisy real world data instead of simulation based data. **b)** The achievable performance does not differ by a large margin between simulation and real world data based classifiers. **c)** The applicability of simulation trained classifiers in real world settings, which can help to avoid expensive real world data collection. **d)** Bootstrapping works not only in simulation but also in real world settings.

VII. Discussion and Conclusion

In this paper we demonstrated the learning of affordance classifiers for means-end actions and investigated how this learning process can be accelerated via bootstrapping. We have presented a method to capture meaningful categorical features of the environment like one object being 'on top' and/or 'inside' another. We have showcased a heuristic to limit this extraction of categories to the ones most likely to be useful to the robot for bootstrapping the learning of affordance predictors. To quantify the bootstrapping performance we have introduced a bootstrap factor. And finally, we have presented

results for bootstrapping the learning of affordance classifiers for nine means-end actions using a variety of existing affordance predictors and category predictors as knowledge sources for bootstrapping.

Our results show that there is no significant difference between bootstrapping with the PCA based feature space versus not bootstrapping with the Relational Histogram Features (RHF). By generating sufficiently sophisticated features (as in case of RHF), the learning problem can be made simple, so that Random Forests can solve the task with very few learning cycles. However, if the feature space is more basic (as in case of the PCA based features), then bootstrapping results in a significant speed up.

Designing a good feature space can be difficult; one needs to have an intuition about the relationships among elements of low level sensor data which discriminate the relevant classes. Furthermore one would ideally like a robot to be able to extend its affordance knowledge during lifelong learning, where there might be no human available to design a feature space. In many applications in computer vision, the approach of learning features works (e.g. via deep learning), because there are large sets of data available. In robot affordance applications such data is not readily available and costly (in time) to acquire. Additionally the PCA feature approach has an advantage over the RHF approach in implementation. The space requirements for processing are reduced.

The recommendations that emerge from our study are as follows. If one cannot design a good feature space and learning is slow, then direct bootstrapping (**DB**) is preferred in general. For example Figs. 6 and 8 show that bootstrapping in the PCA space can achieve accuracies after 25 samples that take 1000 samples without bootstrapping. In our case, and most conceivable cases, the time taken to train a single classifier is very small relative to the time to generate action experience; therefore it is feasible to train in parallel with different bootstrapping inputs, and then after a small number of samples are processed, to make a decision about which input to continue with, as in Sec. V-H. We have shown (Fig. 10) that this approach is superior to bootstrapping with all inputs and letting the random forest do the selection (see Sec. V-D). Although in our case category based bootstrapping (**CB**) was not faster than direct bootstrapping (**DB**), we see the ability to extract such categories autonomously as an important achievement since in a larger cognitive system such categorical knowledge can be exploited in different reasoning contexts (as outlined below).

We believe that bootstrapping is very important for affordance learning because there is a great deal to be learnt in order to achieve a basic level of manipulation competence in everyday environments, such as a child has. Spatial relationships are only one small segment of the common sense knowledge that is required (see [8]). Also, it typically takes a large amount of data to ground knowledge in a robot's own actions, and this data is usually hard to generate (i.e. through time consuming real world experiments). Techniques that can reduce the requirement for data are therefore beneficial.

With our category predictors, e.g., the category of being 'on top', or 'beside', we have also presented a simple form of extracting more symbolic descriptions of the environment. We avoid calling these *concepts* because a concept suggests a complex package of information, e.g. knowledge of situations or associated actions (see Barsalou [51], [52]), whereas we are talking about something more restricted: a classifier determining the presence of a critical aspect of a scene, e.g. spatial relationship category. The next logical step would be to use these categories within a larger cognitive architecture, where this symbolic knowledge is not only used for bootstrapping of new affordances, but also to guide future interactions and for the development of higher cognitive competences. Categories form a first step on the path to high level concepts and a robot which has recently acquired an 'on top' or 'inside' category symbol, for example, may be motivated to generate further experience around these categories, which could lead to a new phase of development that facilitates to the formation of higher level concepts like 'carrying' or 'stacking' or 'containment'.

REFERENCES

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

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

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

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

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

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

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

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

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

[10] H. Chaput, "The Constructivist Learning Architecture: A Model of Cognitive Development for Robust Autonomous Robots," *PhD Thesis, University of Texas at Austin, Artificial Intelligence Laboratory*, 2004.

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

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

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

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

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

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

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

[18] T. Zimmerman and S. Kambhampati, "Learning-assisted automated planning: Looking back, taking stock, going forward," *AI MAGAZINE*, vol. 24, p. 2003, 2003.
[19] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Symbol acquisition for probabilistic high-level planning," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.
[20] J. Mugan and B. Kuipers, "Autonomous learning of high-level states and actions in continuous environments," *IEEE Trans. Autonomous Mental Development*, vol. 4, no. 1, pp. 70–86, 2012.
[21] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, "Learning relational affordance models for robots in multi-object manipulation tasks," in *IEEE Intl. Conf. on Robotics and Automation*, 2012, pp. 4373–4378.
[22] S. Griffith, J. Sinapov, V. Sukhoy, and A. Stoytchev, "A behavior-grounded approach to forming object categories: Separating containers from noncontainers," *Autonomous Mental Development, IEEE Transactions on*, vol. 4, no. 1, pp. 54–69, March 2012.
[23] B. Rosman and S. Ramamoorthy, "Learning spatial relationships between objects," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1328–1342, sep 2011.
[24] S. Panda, A. H. A. Hafez, and C. V. Jawahar, "Learning support order for manipulation in clutter," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, nov 2013, pp. 809–815.
[25] S. Fichtl, J. Alexander, D. Kraft, J. Jorgensen, N. Krüger, and F. Guerin, "Learning object relationships which determine the outcome of actions," *Paladyn*, vol. 3, no. 4, pp. 188–199, 2012.
[26] H. Grabner, J. Gall, and L. Van Gool, "What makes a chair a chair?" in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, June 2011, pp. 1529–1536.
[27] E. Aksoy, A. Abramov, F. Worgotter, and B. Dellen, "Categorizing object-action relations from semantic scene graphs," in *IEEE International Conference on Robotics and Automation*, May 2010, pp. 398–405.
[28] H. S. Koppula, R. Gupta, and A. Saxena, "Learning human activities and object affordances from rgb-d videos," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 951–970, 2013.
[29] A. Lazaric, "Transfer in reinforcement learning: A framework and a survey," in *Reinforcement Learning: State of the Art*, ser. Adaptation, Learning, and Optimization, M. Wiering and M. van Otterlo, Eds. Springer Berlin Heidelberg, 2012, pp. 143–173.
[30] M. H. Lee, Q. Meng, and F. Chao, "Staged competence learning in developmental robotics," *Adaptive Behavior*, vol. 15, no. 3, pp. 241–255, 2007.
[31] S. Hart and R. Grupen, "Learning generalizable control programs," *IEEE Trans. Autonomous Mental Development*, vol. 3, no. 3, pp. 216–231, sept. 2011.
[32] P.-Y. Oudeyer, F. Kaplan, and V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 265–286, 2007.
[33] P.-Y. Oudeyer, A. Baranes, and F. Kaplan, "Intrinsically motivated learning of real-world sensorimotor skills with developmental constraints," in *Intrinsically Motivated Learning in Natural and Artificial Systems*, G. Baldassarre and M. Mirolli, Eds. Springer Berlin Heidelberg, 2013, pp. 303–365.
[34] E. Ugur and J. Piater, "Emergent structuring of interdependent affordance learning tasks," in *Development and Learning and Epigenetic Robotics (ICDL-Epirob), 2014 Joint IEEE International Conferences on*, Oct 2014, pp. 489–494.
[35] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese, "Understanding indoor scenes using 3d geometric phrases," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.
[36] G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "Constructing symbolic representations for high-level planning," 2014.
[37] G. Konidaris, "Constructing abstraction hierarchies using a skill-symbol loop," 2016.
[38] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[39] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.
[40] L. Breiman and A. Cutler, "Random Forests," *https://www.stat.berkeley.edu/˜breiman/RandomForests/cc_home.htm*, 2004.
[41] J. A. Jørgensen, L.-P. Ellekilde, and H. G. Petersen, "RobWorkSim - an Open Simulator for Sensor based Grasping," in *ISR/ROBOTIK 2010 (41st International Symposium)*. VDE-Verlag, Jun. 2010.
[42] K. Khoshelham and S. O. Elberink, "Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012.
[43] S. Olesen, S. Lyder, D. Kraft, N. Krüger, and J. Jessen, "Real-time extraction of surface patches with associated uncertainties by means of Kinect cameras," *Journal of Real-Time Image Processing*, vol. 10, no. 1, pp. 105–118, 2012.
[44] C. V. Nguyen, S. Izadi, and D. Lovell, "Modeling kinect sensor noise for improved 3d reconstruction and tracking," in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization Transmission*, Oct 2012, pp. 524–530.
[45] K. M. Varadarajan and M. Vincze, "Object part segmentation and classification in range images for grasping," in *Advanced Robotics (ICAR), 2011 15th International Conference on*, jun 2011, pp. 21–27.
[46] M. Schoeler, J. Papon, and F. Wörgötter, "Constrained Planar Cuts-Object Partitioning for Point Clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5207–5215.
[47] W. Mustafa, N. Pugeault, and N. Krüger, "Multi-View Object Recognition using View-Point Invariant Shape Relations and Appearance Information," in *IEEE Conf. on Robotics & Automation*, 2013.
[48] S. Fichtl, A. McManus, W. Mustafa, D. Kraft, N. Kruger, and F. Guerin, "Learning spatial relationships from 3D vision using histograms," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, may 2014, pp. 501–508.
[49] S. Fichtl, D. Kraft, N. Krüger, and F. Guerin, "Using Relational Histogram Features and Action Labelled Data to Learn Preconditions for Means-End Actions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (Workshop on Sensorimotor Contingencies for Robotics)*, Hamburg, 2015.
[50] R. E. Bellman, "Adaptive control processes: a guided tour," *Princeton University*, 1961.
[51] W. Yeh and L. W. Barsalou, "The Situated Nature of Concepts," *The American Journal of Psychology*, vol. 119, no. 3, pp. pp. 349–384, 2006.
[52] L. W. Barsalou, "Simulation, situated conceptualization, and prediction," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 364, no. 1521, pp. 1281–1289, 2009.

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

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

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

Frank Guerin obtained his Ph.D. degree from Imperial College, London, in 2002. Since August 2003, he has been a Lecturer in Computing Science at the University of Aberdeen. He is interested in Artificial Intelligence and Cognitive Systems, especially developmental approaches, and transferable knowledge.